
TensorForce Documentation

Release 0.3.3

reinforce.io

Mar 18, 2018

Contents:

1	Quick start	3
1.1	Agent and model overview	4
1.2	Environments	15
1.3	Preprocessing	19
1.4	TensorForce: Details for "summary_spec" agent parameters	21
1.5	Runners	23
1.6	tensorforce package	26
2	More information	361
	Python Module Index	363

TensorForce is an open source reinforcement learning library focused on providing clear APIs, readability and modularisation to deploy reinforcement learning solutions both in research and practice. TensorForce is built on top on TensorFlow.

CHAPTER 1

Quick start

For a quick start, you can run one of our example scripts using the provided configurations, e.g. to run the TRPO agent on CartPole, execute from the examples folder:

```
python examples/openai_gym.py CartPole-v0 -a examples/configs/ppo.json -n examples/  
↪ configs/mlp2_network.json
```

In python, it could look like this:

```
# examples/quickstart.py

import numpy as np

from tensorforce.agents import PPOAgent
from tensorforce.execution import Runner
from tensorforce.contrib.openai_gym import OpenAIGym

# Create an OpenAIGym environment
env = OpenAIGym('CartPole-v0', visualize=True)

# Network as list of layers
network_spec = [
    dict(type='dense', size=32, activation='tanh'),
    dict(type='dense', size=32, activation='tanh')
]

agent = PPOAgent(
    states_spec=env.states,
    actions_spec=env.actions,
    network_spec=network_spec,
    batch_size=4096,
    # BatchAgent
    keep_last_timestep=True,
    # PPOAgent
    step_optimizer=dict(
        type='adam',
```

```
        learning_rate=1e-3
    ),
    optimization_steps=10,
    # Model
    scope='ppo',
    discount=0.99,
    # DistributionModel
    distributions_spec=None,
    entropy_regularization=0.01,
    # PGModel
    baseline_mode=None,
    baseline=None,
    baseline_optimizer=None,
    gae_lambda=None,
    # PGLRModel
    likelihood_ratio_clipping=0.2,
    summary_spec=None,
    distributed_spec=None
)

# Create the runner
runner = Runner(agent=agent, environment=env)

# Callback function printing episode statistics
def episode_finished(r):
    print("Finished episode {ep} after {ts} timesteps (reward: {reward})".format(ep=r.
↪episode, ts=r.episode_timestep,
↪reward=r.episode_rewards[-1]))
    return True

# Start learning
runner.run(episodes=3000, max_episode_timesteps=200, episode_finished=episode_
↪finished)
runner.close()

# Print statistics
print("Learning finished. Total episodes: {ep}. Average reward of last 100 episodes:
↪{ar}.".format(
    ep=runner.episode,
    ar=np.mean(runner.episode_rewards[-100:]))
)
```

1.1 Agent and model overview

A reinforcement learning agent provides methods to process states and return actions, to store past observations, and to load and save models. Most agents employ a `Model` which implements the algorithms to calculate the next action given the current state and to update model parameters from past experiences.

Environment <-> Runner <-> Agent <-> Model

Parameters to the agent are passed as a `Configuration` object. The configuration is passed on to the `Model`.

1.1.1 Ready-to-use algorithms

We implemented some of the most common RL algorithms and try to keep these up-to-date. Here we provide an overview over all implemented agents and models.

Agent / General parameters

Agent is the base class for all reinforcement learning agents. Every agent inherits from this class.

```
class tensorflow.agents.Agent (states, actions, batched_observe=True, batching_capacity=1000)
```

Bases: object

Base class for TensorForce agents.

```
__init__ (states, actions, batched_observe=True, batching_capacity=1000)
```

Initializes the agent.

Parameters **states** – States specification, with the following attributes (required):

Parameters **actions** – Actions specification, with the following attributes (required):

Parameters

- **batched_observe** (*bool*) – Specifies whether calls to `model.observe()` are batched, for improved performance (default: `true`).
- **batching_capacity** (*int*) – Batching capacity of agent and model (default: 1000).

```
act (states, deterministic=False, independent=False, fetch_tensors=None)
```

Return action(s) for given state(s). States preprocessing and exploration are applied if configured accordingly.

Parameters

- **states** (*any*) – One state (usually a value tuple) or dict of states if multiple states are expected.
- **deterministic** (*bool*) – If true, no exploration and sampling is applied.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).
- **fetch_tensors** (*list*) – Optional String of named tensors to fetch

Returns Scalar value of the action or dict of multiple actions the agent wants to execute. (fetched_tensors) Optional dict() with named tensors fetched

```
static from_spec (spec, kwargs)
```

Creates an agent from a specification dict.

```
initialize_model ()
```

Creates the model for the respective agent based on specifications given by user. This is a separate call after constructing the agent because the agent constructor has to perform a number of checks on the specs first, sometimes adjusting them e.g. by converting to a dict.

```
observe (terminal, reward)
```

Observe experience from the environment to learn from. Optionally pre-processes rewards. Child classes should call super to get the processed reward EX: `terminal, reward = super()....`

Parameters

- **terminal** (*bool*) – boolean indicating if the episode terminated after the observation.
- **reward** (*float*) – scalar reward that resulted from executing the action.

reset ()

Reset the agent to its initial state (e.g. on experiment start). Updates the Model's internal episode and time step counter, internal states, and resets preprocessors.

restore_model (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

save_model (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** (*str*) – Optional checkpoint directory.
- **append_timestep** (*bool*) – Appends the current timestep to the checkpoint file if true. If this is set to True, the load path must include the checkpoint timestep suffix. For example, if stored to models/ and set to true, the exported file will be of the form models/model.ckpt-X where X is the last timestep saved. The load path must precisely match this file name. If this option is turned off, the checkpoint will always overwrite the file specified in path and the model can always be loaded under this path.

Returns Checkpoint path where the model was saved.

Model

The `Model` class is the base class for reinforcement learning models.

```
class tensorflow.models.Model (states, actions, scope, device, saver, summarizer,  
                                execution, batching_capacity, variable_noise,  
                                states_preprocessing, actions_exploration, re-  
                                ward_preprocessing)
```

Bases: `object`

Base class for all (TensorFlow-based) models.

```
__init__ (states, actions, scope, device, saver, summarizer, execution, batch-  
          ing_capacity, variable_noise, states_preprocessing, actions_exploration,  
          reward_preprocessing)
```

Model.

Parameters

- **states** (*spec*) – The state-space description dictionary.
- **actions** (*spec*) – The action-space description dictionary.
- **scope** (*str*) – The root scope str to use for tf variable scoping.
- **device** (*str*) – The name of the device to run the graph of this model on.
- **saver** (*spec*) – Dict specifying whether and how to save the model's parameters.
- **summarizer** (*spec*) – Dict specifying which tensorboard summaries should be created and added to the graph.
- **execution** (*spec*) – Dict specifying whether and how to do distributed training on the model's graph.
- **batching_capacity** (*int*) – Batching capacity.

- **variable_noise** (*float*) – The stddev value of a Normal distribution used for adding random noise to the model's output (for each batch, noise can be toggled and - if active - will be resampled). Use None for not adding any noise.
- **states_preprocessing** (*spec / dict of specs*) – Dict specifying whether and how to preprocess state signals (e.g. normalization, greyscale, etc..).
- **actions_exploration** (*spec / dict of specs*) – Dict specifying whether and how to add exploration to the model's "action outputs" (e.g. epsilon-greedy).
- **reward_preprocessing** (*spec*) – Dict specifying whether and how to preprocess rewards coming from the Environment (e.g. reward normalization).

act (*states, internals, deterministic=False, independent=False, fetch_tensors=None*)

Does a forward pass through the model to retrieve action (outputs) given inputs for state (and internal state, if applicable (e.g. RNNs))

Parameters

- **states** (*dict*) – Dict of state values (each key represents one state space component).
- **internals** (*dict*) – Dict of internal state values (each key represents one internal state component).
- **deterministic** (*bool*) – If True, will not apply exploration after actions are calculated.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).

Returns

- Actual action-outputs (batched if state input is a batch).

Return type tuple

create_operations (*states, internals, actions, terminal, reward, deterministic, independent*)

Creates output operations for acting, observing and interacting with the memory.

get_component (*component_name*)

Looks up a component by its name.

Parameters **component_name** – The name of the component to look up.

Returns The component for the provided name or None if there is no such component.

get_components ()

Returns a dictionary of component name to component of all the components within this model.

Returns (dict) The mapping of name to component.

get_savable_components ()

Returns the list of all of the components this model consists of that can be individually saved and restored. For instance the network or distribution.

Returns List of util.SavableComponent

get_summaries ()

Returns the TensorFlow summaries reported by the model

Returns List of summaries

get_variables (*include_submodules=False, include_nontrainable=False*)

Returns the TensorFlow variables used by the model.

Parameters

- **include_submodules** – Includes variables of submodules (e.g. baseline, target network) if true.
- **include_nontrainable** – Includes non-trainable variables if true.

Returns List of variables.

initialize (*custom_getter*)

Creates the TensorFlow placeholders and functions for this model. Moreover adds the internal state placeholders and initialization values to the model.

Parameters **custom_getter** – The `custom_getter_` object to use for `tf.make_template` when creating TensorFlow functions.

observe (*terminal, reward*)

Adds an observation (reward and is-terminal) to the model without updating its trainable variables.

Parameters

- **terminal** (*bool*) – Whether the episode has terminated.
- **reward** (*float*) – The observed reward value.

Returns The value of the model-internal episode counter.

reset ()

Resets the model to its initial state on episode start. This should also reset all preprocessor(s).

Returns Current episode, timestep counter and the shallow-copied list of internal state initialization Tensors.

Return type tuple

restore (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

restore_component (*component_name, save_path*)

Restores a component's parameters from a save location.

Parameters

- **component_name** – The component to restore.
- **save_path** – The save location.

save (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** – Optional checkpoint directory.
- **append_timestep** – Appends the current timestep to the checkpoint file if true.

Returns Checkpoint path where the model was saved.

save_component (*component_name, save_path*)

Saves a component of this model to the designated location.

Parameters

- **component_name** – The component to save.
- **save_path** – The location to save to.

Returns Checkpoint path where the component was saved.

setup ()

Sets up the TensorFlow model graph and initializes (and enters) the TensorFlow session.

tf_action_exploration (*action, exploration, action_spec*)

Applies optional exploration to the action (post-processor for action outputs).

Parameters

- **action** (*tf.Tensor*) – The original output action tensor (to be post-processed).
- **exploration** (*Exploration*) – The Exploration object to use.
- **action_spec** (*dict*) – Dict specifying the action space.

Returns The post-processed action output tensor.

tf_actions_and_internals (*states, internals, deterministic*)

Creates and returns the TensorFlow operations for retrieving the actions and - if applicable - the posterior internal state Tensors in reaction to the given input states (and prior internal states).

Parameters

- **states** (*dict*) – Dict of state tensors (each key represents one state space component).
- **internals** – List of prior internal state tensors.
- **deterministic** – Boolean tensor indicating whether action should be chosen deterministically.

Returns

1. dict of output actions (with or without exploration applied (see *deterministic*))
2. list of posterior internal state Tensors (empty for non-internal state models)

Return type tuple

tf_observe_timestep (*states, internals, actions, terminal, reward*)

Creates the TensorFlow operations for performing the observation of a full time step's information.

Parameters

- **states** (*dict*) – Dict of state tensors (each key represents one state space component).
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.

Returns The observation operation.

MemoryAgent

BatchAgent

Deep-Q-Networks (DQN)

```
class tensorflow.agents.DQNAgent (states, actions, network, batched_observe=True,
                                batching_capacity=1000, scope='dqn',
                                device=None, saver=None, summarizer=None,
                                execution=None, variable_noise=None,
                                states_preprocessing=None, actions_exploration=None,
                                reward_preprocessing=None, update_mode=None,
                                memory=None, optimizer=None, discount=0.99,
                                distributions=None, entropy_regularization=None,
                                target_sync_frequency=10000, target_update_weight=1.0,
                                double_q_model=False, huber_loss=None)
```

Bases: `tensorflow.agents.learning_agent.LearningAgent`

Deep Q-Network agent (Mnih et al., 2015).

```
__init__ (states, actions, network, batched_observe=True, batching_capacity=1000,
          scope='dqn', device=None, saver=None, summarizer=None, execution=None,
          variable_noise=None, states_preprocessing=None, actions_exploration=None,
          reward_preprocessing=None, update_mode=None, memory=None, optimizer=None,
          discount=0.99, distributions=None, entropy_regularization=None,
          target_sync_frequency=10000, target_update_weight=1.0, double_q_model=False,
          huber_loss=None)
```

Initializes the DQN agent.

Parameters `update_mode` – Update mode specification, with the following attributes:

Parameters

- **memory** (*spec*) – Memory specification, see `core.memories` module for more information (default: `{type='replay', include_next_states=True, capacity=1000*batch_size}`).
- **optimizer** (*spec*) – Optimizer specification, see `core.optimizers` module for more information (default: `{type='adam', learning_rate=1e-3}`).
- **target_sync_frequency** (*int*) – Target network sync frequency (default: 10000).
- **target_update_weight** (*float*) – Target network update weight (default: 1.0).
- **double_q_model** (*bool*) – Specifies whether double DQN mode is used (default: false).
- **huber_loss** (*float*) – Huber loss clipping (default: none).

Normalized Advantage Functions

```
class tensorflow.agents.NAFAgent (states, actions, network, batched_observe=True,
                                batching_capacity=1000, scope='naf',
                                device=None, saver=None, summarizer=None,
                                execution=None, variable_noise=None,
                                states_preprocessing=None, actions_exploration=None,
                                reward_preprocessing=None, update_mode=None,
                                memory=None, optimizer=None, discount=0.99,
                                distributions=None, entropy_regularization=None,
                                target_sync_frequency=10000, target_update_weight=1.0,
                                double_q_model=False, huber_loss=None)
```

Bases: `tensorflow.agents.learning_agent.LearningAgent`

Normalized Advantage Function agent (Gu et al., 2016).

```
__init__ (states, actions, network, batched_observe=True, batching_capacity=1000,
          scope='naf', device=None, saver=None, summarizer=None, execution=None,
          variable_noise=None, states_preprocessing=None, actions_exploration=None,
          reward_preprocessing=None, update_mode=None, memory=None, optimizer=None,
          discount=0.99, distributions=None, entropy_regularization=None,
          target_sync_frequency=10000, target_update_weight=1.0, double_q_model=False,
          huber_loss=None)
```

Initializes the NAF agent.

Parameters `update_mode` – Update mode specification, with the following attributes:

Parameters

- **memory** (*spec*) – Memory specification, see `core.memories` module for more information (default: `{type='replay', include_next_states=True, capacity=1000*batch_size}`).
- **optimizer** (*spec*) – Optimizer specification, see `core.optimizers` module for more information (default: `{type='adam', learning_rate=1e-3}`).
- **target_sync_frequency** (*int*) – Target network sync frequency (default: 10000).
- **target_update_weight** (*float*) – Target network update weight (default: 1.0).
- **double_q_model** (*bool*) – Specifies whether double DQN mode is used (default: false).
- **huber_loss** (*float*) – Huber loss clipping (default: none).

Deep-Q-learning from demonstration (DQFD)

```
class tensorforce.agents.DQFDAgent (states,          actions,          network,
                                   batched_observe=True,          batching_capacity=1000,
                                   scope='dqfd', device=None, saver=None, summarizer=None,
                                   execution=None, variable_noise=None,
                                   states_preprocessing=None, actions_exploration=None,
                                   reward_preprocessing=None, update_mode=None,
                                   memory=None, optimizer=None, discount=0.99,
                                   distributions=None, entropy_regularization=None,
                                   target_sync_frequency=10000, target_update_weight=1.0,
                                   huber_loss=None, expert_margin=0.5, supervised_weight=0.1,
                                   demo_memory_capacity=10000, demo_sampling_ratio=0.2)
```

Bases: `tensorforce.agents.learning_agent.LearningAgent`

Deep Q-learning from demonstration agent (Hester et al., 2017).

```
__init__(states, actions, network, batched_observe=True, batching_capacity=1000,
          scope='dqfd', device=None, saver=None, summarizer=None, execution=None,
          variable_noise=None, states_preprocessing=None, actions_exploration=None,
          reward_preprocessing=None, update_mode=None, memory=None, optimizer=None,
          discount=0.99, distributions=None, entropy_regularization=None,
          target_sync_frequency=10000, target_update_weight=1.0, huber_loss=None,
          expert_margin=0.5, supervised_weight=0.1, demo_memory_capacity=10000,
          demo_sampling_ratio=0.2)
```

Initializes the DQFD agent.

Parameters `update_mode` – Update mode specification, with the following attributes:

Parameters

- **memory** (*spec*) – Memory specification, see `core.memories` module for more information (default: `{type='replay', include_next_states=True, capacity=1000*batch_size}`).
- **optimizer** (*spec*) – Optimizer specification, see `core.optimizers` module for more information (default: `{type='adam', learning_rate=1e-3}`).
- **target_sync_frequency** (*int*) – Target network sync frequency (default: 10000).
- **target_update_weight** (*float*) – Target network update weight (default: 1.0).
- **huber_loss** (*float*) – Huber loss clipping (default: none).
- **expert_margin** (*float*) – Enforced supervised margin between expert action Q-value and other Q-values (default: 0.5).
- **supervised_weight** (*float*) – Weight of supervised loss term (default: 0.1).
- **demo_memory_capacity** (*int*) – Capacity of expert demonstration memory (default: 10000).
- **demo_sampling_ratio** (*float*) – Runtime sampling ratio of expert data (default: 0.2).

```
import demonstrations (demonstrations)
```

Imports demonstrations, i.e. expert observations. Note that for large numbers of observations,

`set_demonstrations` is more appropriate, which directly sets memory contents to an array and expects a different layout.

Parameters `demonstrations` – List of observation dicts

pretrain (*steps*)

Computes pre-train updates.

Parameters `steps` – Number of updates to execute.

Vanilla Policy Gradient

```
class tensorflowforce.agents.VPGAgent (states, actions, network, batched_observe=True,
                                       batching_capacity=1000, scope='vpg',
                                       device=None, saver=None, summarizer=None, execution=None, variable_noise=None,
                                       states_preprocessing=None, actions_exploration=None, reward_preprocessing=None,
                                       update_mode=None, memory=None, optimizer=None, discount=0.99, distributions=None,
                                       entropy_regularization=None, baseline_mode=None, baseline=None, baseline_optimizer=None,
                                       gae_lambda=None)
```

Bases: `tensorflowforce.agents.learning_agent.LearningAgent`

Vanilla policy gradient agent (Williams, 1992)).

```
__init__ (states, actions, network, batched_observe=True, batching_capacity=1000,
          scope='vpg', device=None, saver=None, summarizer=None, execution=None, variable_noise=None,
          states_preprocessing=None, actions_exploration=None, reward_preprocessing=None, update_mode=None,
          memory=None, optimizer=None, discount=0.99, distributions=None, entropy_regularization=None,
          baseline_mode=None, baseline=None, baseline_optimizer=None, gae_lambda=None)
```

Initializes the VPG agent.

Parameters `update_mode` – Update mode specification, with the following attributes:

Parameters

- **memory** (*spec*) – Memory specification, see `core.memories` module for more information (default: `{type='latest', include_next_states=False, capacity=1000*batch_size}`).
- **optimizer** (*spec*) – Optimizer specification, see `core.optimizers` module for more information (default: `{type='adam', learning_rate=1e-3}`).
- **baseline_mode** (*str*) – One of 'states', 'network' (default: none).
- **baseline** (*spec*) – Baseline specification, see `core.baselines` module for more information (default: none).
- **baseline_optimizer** (*spec*) – Baseline optimizer specification, see `core.optimizers` module for more information (default: none).
- **gae_lambda** (*float*) – Lambda factor for generalized advantage estimation (default: none).

Trust Region Policy Optimization (TRPO)

```
class tensorflow.agents.TRPOAgent (states,          actions,          network,
                                   batched_observe=True,          batching_capacity=1000,
                                   scope='trpo', device=None, saver=None, summarizer=None,
                                   execution=None,          variable_noise=None,
                                   states_preprocessing=None,          actions_exploration=None,
                                   reward_preprocessing=None,          update_mode=None,
                                   memory=None, discount=0.99,          distributions=None,
                                   entropy_regularization=None,          baseline_mode=None,
                                   baseline=None, baseline_optimizer=None,
                                   gae_lambda=None, likelihood_ratio_clipping=None,
                                   learning_rate=0.001,          cg_max_iterations=20,
                                   cg_damping=0.001,          cg_unroll_loop=False,
                                   ls_max_iterations=10,          ls_accept_ratio=0.9,
                                   ls_unroll_loop=False)
```

Bases: `tensorflow.agents.learning_agent.LearningAgent`

Trust Region Policy Optimization agent (Schulman et al., 2015).

```
__init__ (states, actions, network, batched_observe=True, batching_capacity=1000,
           scope='trpo', device=None, saver=None, summarizer=None, execution=None,
           variable_noise=None, states_preprocessing=None, actions_exploration=None,
           reward_preprocessing=None,          update_mode=None,          memory=None,
           discount=0.99,          distributions=None,          entropy_regularization=None,
           baseline_mode=None,          baseline=None,          baseline_optimizer=None,
           gae_lambda=None, likelihood_ratio_clipping=None, learning_rate=0.001,
           cg_max_iterations=20,          cg_damping=0.001,          cg_unroll_loop=False,
           ls_max_iterations=10, ls_accept_ratio=0.9, ls_unroll_loop=False)
```

Initializes the TRPO agent.

Parameters `update_mode` – Update mode specification, with the following attributes:

Parameters

- **memory** (*spec*) – Memory specification, see `core.memories` module for more information (default: `{type='latest', include_next_states=false, capacity=1000*batch_size}`).
- **optimizer** (*spec*) – TRPO agent implicitly defines a optimized-step natural-gradient optimizer.
- **baseline_mode** (*str*) – One of 'states', 'network' (default: none).
- **baseline** (*spec*) – Baseline specification, see `core.baselines` module for more information (default: none).
- **baseline_optimizer** (*spec*) – Baseline optimizer specification, see `core.optimizers` module for more information (default: none).
- **gae_lambda** (*float*) – Lambda factor for generalized advantage estimation (default: none).
- **likelihood_ratio_clipping** (*float*) – Likelihood ratio clipping for policy gradient (default: none).
- **learning_rate** (*float*) – Learning rate of natural-gradient optimizer (default: `1e-3`).
- **cg_max_iterations** (*int*) – Conjugate-gradient max iterations (default: 20).

- **cg_damping** (*float*) – Conjugate-gradient damping (default: 1e-3).
- **cg_unroll_loop** (*bool*) – Conjugate-gradient unroll loop (default: false).
- **ls_max_iterations** (*int*) – Line-search max iterations (default: 10).
- **ls_accept_ratio** (*float*) – Line-search accept ratio (default: 0.9).
- **ls_unroll_loop** (*bool*) – Line-search unroll loop (default: false).

1.1.2 State preprocessing

The agent handles state preprocessing. A preprocessor takes the raw state input from the environment and modifies it (for instance, image resize, state concatenation, etc.). You can find information about our ready-to-use preprocessors [here](#).

1.1.3 Building your own agent

If you want to build your own agent, it should always inherit from `Agent`. If your agent uses a replay memory, it should probably inherit from `MemoryAgent`, if it uses a batch replay that is emptied after each update, it should probably inherit from `BatchAgent`.

We distinguish between agents and models. The `Agent` class handles the interaction with the environment, such as state preprocessing, exploration and observation of rewards. The `Model` class handles the mathematical operations, such as building the tensorflow operations, calculating the desired action and updating (i.e. optimizing) the model weights.

To start building your own agent, please refer to [this blogpost](#) to gain a deeper understanding of the internals of the TensorForce library. Afterwards, have look on a sample implementation, e.g. the [DQN Agent](#) and [DQN Model](#).

1.2 Environments

A reinforcement learning environment provides the API to a simulated or real environment as the subject for optimization. It could be anything from video games (e.g. Atari) to robots or trading systems. The agent interacts with this environment and learns to act optimally in its dynamics.

Environment <-> Runner <-> Agent <-> Model

class `tensorflow.environments.Environment`

Base environment class.

actions

Return the action space. Might include subdicts if multiple actions are available simultaneously.

Returns: dict of action properties (continuous, number of actions)

close()

Close environment. No other method calls possible afterwards.

execute (*actions*)

Executes action, observes next state(s) and reward.

Parameters **actions** – Actions to execute.

Returns (Dict of) next state(s), boolean indicating terminal, and reward signal.

static from_spec (*spec, kwargs*)

Creates an environment from a specification dict.

reset()

Reset environment and setup for new episode.

Returns initial state of reset environment.

seed (*seed*)

Sets the random seed of the environment to the given value (current time, if *seed*=None). Naturally deterministic Environments (e.g. ALE or some gym Envs) don't have to implement this method.

Parameters *seed* (*int*) – The seed to use for initializing the pseudo-random number generator (default=epoch time in sec).

Returns: The actual seed (int) used OR None if Environment did not override this method (no seeding supported).

states

Return the state space. Might include subdicts if multiple states are available simultaneously.

Returns: dict of state properties (shape and type).

1.2.1 Ready-to-use environments

OpenAI Gym

```
class tensorflow.contrib.openai_gym.OpenAIGym (gym_id,          monitor=None,  
                                              monitor_safe=False,  
                                              monitor_video=0,      visual-  
                                              ize=False)
```

Bases: *tensorflow.environments.environment.Environment*

```
__init__ (gym_id, monitor=None, monitor_safe=False, monitor_video=0, visual-  
          ize=False)
```

Initialize OpenAI Gym.

Parameters

- **gym_id** – OpenAI Gym environment ID. See <https://gym.openai.com/envs>
- **monitor** – Output directory. Setting this to None disables monitoring.
- **monitor_safe** – Setting this to True prevents existing log files to be overwritten. Default False.
- **monitor_video** – Save a video every *monitor_video* steps. Setting this to 0 disables recording of videos.
- **visualize** – If set True, the program will visualize the trainings of gym's environment. Note that such visualization is probably going to slow down the training.

OpenAI Universe

```
class tensorflow.contrib.openai_universe.OpenAIUniverse (env_id)
```

Bases: *tensorflow.environments.environment.Environment*

OpenAI Universe Integration: <https://universe.openai.com/>. Contains OpenAI Gym: <https://gym.openai.com/>.

```
__init__ (env_id)
```

Initialize OpenAI universe environment.

Parameters *env_id* – string with id/descriptor of the universe environment, e.g. 'HarvestDay-v0'.

Deepmind Lab

```
class tensorflow.contrib.deepmind_lab.DeepMindLab(level_id, repeat_action=1,
state_attribute='RGB_INTERLACED',
settings={'width': '320', 'appendCommand': '', 'fps': '60', 'height': '240'})
```

Bases: `tensorflow.environments.environment.Environment`

DeepMind Lab Integration: <https://arxiv.org/abs/1612.03801> <https://github.com/deepmind/lab>

Since DeepMind lab is only available as source code, a manual install via bazel is required. Further, due to the way bazel handles external dependencies, cloning TensorForce into lab is the most convenient way to run it using the bazel BUILD file we provide. To use lab, first download and install it according to instructions <https://github.com/deepmind/lab/blob/master/docs/build.md>:

```
git clone https://github.com/deepmind/lab.git
```

Add to the lab main BUILD file:

Clone TensorForce into the lab directory, then run the TensorForce bazel runner.

Note that using any specific configuration file currently requires changing the Tensorforce BUILD file to adjust environment parameters.

```
bazel run //tensorflow:lab_runner
```

Please note that we have not tried to reproduce any lab results yet, and these instructions just explain connectivity in case someone wants to get started there.

```
__init__(level_id, repeat_action=1, state_attribute='RGB_INTERLACED',
settings={'width': '320', 'appendCommand': '', 'fps': '60', 'height': '240'})
```

Initialize DeepMind Lab environment.

Parameters

- **level_id** – string with id/descriptor of the level, e.g. 'seekavoid_arena_01'.
- **repeat_action** – number of frames the environment is advanced, executing the given action during every frame.
- **state_attribute** – Attributes which represents the state for this environment, should adhere to the specification given in `DeepMindLabEnvironment.state_spec(level_id)`.
- **settings** – dict specifying additional settings as key-value string pairs. The following options are recognized: 'width' (horizontal resolution of the observation frames), 'height' (vertical resolution of the observation frames), 'fps' (frames per second) and 'appendCommand' (commands for the internal Quake console).

close()

Closes the environment and releases the underlying Quake III Arena instance. No other method calls possible afterwards.

execute(actions)

Pass action to universe environment, return reward, next step, terminal state and additional info.

Parameters **action** – action to execute as numpy array, should have dtype `np.intc` and should adhere to the specification given in `DeepMindLabEnvironment.action_spec(level_id)`

Returns dict containing the next state, the reward, and a boolean indicating if the next state is a terminal state

fps

An advisory metric that correlates discrete environment steps ("frames") with real (wallclock) time: the number of frames per (real) second.

num_steps

Number of frames since the last reset() call.

reset()

Resets the environment to its initialization state. This method needs to be called to start a new episode after the last episode ended.

Returns initial state

Unreal Engine 4 Games

```
class tensorflow.contrib.unreal_engine.UE4Environment (host='localhost',
                                                         port=6025, connect=True, discretize_actions=False,
                                                         delta_time=0, num_ticks=4)
```

Bases: `tensorflow.contrib.remote_environment.RemoteEnvironment`,
`tensorflow.contrib.state_settable_environment.StateSettableEnvironment`

A special RemoteEnvironment for UE4 game connections. Communicates with the remote to receive information on the definitions of action- and observation spaces. Sends UE4 Action- and Axis-mappings as RL-actions and receives observations back defined by MLObserver objects placed in the Game (these could be camera pixels or other observations, e.g. a x/y/z position of some game actor).

```
__init__ (host='localhost', port=6025, connect=True, discretize_actions=False,
          delta_time=0, num_ticks=4)
```

Parameters

- **host** (*str*) – The hostname to connect to.
- **port** (*int*) – The port to connect to.
- **connect** (*bool*) – Whether to connect already in this c'tor.
- **discretize_actions** (*bool*) – Whether to treat axis-mappings defined in UE4 game as discrete actions. This would be necessary e.g. for agents that use q-networks where the output are q-values per discrete state-action pair.
- **delta_time** (*float*) – The fake delta time to use for each single game tick.
- **num_ticks** (*int*) – The number of ticks to be executed in a single act call (each tick will repeat the same given actions).

discretize_action_space_desc()

Creates a list of discrete action(-combinations) in case we want to learn with a discrete set of actions, but only have action-combinations (maybe even continuous) available from the env. E.g. the UE4 game has the following action/axis-mappings:

```
{
  'Fire':
    {'type': 'action', 'keys': ('SpaceBar',)},
  'MoveRight':
    {'type': 'axis', 'keys': (('Right', 1.0), ('Left', -1.0), ('A', -
↪1.0), ('D', 1.0))},
}
```

-> this method will discretize them into the following 6 discrete actions:

```
[
  (Right, 0.0), (SpaceBar, False)],
  (Right, 0.0), (SpaceBar, True)]
  (Right, -1.0), (SpaceBar, False)],
  (Right, -1.0), (SpaceBar, True)],
  (Right, 1.0), (SpaceBar, False)],
  (Right, 1.0), (SpaceBar, True)],
]
```

execute (*actions*)

Executes a single step in the UE4 game. This step may be comprised of one or more actual game ticks for all of which the same given action- and axis-inputs (or action number in case of discretized actions) are repeated. UE4 distinguishes between action-mappings, which are boolean actions (e.g. jump or dont-jump) and axis-mappings, which are continuous actions like MoveForward with values between -1.0 (run backwards) and 1.0 (run forwards), 0.0 would mean: stop.

reset ()

same as step (no kwargs to pass), but needs to block and return `observation_dict`

- stores the received observation in `self.last_observation`

translate_abstract_actions_to_keys (*abstract*)

Translates a list of tuples ([pretty mapping], [value]) to a list of tuples ([some key], [translated value]) each single item in abstract will undergo the following translation:

Example1: we want: "MoveRight": 5.0 possible keys for the action are: ("Right", 1.0), ("Left", -1.0) result: "Right": $5.0 * 1.0 = 5.0$

Example2: we want: "MoveRight": -0.5 possible keys for the action are: ("Left", -1.0), ("Right", 1.0) result: "Left": $-0.5 * -1.0 = 0.5$ (same as "Right": -0.5)

1.3 Preprocessing

Often it is necessary to modify state input tensors before passing them to the reinforcement learning agent. This could be due to various reasons, e.g.:

- Feature scaling / input normalization,
- Data reduction,
- Ensuring the Markov property by concatenating multiple states (e.g. in Atari)

TensorForce comes with a number of ready-to-use preprocessors, a preprocessing stack and easy ways to implement your own preprocessors.

1.3.1 Usage

The

Each preprocessor implements three methods:

1. The constructor (`__init__`) for parameter initialization
2. `process(state)` takes a state and returns the processed state
3. `processed_shape(original_shape)` takes a shape and returns the processed shape

The preprocessing stack iteratively calls these functions of all preprocessors in the stack and returns the result.

Using one preprocessor

```
from tensorforce.core.preprocessing import Sequence

pp_seq = Sequence(4)  # initialize preprocessor (return sequence of last 4 states)

state = env.reset()  # reset environment
processed_state = pp_seq.process(state)  # process state
```

Using a preprocessing stack

You can stack multiple preprocessors:

```
from tensorforce.core.preprocessing import Preprocessing, Grayscale, Sequence

pp_gray = Grayscale()  # initialize grayscale preprocessor
pp_seq = Sequence(4)  # initialize sequence preprocessor

stack = Preprocessing()  # initialize preprocessing stack
stack.add(pp_gray)  # add grayscale preprocessor to stack
stack.add(pp_seq)  # add maximum preprocessor to stack

state = env.reset()  # reset environment
processed_state = stack.process(state)  # process state
```

Using a configuration dict

If you use configuration objects, you can build your preprocessing stack from a config:

```
from tensorforce.core.preprocessing import Preprocessing

preprocessing_config = [
    {
        "type": "image_resize",
        "width": 84,
        "height": 84
    }, {
        "type": "grayscale"
    }, {
        "type": "center"
    }, {
        "type": "sequence",
        "length": 4
    }
]

stack = Preprocessing.from_spec(preprocessing_config)
config.state_shape = stack.shape(config.state_shape)
```

The Agent class expects a *preprocessing* configuration parameter and then handles preprocessing automatically:

```
from tensorforce.agents import DQNAgent

agent = DQNAgent(config=dict(
    states=...,
```



```
actions=...,
preprocessing=preprocessing_config,
# ...
))
```

1.3.2 Ready-to-use preprocessors

These are the preprocessors that come with TensorForce:

Standardize

Grayscale

ImageResize

Normalize

Sequence

1.3.3 Building your own preprocessor

All preprocessors should inherit from `tensorforce.core.preprocessing.Preprocessor`.

For a start, please refer to the source of the [Grayscale preprocessor](#).

1.4 TensorForce: Details for "summary_spec" agent parameters

1.4.1 summarizer

TensorForce has the ability to record summary data for use with TensorBoard as well STDIO and file export. This is accomplished through dictionary parameter called "summarizer" passed to the agent on initialization.

"summarizer" supports the following optional dictionary entries:

Key	Value
directory	(str) Path to storage for TensorBoard summary data
steps	(int) Frequency in steps between storage of summary data
seconds	(int) Frequency in seconds to store summary data
labels	(list) Requested Export, See " <i>LABELS</i> " section
meta_dict	(dict) For used with label "configuration"

1.4.2 LABELS

Entry	Data produced
losses	Training total-loss and "loss-without-regularization"
total-loss	Final calculated loss value
variables	Network variables
inputs	Equivalent to: ['states', 'actions', 'rewards']
states	Histogram of input state space
actions	Histogram of input action space
rewards	Histogram of input reward space
gradients	Histogram and scalar gradients
gradients_histogram	Variable gradients as histograms
gradients_scalar	Variable Mean/Variance of gradients as scalar
regularization	Regularization values
configuration	See <i>Configuration Export</i> for more detail
configuration	Export configuration to "TEXT" tab in TensorBoard
print_configuration	Prints configuration to STDOUT

```
from tensorforce.agents import PPOAgent

# Create a Proximal Policy Optimization agent
agent = PPOAgent(
    states=...,
    actions=...,
    network=...,
    summarizer=dict(directory="./board/",
                    steps=50,
                    labels=['configuration',
                          'gradients_scalar',
                          'regularization',
                          'inputs',
                          'losses',
                          'variables']
    ),
    ...
)
```

1.4.3 Configuration Export

Adding the "configuration" label will create a "TEXT" tab in TensorBoard that contains all the parameters passed to the Agent. By using the additional "summarizer" dictionary key "meta_dict", custom keys and values can be added to the data export. The user may want to pass "Description", "Experiment #", "InputDataSet", etc.

If a key is already in use within TensorForce an error will be raised to notify you to change the key value. To use the custom feature, create a dictionary with keys to export:

```
from tensorforce.agents import PPOAgent

metaparams['MyDescription'] = "This experiment covers the first test ...."
metaparams['My2D'] = np.ones((9,9))    # 9x9 matrix of 1.0's
metaparams['My1D'] = np.ones((9))      # Column of 9 1.0's

# Create a Proximal Policy Optimization agent
agent = PPOAgent(
    states=...,
    actions=...,
    network=...,
    summarizer=dict(directory="./board/",
                     steps=50,
                     meta_dict=metaparams, #Add custom keys to export
                     labels=['configuration',
                             'gradients_scalar',
                             'regularization',
                             'inputs',
                             'losses',
                             'variables']
    ),
    ...
)
```

Use the "print_configuration" label to export the configuration data to the command line's STDOUT.

1.5 Runners

A "runner" manages the interaction between the Environment and the Agent. TensorForce comes with ready-to-use runners. Of course, you can implement your own runners, too. If you are not using simulation environments, the runner is simply your application code using the Agent API.

Environment <-> Runner <-> Agent <-> Model

1.5.1 Ready-to-use runners

We implemented a standard runner, a threaded runner (for real-time interaction e.g. with OpenAI Universe) and a distributed runner for A3C variants.

Runner

This is the standard runner. It requires an agent and an environment for initialization:

```
from tensorforce.execution import Runner

runner = Runner(
    agent = agent, # Agent object
    environment = env # Environment object
)
```

A reinforcement learning agent observes states from the environment, selects actions and collect experience which is used to update its model and improve action selection. You can get information about our ready-to-use agents [here](#).

The environment object is either the "real" environment, or a proxy which fulfills the actions selected by the agent in the real world. You can find information about environments [here](#).

The runner is started with the `Runner.run(...)` method:

```
runner.run(
    episodes = int, # number of episodes to run
    max_timesteps = int, # maximum timesteps per episode
    episode_finished = object, # callback function called when episode is finished
)
runner.close()
```

You can use the `episode_finished` callback for printing performance feedback:

```
def episode_finished(r):
    if r.episode % 10 == 0:
        print("Finished episode {ep} after {ts} timesteps".format(ep=r.episode + 1,
        ↪ts=r.timestep + 1))
        print("Episode reward: {}".format(r.episode_rewards[-1]))
        print("Average of last 10 rewards: {}".format(np.mean(r.episode_rewards[-
        ↪10:])))
    return True
```

Using the Runner

Here is some example code for using the runner (without preprocessing).

```
import logging

from tensorforce.contrib.openai_gym import OpenAIGym
from tensorforce.agents import DQNAgent
from tensorforce.execution import Runner

def main():
    gym_id = 'CartPole-v0'
    max_episodes = 10000
    max_timesteps = 1000

    env = OpenAIGym(gym_id)
    network_spec = [
        dict(type='dense', size=32, activation='tanh'),
        dict(type='dense', size=32, activation='tanh')
    ]

    agent = DQNAgent(
        states_spec=env.states,
        actions_spec=env.actions,
        network_spec=network_spec,
        batch_size=64
    )

    runner = Runner(agent, env)

    report_episodes = 10
```

```

def episode_finished(r):
    if r.episode % report_episodes == 0:
        logging.info("Finished episode {ep} after {ts} timesteps".format(ep=r.
↪episode, ts=r.timestep))
        logging.info("Episode reward: {}".format(r.episode_rewards[-1]))
        logging.info("Average of last 100 rewards: {}".format(sum(r.episode_
↪rewards[-100:]) / 100))
        return True

    print("Starting {agent} for Environment '{env}'".format(agent=agent, env=env))

    runner.run(max_episodes, max_timesteps, episode_finished=episode_finished)
    runner.close()

    print("Learning finished. Total episodes: {ep}".format(ep=runner.episode))

if __name__ == '__main__':
    main()

```

1.5.2 Building your own runner

There are three mandatory tasks any runner implements: Obtaining an action from the agent, passing it to the environment, and passing the resulting observation to the agent.

```

# Get action
action = agent.act(state)

# Execute action in the environment
state, reward, terminal_state = environment.execute(action)

# Pass observation to the agent
agent.observe(state, action, reward, terminal_state)

```

The key idea here is the separation of concerns. External code should not need to manage batches or remember network features, this is that the agent is for. Conversely, an agent need not concern itself with how a model is implemented and the API should facilitate easy combination of different agents and models.

If you would like to build your own runner, it is probably a good idea to take a look at the [source code of our Runner class](#).

1.6 tensorforce package

1.6.1 Subpackages

tensorforce.agents package

Submodules

tensorforce.agents.agent module

class tensorforce.agents.agent.**Agent** (*states, actions, batched_observe=True, batching_capacity=1000*)

Bases: object

Base class for TensorForce agents.

__init__ (*states, actions, batched_observe=True, batching_capacity=1000*)
Initializes the agent.

Parameters **states** – States specification, with the following attributes (required):

Parameters **actions** – Actions specification, with the following attributes (required):

Parameters

- **batched_observe** (*bool*) – Specifies whether calls to `model.observe()` are batched, for improved performance (default: `true`).
- **batching_capacity** (*int*) – Batching capacity of agent and model (default: 1000).

act (*states, deterministic=False, independent=False, fetch_tensors=None*)

Return action(s) for given state(s). States preprocessing and exploration are applied if configured accordingly.

Parameters

- **states** (*any*) – One state (usually a value tuple) or dict of states if multiple states are expected.
- **deterministic** (*bool*) – If true, no exploration and sampling is applied.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).
- **fetch_tensors** (*list*) – Optional String of named tensors to fetch

Returns Scalar value of the action or dict of multiple actions the agent wants to execute. (fetched_tensors) Optional dict() with named tensors fetched

close ()

static from_spec (*spec, kwargs*)
Creates an agent from a specification dict.

initialize_model ()
Creates the model for the respective agent based on specifications given by user. This is a separate call after constructing the agent because the agent constructor has to perform a number of checks on the specs first, sometimes adjusting them e.g. by converting to a dict.

last_observation ()

observe (*terminal*, *reward*)

Observe experience from the environment to learn from. Optionally pre-processes rewards Child classes should call super to get the processed reward EX: `terminal, reward = super(). . .`

Parameters

- **terminal** (*bool*) – boolean indicating if the episode terminated after the observation.
- **reward** (*float*) – scalar reward that resulted from executing the action.

reset ()

Reset the agent to its initial state (e.g. on experiment start). Updates the Model's internal episode and time step counter, internal states, and resets preprocessors.

restore_model (*directory=None*, *file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

save_model (*directory=None*, *append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** (*str*) – Optional checkpoint directory.
- **append_timestep** (*bool*) – Appends the current timestep to the checkpoint file if true. If this is set to True, the load path must include the checkpoint timestep suffix. For example, if stored to `models/` and set to true, the exported file will be of the form `models/model.ckpt-X` where X is the last timestep saved. The load path must precisely match this file name. If this option is turned off, the checkpoint will always overwrite the file specified in path and the model can always be loaded under this path.

Returns Checkpoint path where the model was saved.

set_normalized_actions (*actions*)

set_normalized_states (*states*)

should_stop ()

tensorforce.agents.batch_agent module

tensorforce.agents.constant_agent module

```
class tensorforce.agents.constant_agent.ConstantAgent (states, actions, action_values,
                                                         batched_observe=True,
                                                         batching_capacity=1000,
                                                         scope='constant', device=None,
                                                         saver=None,
                                                         summarizer=None, distributed=None)
```

Bases: `tensorforce.agents.agent.Agent`

Agent returning constant action values.

__init__ (*states, actions, action_values, batched_observe=True, batching_capacity=1000, scope='constant', device=None, saver=None, summarizer=None, distributed=None*)
Initializes the constant agent.

Parameters

- **action_values** (*value, or dict of values*) – Action values returned by the agent (required).
- **scope** (*str*) – TensorFlow scope (default: name of agent).
- **device** – TensorFlow device (default: none)
- **saver** – Saver specification, with the following attributes (default: none):

Parameters summarizer – Summarizer specification, with the following attributes (default: none):

Parameters distributed – Distributed specification, with the following attributes (default: none):

act (*states, deterministic=False, independent=False, fetch_tensors=None*)

Return action(s) for given state(s). States preprocessing and exploration are applied if configured accordingly.

Parameters

- **states** (*any*) – One state (usually a value tuple) or dict of states if multiple states are expected.
- **deterministic** (*bool*) – If true, no exploration and sampling is applied.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).
- **fetch_tensors** (*list*) – Optional String of named tensors to fetch

Returns Scalar value of the action or dict of multiple actions the agent wants to execute. (fetched_tensors) Optional dict() with named tensors fetched

close ()

from_spec (*spec, kwargs*)

Creates an agent from a specification dict.

initialize_model ()

last_observation ()

observe (*terminal, reward*)

Observe experience from the environment to learn from. Optionally pre-processes rewards Child classes should call super to get the processed reward EX: terminal, reward = super(). . .

Parameters

- **terminal** (*bool*) – boolean indicating if the episode terminated after the observation.
- **reward** (*float*) – scalar reward that resulted from executing the action.

reset ()

Reset the agent to its initial state (e.g. on experiment start). Updates the Model's internal episode and time step counter, internal states, and resets preprocessors.

restore_model (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

save_model (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** (*str*) – Optional checkpoint directory.
- **append_timestep** (*bool*) – Appends the current timestep to the checkpoint file if true. If this is set to True, the load path must include the checkpoint timestep suffix. For example, if stored to models/ and set to true, the exported file will be of the form models/model.ckpt-X where X is the last timestep saved. The load path must precisely match this file name. If this option is turned off, the checkpoint will always overwrite the file specified in path and the model can always be loaded under this path.

Returns Checkpoint path where the model was saved.

set_normalized_actions (*actions*)

set_normalized_states (*states*)

should_stop ()

tensorforce.agents.ddqn_agent module**tensorforce.agents.dqfd_agent module**

```
class tensorforce.agents.dqfd_agent.DQFDAgent (states,          actions,          network,
                                              batched_observe=True,      batch-
                                              ing_capacity=1000,          scope='dqfd',
                                              device=None,              saver=None,
                                              summarizer=None,           execu-
                                              tion=None,              variable_noise=None,
                                              states_preprocessing=None,      ac-
                                              tions_exploration=None,          re-
                                              ward_preprocessing=None,
                                              update_mode=None,              mem-
                                              ory=None,          optimizer=None,      dis-
                                              count=0.99,          distributions=None,
                                              entropy_regularization=None,
                                              target_sync_frequency=10000,
                                              target_update_weight=1.0,          hu-
                                              ber_loss=None,          expert_margin=0.5,
                                              supervised_weight=0.1,
                                              demo_memory_capacity=10000,
                                              demo_sampling_ratio=0.2)
```

Bases: `tensorforce.agents.learning_agent.LearningAgent`

Deep Q-learning from demonstration agent (Hester et al., 2017).

```
__init__ (states, actions, network, batched_observe=True, batching_capacity=1000, scope='dqfd',
          device=None, saver=None, summarizer=None, execution=None, variable_noise=None,
          states_preprocessing=None, actions_exploration=None, reward_preprocessing=None,
          update_mode=None, memory=None, optimizer=None, discount=0.99, distribu-
          tions=None, entropy_regularization=None, target_sync_frequency=10000, tar-
          get_update_weight=1.0, huber_loss=None, expert_margin=0.5, supervised_weight=0.1,
          demo_memory_capacity=10000, demo_sampling_ratio=0.2)
```

Initializes the DQFD agent.

Parameters `update_mode` – Update mode specification, with the following attributes:

Parameters

- **memory** (*spec*) – Memory specification, see `core.memories` module for more information (default: `{type='replay', include_next_states=true, capacity=1000*batch_size}`).
- **optimizer** (*spec*) – Optimizer specification, see `core.optimizers` module for more information (default: `{type='adam', learning_rate=1e-3}`).
- **target_sync_frequency** (*int*) – Target network sync frequency (default: 10000).
- **target_update_weight** (*float*) – Target network update weight (default: 1.0).
- **huber_loss** (*float*) – Huber loss clipping (default: none).
- **expert_margin** (*float*) – Enforced supervised margin between expert action Q-value and other Q-values (default: 0.5).
- **supervised_weight** (*float*) – Weight of supervised loss term (default: 0.1).
- **demo_memory_capacity** (*int*) – Capacity of expert demonstration memory (default: 10000).

- **demo_sampling_ratio** (*float*) – Runtime sampling ratio of expert data (default: 0.2).

act (*states, deterministic=False, independent=False, fetch_tensors=None*)

Return action(s) for given state(s). States preprocessing and exploration are applied if configured accordingly.

Parameters

- **states** (*any*) – One state (usually a value tuple) or dict of states if multiple states are expected.
- **deterministic** (*bool*) – If true, no exploration and sampling is applied.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).
- **fetch_tensors** (*list*) – Optional String of named tensors to fetch

Returns Scalar value of the action or dict of multiple actions the agent wants to execute. (fetched_tensors) Optional dict() with named tensors fetched

close ()

from_spec (*spec, kwargs*)

Creates an agent from a specification dict.

import_demonstrations (*demonstrations*)

Imports demonstrations, i.e. expert observations. Note that for large numbers of observations, `set_demonstrations` is more appropriate, which directly sets memory contents to an array an expects a different layout.

Parameters demonstrations – List of observation dicts

import_experience (*experiences*)

Imports experiences.

Parameters experiences –

initialize_model ()

last_observation ()

observe (*terminal, reward*)

Observe experience from the environment to learn from. Optionally pre-processes rewards Child classes should call `super` to get the processed reward EX: `terminal, reward = super(). . .`

Parameters

- **terminal** (*bool*) – boolean indicating if the episode terminated after the observation.
- **reward** (*float*) – scalar reward that resulted from executing the action.

pretrain (*steps*)

Computes pre-train updates.

Parameters steps – Number of updates to execute.

reset ()

Reset the agent to its initial state (e.g. on experiment start). Updates the Model's internal episode and time step counter, internal states, and resets preprocessors.

restore_model (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

save_model (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** (*str*) – Optional checkpoint directory.
- **append_timestep** (*bool*) – Appends the current timestep to the checkpoint file if true. If this is set to True, the load path must include the checkpoint timestep suffix. For example, if stored to models/ and set to true, the exported file will be of the form models/model.ckpt-X where X is the last timestep saved. The load path must precisely match this file name. If this option is turned off, the checkpoint will always overwrite the file specified in path and the model can always be loaded under this path.

Returns Checkpoint path where the model was saved.

set_normalized_actions (*actions*)

set_normalized_states (*states*)

should_stop ()

tensorforce.agents.dqn_agent module

```
class tensorforce.agents.dqn_agent.DQNAgent (states, actions, network,  
                                             batched_observe=True, batching_capacity=1000, scope='dqn', de-  
                                             vice=None, saver=None, summarizer=None,  
                                             execution=None, variable_noise=None,  
                                             states_preprocessing=None, ac-  
                                             tions_exploration=None, re-  
                                             ward_preprocessing=None, up-  
                                             date_mode=None, memory=None, op-  
                                             timizer=None, discount=0.99, distribu-  
                                             tions=None, entropy_regularization=None,  
                                             target_sync_frequency=10000, tar-  
                                             get_update_weight=1.0, dou-  
                                             ble_q_model=False, huber_loss=None)
```

Bases: `tensorforce.agents.learning_agent.LearningAgent`

Deep Q-Network agent (Mnih et al., 2015).

```
__init__ (states, actions, network, batched_observe=True, batching_capacity=1000, scope='dqn',  
          device=None, saver=None, summarizer=None, execution=None, variable_noise=None,  
          states_preprocessing=None, actions_exploration=None, reward_preprocessing=None, up-  
          date_mode=None, memory=None, optimizer=None, discount=0.99, distributions=None,  
          entropy_regularization=None, target_sync_frequency=10000, target_update_weight=1.0,  
          double_q_model=False, huber_loss=None)
```

Initializes the DQN agent.

Parameters **update_mode** – Update mode specification, with the following attributes:

Parameters

- **memory** (*spec*) – Memory specification, see `core.memories` module for more information (default: `{type='replay', include_next_states=True, capacity=1000*batch_size}`).
- **optimizer** (*spec*) – Optimizer specification, see `core.optimizers` module for more information (default: `{type='adam', learning_rate=1e-3}`).
- **target_sync_frequency** (*int*) – Target network sync frequency (default: 10000).
- **target_update_weight** (*float*) – Target network update weight (default: 1.0).
- **double_q_model** (*bool*) – Specifies whether double DQN mode is used (default: `false`).
- **huber_loss** (*float*) – Huber loss clipping (default: `none`).

act (*states*, *deterministic=False*, *independent=False*, *fetch_tensors=None*)

Return action(s) for given state(s). States preprocessing and exploration are applied if configured accordingly.

Parameters

- **states** (*any*) – One state (usually a value tuple) or dict of states if multiple states are expected.
- **deterministic** (*bool*) – If true, no exploration and sampling is applied.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).
- **fetch_tensors** (*list*) – Optional String of named tensors to fetch

Returns Scalar value of the action or dict of multiple actions the agent wants to execute.
(*fetch_tensors*) Optional dict() with named tensors fetched

close ()

from_spec (*spec*, *kwargs*)

Creates an agent from a specification dict.

import_experience (*experiences*)

Imports experiences.

Parameters experiences –

initialize_model ()

last_observation ()

observe (*terminal*, *reward*)

Observe experience from the environment to learn from. Optionally pre-processes rewards Child classes should call `super` to get the processed reward EX: `terminal, reward = super()....`

Parameters

- **terminal** (*bool*) – boolean indicating if the episode terminated after the observation.
- **reward** (*float*) – scalar reward that resulted from executing the action.

reset ()

Reset the agent to its initial state (e.g. on experiment start). Updates the Model's internal episode and time step counter, internal states, and resets preprocessors.

restore_model (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

save_model (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** (*str*) – Optional checkpoint directory.
- **append_timestep** (*bool*) – Appends the current timestep to the checkpoint file if true. If this is set to True, the load path must include the checkpoint timestep suffix. For example, if stored to models/ and set to true, the exported file will be of the form models/model.ckpt-X where X is the last timestep saved. The load path must precisely match this file name. If this option is turned off, the checkpoint will always overwrite the file specified in path and the model can always be loaded under this path.

Returns Checkpoint path where the model was saved.

set_normalized_actions (*actions*)

set_normalized_states (*states*)

should_stop ()

tensorforce.agents.dqn_nstep_agent module

```
class tensorforce.agents.dqn_nstep_agent.DQNNStepAgent (states, actions, network,  
                                                    batched_observe=True,  
                                                    batching_capacity=1000,  
                                                    scope='dqn-nstep', device=None,  
                                                    saver=None,  
                                                    summarizer=None, execution=None,  
                                                    variable_noise=None,  
                                                    states_preprocessing=None,  
                                                    actions_exploration=None,  
                                                    reward_preprocessing=None,  
                                                    update_mode=None, memory=None,  
                                                    optimizer=None,  
                                                    discount=0.99, distributions=None,  
                                                    entropy_regularization=None,  
                                                    target_update_weight=1.0,  
                                                    double_q_model=False,  
                                                    huber_loss=None)
```

Bases: *tensorforce.agents.learning_agent.LearningAgent*

DQN n-step agent.

```
__init__(states, actions, network, batched_observe=True, batching_capacity=1000,
          scope='dqn-nstep', device=None, saver=None, summarizer=None, execution=None,
          variable_noise=None, states_preprocessing=None, actions_exploration=None,
          reward_preprocessing=None, update_mode=None, memory=None, optimizer=None,
          discount=0.99, distributions=None, entropy_regularization=None,
          target_sync_frequency=10000, target_update_weight=1.0, double_q_model=False,
          huber_loss=None)
```

Initializes the DQN n-step agent.

Parameters `update_mode` – Update mode specification, with the following attributes:

Parameters

- **memory** (*spec*) – Memory specification, see core.memories module for more information (default: {type='latest', include_next_states=true, capacity=1000*batch_size}).
- **optimizer** (*spec*) – Optimizer specification, see core.optimizers module for more information (default: {type='adam', learning_rate=1e-3}).
- **target_sync_frequency** (*int*) – Target network sync frequency (default: 10000).
- **target_update_weight** (*float*) – Target network update weight (default: 1.0).
- **double_q_model** (*bool*) – Specifies whether double DQN mode is used (default: false).
- **huber_loss** (*float*) – Huber loss clipping (default: none).

```
act(states, deterministic=False, independent=False, fetch_tensors=None)
```

Return action(s) for given state(s). States preprocessing and exploration are applied if configured accordingly.

Parameters

- **states** (*any*) – One state (usually a value tuple) or dict of states if multiple states are expected.
- **deterministic** (*bool*) – If true, no exploration and sampling is applied.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).
- **fetch_tensors** (*list*) – Optional String of named tensors to fetch

Returns Scalar value of the action or dict of multiple actions the agent wants to execute. (fetched_tensors) Optional dict() with named tensors fetched

```
close()
```

```
from_spec(spec, kwargs)
```

Creates an agent from a specification dict.

```
import_experience(experiences)
```

Imports experiences.

Parameters `experiences` –

```
initialize_model()
```

```
last_observation()
```

observe (*terminal*, *reward*)

Observe experience from the environment to learn from. Optionally pre-processes rewards Child classes should call super to get the processed reward EX: `terminal, reward = super(). .`

Parameters

- **terminal** (*bool*) – boolean indicating if the episode terminated after the observation.
- **reward** (*float*) – scalar reward that resulted from executing the action.

reset ()

Reset the agent to its initial state (e.g. on experiment start). Updates the Model's internal episode and time step counter, internal states, and resets preprocessors.

restore_model (*directory=None*, *file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

save_model (*directory=None*, *append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** (*str*) – Optional checkpoint directory.
- **append_timestep** (*bool*) – Appends the current timestep to the checkpoint file if true. If this is set to True, the load path must include the checkpoint timestep suffix. For example, if stored to `models/` and set to true, the exported file will be of the form `models/model.ckpt-X` where X is the last timestep saved. The load path must precisely match this file name. If this option is turned off, the checkpoint will always overwrite the file specified in path and the model can always be loaded under this path.

Returns Checkpoint path where the model was saved.

set_normalized_actions (*actions*)

set_normalized_states (*states*)

should_stop ()

tensorforce.agents.learning_agent module

```
class tensorforce.agents.learning_agent.LearningAgent (states,      actions,      net-
                                                         work,          update_mode,
                                                         memory,        optimizer,
                                                         batched_observe=True,
                                                         batching_capacity=1000,
                                                         scope='learning-agent',
                                                         device=None,   saver=None,
                                                         summarizer=None, execution=None,
                                                         variable_noise=None,
                                                         states_preprocessing=None,
                                                         actions_exploration=None,
                                                         reward_preprocessing=None,
                                                         discount=0.99, distributions=None,
                                                         entropy_regularization=None)
```

Bases: *tensorforce.agents.agent.Agent*

Base class for learning agents, using as model a subclass of MemoryModel and DistributionModel.

```
__init__ (states, actions, network, update_mode, memory, optimizer, batched_observe=True,
          batching_capacity=1000, scope='learning-agent', device=None, saver=None, summarizer=None,
          execution=None, variable_noise=None, states_preprocessing=None, actions_exploration=None,
          reward_preprocessing=None, discount=0.99, distributions=None, entropy_regularization=None)
```

Initializes the learning agent.

Parameters **update_mode** – Update mode specification, with the following attributes (required):

Parameters

- **memory** (*spec*) – Memory specification, see core.memories module for more information (required).
- **optimizer** (*spec*) – Optimizer specification, see core.optimizers module for more information (required).
- **network** (*spec*) – Network specification, usually a list of layer specifications, see core.networks module for more information (required).
- **scope** (*str*) – TensorFlow scope (default: name of agent).
- **device** – TensorFlow device (default: none)
- **saver** – Saver specification, with the following attributes (default: none):

Parameters **summarizer** – Summarizer specification, with the following attributes (default: none):

Parameters **execution** – Distributed specification, with the following attributes (default: none):

Parameters

- **variable_noise** (*float*) – Standard deviation of variable noise (default: none).

- **states_preprocessing** (*spec, or dict of specs*) – States preprocessing specification, see `core.preprocessors` module for more information (default: none)
- **actions_exploration** (*spec, or dict of specs*) – Actions exploration specification, see `core.explorations` module for more information (default: none).
- **reward_preprocessing** (*spec*) – Reward preprocessing specification, see `core.preprocessors` module for more information (default: none).
- **discount** (*float*) – Discount factor for future rewards (default: 0.99).
- **distributions** (*spec / dict of specs*) – Distributions specifications, see `core.distributions` module for more information (default: none).
- **entropy_regularization** (*float*) – Entropy regularization weight (default: none).

act (*states, deterministic=False, independent=False, fetch_tensors=None*)

Return action(s) for given state(s). States preprocessing and exploration are applied if configured accordingly.

Parameters

- **states** (*any*) – One state (usually a value tuple) or dict of states if multiple states are expected.
- **deterministic** (*bool*) – If true, no exploration and sampling is applied.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).
- **fetch_tensors** (*list*) – Optional String of named tensors to fetch

Returns Scalar value of the action or dict of multiple actions the agent wants to execute.
(*fetch_tensors*) Optional dict() with named tensors fetched

close ()

from_spec (*spec, kwargs*)

Creates an agent from a specification dict.

import_experience (*experiences*)

Imports experiences.

Parameters experiences –

initialize_model ()

Creates the model for the respective agent based on specifications given by user. This is a separate call after constructing the agent because the agent constructor has to perform a number of checks on the specs first, sometimes adjusting them e.g. by converting to a dict.

last_observation ()

observe (*terminal, reward*)

Observe experience from the environment to learn from. Optionally pre-processes rewards Child classes should call super to get the processed reward EX: `terminal, reward = super()....`

Parameters

- **terminal** (*bool*) – boolean indicating if the episode terminated after the observation.
- **reward** (*float*) – scalar reward that resulted from executing the action.

reset ()

Reset the agent to its initial state (e.g. on experiment start). Updates the Model's internal episode and time step counter, internal states, and resets preprocessors.

restore_model (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

save_model (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** (*str*) – Optional checkpoint directory.
- **append_timestep** (*bool*) – Appends the current timestep to the checkpoint file if true. If this is set to True, the load path must include the checkpoint timestep suffix. For example, if stored to models/ and set to true, the exported file will be of the form models/model.ckpt-X where X is the last timestep saved. The load path must precisely match this file name. If this option is turned off, the checkpoint will always overwrite the file specified in path and the model can always be loaded under this path.

Returns Checkpoint path where the model was saved.

set_normalized_actions (*actions*)

set_normalized_states (*states*)

should_stop ()

tensorforce.agents.memory_agent module

tensorforce.agents.naf_agent module

```
class tensorforce.agents.naf_agent.NAFAgent (states,           actions,           network,
                                             batched_observe=True,           batch-
                                             ing_capacity=1000,           scope='naf', de-
                                             vice=None, saver=None, summarizer=None,
                                             execution=None,           variable_noise=None,
                                             states_preprocessing=None,           ac-
                                             tions_exploration=None,           re-
                                             ward_preprocessing=None,           up-
                                             date_mode=None, memory=None, op-
                                             timizer=None, discount=0.99, distribu-
                                             tions=None, entropy_regularization=None,
                                             target_sync_frequency=10000,           tar-
                                             get_update_weight=1.0,           dou-
                                             ble_q_model=False, huber_loss=None)
```

Bases: `tensorforce.agents.learning_agent.LearningAgent`

Normalized Advantage Function agent (Gu et al., 2016).

```
__init__(states, actions, network, batched_observe=True, batching_capacity=1000, scope='naf',
          device=None, saver=None, summarizer=None, execution=None, variable_noise=None,
          states_preprocessing=None, actions_exploration=None, reward_preprocessing=None, up-
          date_mode=None, memory=None, optimizer=None, discount=0.99, distributions=None,
          entropy_regularization=None, target_sync_frequency=10000, target_update_weight=1.0,
          double_q_model=False, huber_loss=None)
```

Initializes the NAF agent.

Parameters **update_mode** – Update mode specification, with the following attributes:

Parameters

- **memory** (*spec*) – Memory specification, see core.memories module for more information (default: {type='replay', include_next_states=true, capacity=1000*batch_size}).
- **optimizer** (*spec*) – Optimizer specification, see core.optimizers module for more information (default: {type='adam', learning_rate=1e-3}).
- **target_sync_frequency** (*int*) – Target network sync frequency (default: 10000).
- **target_update_weight** (*float*) – Target network update weight (default: 1.0).
- **double_q_model** (*bool*) – Specifies whether double DQN mode is used (default: false).
- **huber_loss** (*float*) – Huber loss clipping (default: none).

```
act(states, deterministic=False, independent=False, fetch_tensors=None)
```

Return action(s) for given state(s). States preprocessing and exploration are applied if configured accordingly.

Parameters

- **states** (*any*) – One state (usually a value tuple) or dict of states if multiple states are expected.
- **deterministic** (*bool*) – If true, no exploration and sampling is applied.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).
- **fetch_tensors** (*list*) – Optional String of named tensors to fetch

Returns Scalar value of the action or dict of multiple actions the agent wants to execute. (fetched_tensors) Optional dict() with named tensors fetched

```
close()
```

```
from_spec(spec, kwargs)
```

Creates an agent from a specification dict.

```
import_experience(experiences)
```

Imports experiences.

Parameters experiences –

```
initialize_model()
```

```
last_observation()
```

```
observe(terminal, reward)
```

Observe experience from the environment to learn from. Optionally pre-processes rewards Child classes should call super to get the processed reward EX: terminal, reward = super()....

Parameters

- **terminal** (*bool*) – boolean indicating if the episode terminated after the observation.
- **reward** (*float*) – scalar reward that resulted from executing the action.

reset ()

Reset the agent to its initial state (e.g. on experiment start). Updates the Model's internal episode and time step counter, internal states, and resets preprocessors.

restore_model (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

save_model (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** (*str*) – Optional checkpoint directory.
- **append_timestep** (*bool*) – Appends the current timestep to the checkpoint file if true. If this is set to True, the load path must include the checkpoint timestep suffix. For example, if stored to models/ and set to true, the exported file will be of the form models/model.ckpt-X where X is the last timestep saved. The load path must precisely match this file name. If this option is turned off, the checkpoint will always overwrite the file specified in path and the model can always be loaded under this path.

Returns Checkpoint path where the model was saved.

set_normalized_actions (*actions*)

set_normalized_states (*states*)

should_stop ()

tensorforce.agents.ppo_agent module

```
class tensorforce.agents.ppo_agent.PPOAgent (states,          actions,          network,
                                             batched_observe=True,      batching_capacity=1000,
                                             scope='ppo',      device=None, saver=None,
                                             summarizer=None, execution=None,
                                             variable_noise=None, states_preprocessing=None,
                                             actions_exploration=None, reward_preprocessing=None,
                                             update_mode=None, memory=None, discount=0.99,
                                             distributions=None, entropy_regularization=None,
                                             baseline_mode=None, baseline=None,
                                             baseline_optimizer=None, gae_lambda=None,
                                             likelihood_ratio_clipping=0.2, step_optimizer=None,
                                             subsampling_fraction=0.1, optimization_steps=50)
```

Bases: `tensorforce.agents.learning_agent.LearningAgent`

Proximal Policy Optimization agent (Schulman et al., 2017).

```
__init__ (states, actions, network, batched_observe=True, batching_capacity=1000,
           scope='ppo', device=None, saver=None, summarizer=None, execution=None,
           variable_noise=None, states_preprocessing=None, actions_exploration=None,
           reward_preprocessing=None, update_mode=None, memory=None, discount=0.99,
           distributions=None, entropy_regularization=None, baseline_mode=None,
           baseline=None, baseline_optimizer=None, gae_lambda=None, likelihood_ratio_clipping=0.2,
           step_optimizer=None, subsampling_fraction=0.1, optimization_steps=50)
```

Initializes the PPO agent.

Parameters `update_mode` – Update mode specification, with the following attributes:

Parameters

- **memory** (*spec*) – Memory specification, see `core.memories` module for more information (default: `{type='latest', include_next_states=False, capacity=1000*batch_size}`).
- **optimizer** (*spec*) – PPO agent implicitly defines a multi-step subsampling optimizer.
- **baseline_mode** (*str*) – One of 'states', 'network' (default: none).
- **baseline** (*spec*) – Baseline specification, see `core.baselines` module for more information (default: none).
- **baseline_optimizer** (*spec*) – Baseline optimizer specification, see `core.optimizers` module for more information (default: none).
- **gae_lambda** (*float*) – Lambda factor for generalized advantage estimation (default: none).
- **likelihood_ratio_clipping** (*float*) – Likelihood ratio clipping for policy gradient (default: 0.2).
- **step_optimizer** (*spec*) – Step optimizer specification of implicit multi-step subsampling optimizer, see `core.optimizers` module for more information (default: `{type='adam', learning_rate=1e-3}`).
- **subsampling_fraction** (*float*) – Subsampling fraction of implicit subsampling optimizer (default: 0.1).

- **optimization_steps** (*int*) – Number of optimization steps for implicit multi-step optimizer (default: 50).

act (*states*, *deterministic=False*, *independent=False*, *fetch_tensors=None*)

Return action(s) for given state(s). States preprocessing and exploration are applied if configured accordingly.

Parameters

- **states** (*any*) – One state (usually a value tuple) or dict of states if multiple states are expected.
- **deterministic** (*bool*) – If true, no exploration and sampling is applied.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).
- **fetch_tensors** (*list*) – Optional String of named tensors to fetch

Returns Scalar value of the action or dict of multiple actions the agent wants to execute. (fetched_tensors) Optional dict() with named tensors fetched

close ()

from_spec (*spec*, *kwargs*)

Creates an agent from a specification dict.

import_experience (*experiences*)

Imports experiences.

Parameters experiences –

initialize_model ()

last_observation ()

observe (*terminal*, *reward*)

Observe experience from the environment to learn from. Optionally pre-processes rewards Child classes should call super to get the processed reward EX: `terminal, reward = super()....`

Parameters

- **terminal** (*bool*) – boolean indicating if the episode terminated after the observation.
- **reward** (*float*) – scalar reward that resulted from executing the action.

reset ()

Reset the agent to its initial state (e.g. on experiment start). Updates the Model's internal episode and time step counter, internal states, and resets preprocessors.

restore_model (*directory=None*, *file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

save_model (*directory=None*, *append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** (*str*) – Optional checkpoint directory.
- **append_timestep** (*bool*) – Appends the current timestep to the checkpoint file if true. If this is set to True, the load path must include the checkpoint timestep suffix. For example, if stored to models/ and set to true, the exported file will be of the form models/model.ckpt-X where X is the last timestep saved. The load path must precisely match this file name. If this option is turned off, the checkpoint will always overwrite the file specified in path and the model can always be loaded under this path.

Returns Checkpoint path where the model was saved.

set_normalized_actions (*actions*)

set_normalized_states (*states*)

should_stop ()

tensorflow.agents.random_agent module

```
class tensorflow.agents.random_agent.RandomAgent (states, actions,
                                                  batched_observe=True,
                                                  batching_capacity=1000,
                                                  scope='random', device=None,
                                                  saver=None, summarizer=None,
                                                  distributed=None)
```

Bases: *tensorflow.agents.agent.Agent*

Agent returning random action values.

```
__init__(states, actions, batched_observe=True, batching_capacity=1000, scope='random', device=None, saver=None, summarizer=None, distributed=None)
```

Initializes the random agent.

Parameters

- **scope** (*str*) – TensorFlow scope (default: name of agent).
- **device** – TensorFlow device (default: none)
- **saver** – Saver specification, with the following attributes (default: none):

Parameters summarizer – Summarizer specification, with the following attributes (default: none):

Parameters distributed – Distributed specification, with the following attributes (default: none):

```
act (states, deterministic=False, independent=False, fetch_tensors=None)
```

Return action(s) for given state(s). States preprocessing and exploration are applied if configured accordingly.

Parameters

- **states** (*any*) – One state (usually a value tuple) or dict of states if multiple states are expected.
- **deterministic** (*bool*) – If true, no exploration and sampling is applied.

- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).
- **fetch_tensors** (*list*) – Optional String of named tensors to fetch

Returns Scalar value of the action or dict of multiple actions the agent wants to execute.
(*fetch_tensors*) Optional dict() with named tensors fetched

close ()

from_spec (*spec, kwargs*)

Creates an agent from a specification dict.

initialize_model ()

last_observation ()

observe (*terminal, reward*)

Observe experience from the environment to learn from. Optionally pre-processes rewards Child classes should call super to get the processed reward EX: `terminal, reward = super()....`

Parameters

- **terminal** (*bool*) – boolean indicating if the episode terminated after the observation.
- **reward** (*float*) – scalar reward that resulted from executing the action.

reset ()

Reset the agent to its initial state (e.g. on experiment start). Updates the Model's internal episode and time step counter, internal states, and resets preprocessors.

restore_model (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

save_model (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** (*str*) – Optional checkpoint directory.
- **append_timestep** (*bool*) – Appends the current timestep to the checkpoint file if true. If this is set to True, the load path must include the checkpoint timestep suffix. For example, if stored to `models/` and set to true, the exported file will be of the form `models/model.ckpt-X` where X is the last timestep saved. The load path must precisely match this file name. If this option is turned off, the checkpoint will always overwrite the file specified in path and the model can always be loaded under this path.

Returns Checkpoint path where the model was saved.

set_normalized_actions (*actions*)

set_normalized_states (*states*)

should_stop ()

tensorforce.agents.trpo_agent module

```
class tensorforce.agents.trpo_agent.TRPOAgent (states,          actions,          network,
                                                batched_observe=True,          batch-
ing_capacity=1000,          scope='trpo',
                                                device=None,          saver=None,
                                                summarizer=None,          execu-
tion=None,          variable_noise=None,
states_preprocessing=None,          ac-
tions_exploration=None,          re-
ward_preprocessing=None,          up-
date_mode=None,          memory=None,
discount=0.99,          distributions=None,
entropy_regularization=None,
baseline_mode=None,          base-
line=None,          baseline_optimizer=None,
gae_lambda=None,          likeli-
hood_ratio_clipping=None,          learn-
ing_rate=0.001, cg_max_iterations=20,
cg_damping=0.001,
cg_unroll_loop=False,
ls_max_iterations=10,
ls_accept_ratio=0.9,
ls_unroll_loop=False)
```

Bases: `tensorforce.agents.learning_agent.LearningAgent`

Trust Region Policy Optimization agent (Schulman et al., 2015).

```
__init__ (states, actions, network, batched_observe=True, batching_capacity=1000,
scope='trpo', device=None, saver=None, summarizer=None, execution=None,
variable_noise=None, states_preprocessing=None, actions_exploration=None, re-
ward_preprocessing=None, update_mode=None, memory=None, discount=0.99, dis-
tributions=None, entropy_regularization=None, baseline_mode=None, baseline=None,
baseline_optimizer=None, gae_lambda=None, likelihood_ratio_clipping=None, learn-
ing_rate=0.001, cg_max_iterations=20, cg_damping=0.001, cg_unroll_loop=False,
ls_max_iterations=10, ls_accept_ratio=0.9, ls_unroll_loop=False)
```

Initializes the TRPO agent.

Parameters `update_mode` – Update mode specification, with the following attributes:

Parameters

- **memory** (*spec*) – Memory specification, see `core.memories` module for more information (default: `{type='latest', include_next_states=false, capacity=1000*batch_size}`).
- **optimizer** (*spec*) – TRPO agent implicitly defines a optimized-step natural-gradient optimizer.
- **baseline_mode** (*str*) – One of 'states', 'network' (default: none).
- **baseline** (*spec*) – Baseline specification, see `core.baselines` module for more information (default: none).
- **baseline_optimizer** (*spec*) – Baseline optimizer specification, see `core.optimizers` module for more information (default: none).
- **gae_lambda** (*float*) – Lambda factor for generalized advantage estimation (default: none).

- **likelihood_ratio_clipping** (*float*) – Likelihood ratio clipping for policy gradient (default: none).
- **learning_rate** (*float*) – Learning rate of natural-gradient optimizer (default: 1e-3).
- **cg_max_iterations** (*int*) – Conjugate-gradient max iterations (default: 20).
- **cg_damping** (*float*) – Conjugate-gradient damping (default: 1e-3).
- **cg_unroll_loop** (*bool*) – Conjugate-gradient unroll loop (default: false).
- **ls_max_iterations** (*int*) – Line-search max iterations (default: 10).
- **ls_accept_ratio** (*float*) – Line-search accept ratio (default: 0.9).
- **ls_unroll_loop** (*bool*) – Line-search unroll loop (default: false).

act (*states*, *deterministic=False*, *independent=False*, *fetch_tensors=None*)

Return action(s) for given state(s). States preprocessing and exploration are applied if configured accordingly.

Parameters

- **states** (*any*) – One state (usually a value tuple) or dict of states if multiple states are expected.
- **deterministic** (*bool*) – If true, no exploration and sampling is applied.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).
- **fetch_tensors** (*list*) – Optional String of named tensors to fetch

Returns Scalar value of the action or dict of multiple actions the agent wants to execute. (fetched_tensors) Optional dict() with named tensors fetched

close ()

from_spec (*spec*, *kwargs*)

Creates an agent from a specification dict.

import_experience (*experiences*)

Imports experiences.

Parameters experiences –

initialize_model ()

last_observation ()

observe (*terminal*, *reward*)

Observe experience from the environment to learn from. Optionally pre-processes rewards Child classes should call super to get the processed reward EX: `terminal, reward = super()....`

Parameters

- **terminal** (*bool*) – boolean indicating if the episode terminated after the observation.
- **reward** (*float*) – scalar reward that resulted from executing the action.

reset ()

Reset the agent to its initial state (e.g. on experiment start). Updates the Model's internal episode and time step counter, internal states, and resets preprocessors.

restore_model (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

save_model (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** (*str*) – Optional checkpoint directory.
- **append_timestep** (*bool*) – Appends the current timestep to the checkpoint file if true. If this is set to True, the load path must include the checkpoint timestep suffix. For example, if stored to models/ and set to true, the exported file will be of the form models/model.ckpt-X where X is the last timestep saved. The load path must precisely match this file name. If this option is turned off, the checkpoint will always overwrite the file specified in path and the model can always be loaded under this path.

Returns Checkpoint path where the model was saved.

set_normalized_actions (*actions*)

set_normalized_states (*states*)

should_stop ()

tensorforce.agents.vpg_agent module

```
class tensorforce.agents.vpg_agent.VPGAgent (states,           actions,           network,  
                                             batched_observe=True,           batching_capacity=1000, scope='vpg', device=None, saver=None, summarizer=None,  
                                             execution=None,           variable_noise=None,  
                                             states_preprocessing=None,           actions_exploration=None,           reward_preprocessing=None, update_mode=None, memory=None, optimizer=None, discount=0.99, distributions=None, entropy_regularization=None, baseline_mode=None, baseline=None, baseline_optimizer=None, gae_lambda=None)
```

Bases: `tensorforce.agents.learning_agent.LearningAgent`

Vanilla policy gradient agent (Williams, 1992)).

```
__init__ (states, actions, network, batched_observe=True, batching_capacity=1000, scope='vpg',  
          device=None, saver=None, summarizer=None, execution=None, variable_noise=None,  
          states_preprocessing=None, actions_exploration=None, reward_preprocessing=None,  
          update_mode=None, memory=None, optimizer=None, discount=0.99, distributions=None,  
          entropy_regularization=None, baseline_mode=None, baseline=None,  
          baseline_optimizer=None, gae_lambda=None)
```

Initializes the VPG agent.

Parameters `update_mode` – Update mode specification, with the following attributes:

Parameters

- **memory** (*spec*) – Memory specification, see `core.memories` module for more information (default: `{type='latest', include_next_states=False, capacity=1000*batch_size}`).
- **optimizer** (*spec*) – Optimizer specification, see `core.optimizers` module for more information (default: `{type='adam', learning_rate=1e-3}`).
- **baseline_mode** (*str*) – One of 'states', 'network' (default: none).
- **baseline** (*spec*) – Baseline specification, see `core.baselines` module for more information (default: none).
- **baseline_optimizer** (*spec*) – Baseline optimizer specification, see `core.optimizers` module for more information (default: none).
- **gae_lambda** (*float*) – Lambda factor for generalized advantage estimation (default: none).

act (*states*, *deterministic=False*, *independent=False*, *fetch_tensors=None*)

Return action(s) for given state(s). States preprocessing and exploration are applied if configured accordingly.

Parameters

- **states** (*any*) – One state (usually a value tuple) or dict of states if multiple states are expected.
- **deterministic** (*bool*) – If true, no exploration and sampling is applied.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).
- **fetch_tensors** (*list*) – Optional String of named tensors to fetch

Returns Scalar value of the action or dict of multiple actions the agent wants to execute.
(`fetch_tensors`) Optional dict() with named tensors fetched

close ()

from_spec (*spec*, *kwargs*)

Creates an agent from a specification dict.

import_experience (*experiences*)

Imports experiences.

Parameters experiences –

initialize_model ()

last_observation ()

observe (*terminal*, *reward*)

Observe experience from the environment to learn from. Optionally pre-processes rewards Child classes should call super to get the processed reward EX: `terminal, reward = super()....`

Parameters

- **terminal** (*bool*) – boolean indicating if the episode terminated after the observation.
- **reward** (*float*) – scalar reward that resulted from executing the action.

reset ()

Reset the agent to its initial state (e.g. on experiment start). Updates the Model's internal episode and time step counter, internal states, and resets preprocessors.

restore_model (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

save_model (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** (*str*) – Optional checkpoint directory.
- **append_timestep** (*bool*) – Appends the current timestep to the checkpoint file if true. If this is set to True, the load path must include the checkpoint timestep suffix. For example, if stored to models/ and set to true, the exported file will be of the form models/model.ckpt-X where X is the last timestep saved. The load path must precisely match this file name. If this option is turned off, the checkpoint will always overwrite the file specified in path and the model can always be loaded under this path.

Returns Checkpoint path where the model was saved.

set_normalized_actions (*actions*)**set_normalized_states** (*states*)**should_stop ()**

Module contents

class tensorforce.agents.**Agent** (*states, actions, batched_observe=True, batching_capacity=1000*)

Bases: object

Base class for TensorForce agents.

__init__ (*states, actions, batched_observe=True, batching_capacity=1000*)

Initializes the agent.

Parameters **states** – States specification, with the following attributes (required):

Parameters **actions** – Actions specification, with the following attributes (required):

Parameters

- **batched_observe** (*bool*) – Specifies whether calls to model.observe() are batched, for improved performance (default: true).
- **batching_capacity** (*int*) – Batching capacity of agent and model (default: 1000).

act (*states*, *deterministic=False*, *independent=False*, *fetch_tensors=None*)

Return action(s) for given state(s). States preprocessing and exploration are applied if configured accordingly.

Parameters

- **states** (*any*) – One state (usually a value tuple) or dict of states if multiple states are expected.
- **deterministic** (*bool*) – If true, no exploration and sampling is applied.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).
- **fetch_tensors** (*list*) – Optional String of named tensors to fetch

Returns Scalar value of the action or dict of multiple actions the agent wants to execute.
(*fetch_tensors*) Optional dict() with named tensors fetched

close ()

static from_spec (*spec*, *kwargs*)

Creates an agent from a specification dict.

initialize_model ()

Creates the model for the respective agent based on specifications given by user. This is a separate call after constructing the agent because the agent constructor has to perform a number of checks on the specs first, sometimes adjusting them e.g. by converting to a dict.

last_observation ()

observe (*terminal*, *reward*)

Observe experience from the environment to learn from. Optionally pre-processes rewards Child classes should call super to get the processed reward EX: `terminal, reward = super(). . .`

Parameters

- **terminal** (*bool*) – boolean indicating if the episode terminated after the observation.
- **reward** (*float*) – scalar reward that resulted from executing the action.

reset ()

Reset the agent to its initial state (e.g. on experiment start). Updates the Model's internal episode and time step counter, internal states, and resets preprocessors.

restore_model (*directory=None*, *file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

save_model (*directory=None*, *append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** (*str*) – Optional checkpoint directory.

- **append_timestep** (*bool*) – Appends the current timestep to the checkpoint file if true. If this is set to True, the load path must include the checkpoint timestep suffix. For example, if stored to models/ and set to true, the exported file will be of the form models/model.ckpt-X where X is the last timestep saved. The load path must precisely match this file name. If this option is turned off, the checkpoint will always overwrite the file specified in path and the model can always be loaded under this path.

Returns Checkpoint path where the model was saved.

set_normalized_actions (*actions*)

set_normalized_states (*states*)

should_stop ()

```
class tensorflow.agents.ConstantAgent (states,          actions,          action_values,
                                       batched_observe=True, batching_capacity=1000,
                                       scope='constant', device=None, saver=None,
                                       summarizer=None, distributed=None)
```

Bases: `tensorflow.agents.agent.Agent`

Agent returning constant action values.

```
__init__(states, actions, action_values, batched_observe=True, batching_capacity=1000,
          scope='constant', device=None, saver=None, summarizer=None, distributed=None)
Initializes the constant agent.
```

Parameters

- **action_values** (*value, or dict of values*) – Action values returned by the agent (required).
- **scope** (*str*) – TensorFlow scope (default: name of agent).
- **device** – TensorFlow device (default: none)
- **saver** – Saver specification, with the following attributes (default: none):

Parameters summarizer – Summarizer specification, with the following attributes (default: none):

Parameters distributed – Distributed specification, with the following attributes (default: none):

```
act (states, deterministic=False, independent=False, fetch_tensors=None)
```

Return action(s) for given state(s). States preprocessing and exploration are applied if configured accordingly.

Parameters

- **states** (*any*) – One state (usually a value tuple) or dict of states if multiple states are expected.
- **deterministic** (*bool*) – If true, no exploration and sampling is applied.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).
- **fetch_tensors** (*list*) – Optional String of named tensors to fetch

Returns Scalar value of the action or dict of multiple actions the agent wants to execute.
(*fetcheds_tensors*) Optional dict() with named tensors fetched

close()

from_spec (*spec, kwargs*)

Creates an agent from a specification dict.

initialize_model()

last_observation()

observe (*terminal, reward*)

Observe experience from the environment to learn from. Optionally pre-processes rewards Child classes should call super to get the processed reward EX: `terminal, reward = super().`

Parameters

- **terminal** (*bool*) – boolean indicating if the episode terminated after the observation.
- **reward** (*float*) – scalar reward that resulted from executing the action.

reset()

Reset the agent to its initial state (e.g. on experiment start). Updates the Model's internal episode and time step counter, internal states, and resets preprocessors.

restore_model (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

save_model (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** (*str*) – Optional checkpoint directory.
- **append_timestep** (*bool*) – Appends the current timestep to the checkpoint file if true. If this is set to True, the load path must include the checkpoint timestep suffix. For example, if stored to `models/` and set to true, the exported file will be of the form `models/model.ckpt-X` where X is the last timestep saved. The load path must precisely match this file name. If this option is turned off, the checkpoint will always overwrite the file specified in path and the model can always be loaded under this path.

Returns Checkpoint path where the model was saved.

set_normalized_actions (*actions*)

set_normalized_states (*states*)

should_stop()

class `tensorforce.agents.RandomAgent` (*states, actions, batched_observe=True, batching_capacity=1000, scope='random', device=None, saver=None, summarizer=None, distributed=None*)

Bases: `tensorforce.agents.agent.Agent`

Agent returning random action values.

__init__ (*states, actions, batched_observe=True, batching_capacity=1000, scope='random', device=None, saver=None, summarizer=None, distributed=None*)
Initializes the random agent.

Parameters

- **scope** (*str*) – TensorFlow scope (default: name of agent).
- **device** – TensorFlow device (default: none)
- **saver** – Saver specification, with the following attributes (default: none):

Parameters summarizer – Summarizer specification, with the following attributes (default: none):

Parameters distributed – Distributed specification, with the following attributes (default: none):

act (*states, deterministic=False, independent=False, fetch_tensors=None*)
Return action(s) for given state(s). States preprocessing and exploration are applied if configured accordingly.

Parameters

- **states** (*any*) – One state (usually a value tuple) or dict of states if multiple states are expected.
- **deterministic** (*bool*) – If true, no exploration and sampling is applied.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).
- **fetch_tensors** (*list*) – Optional String of named tensors to fetch

Returns Scalar value of the action or dict of multiple actions the agent wants to execute.
(*fetch_tensors*) Optional dict() with named tensors fetched

close ()

from_spec (*spec, kwargs*)
Creates an agent from a specification dict.

initialize_model ()

last_observation ()

observe (*terminal, reward*)
Observe experience from the environment to learn from. Optionally pre-processes rewards Child classes should call super to get the processed reward EX: `terminal, reward = super()....`

Parameters

- **terminal** (*bool*) – boolean indicating if the episode terminated after the observation.
- **reward** (*float*) – scalar reward that resulted from executing the action.

reset ()
Reset the agent to its initial state (e.g. on experiment start). Updates the Model's internal episode and time step counter, internal states, and resets preprocessors.

restore_model (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

save_model (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** (*str*) – Optional checkpoint directory.
- **append_timestep** (*bool*) – Appends the current timestep to the checkpoint file if true. If this is set to True, the load path must include the checkpoint timestep suffix. For example, if stored to models/ and set to true, the exported file will be of the form models/model.ckpt-X where X is the last timestep saved. The load path must precisely match this file name. If this option is turned off, the checkpoint will always overwrite the file specified in path and the model can always be loaded under this path.

Returns Checkpoint path where the model was saved.

set_normalized_actions (*actions*)

set_normalized_states (*states*)

should_stop ()

```
class tensorflow.agents.LearningAgent (states, actions, network, update_mode,  
                                     memory, optimizer, batched_observe=True,  
                                     batching_capacity=1000, scope='learning-  
                                     agent', device=None, saver=None, sum-  
                                     marizer=None, execution=None, vari-  
                                     able_noise=None, states_preprocessing=None,  
                                     actions_exploration=None, re-  
                                     ward_preprocessing=None, discount=0.99, dis-  
                                     tributions=None, entropy_regularization=None)
```

Bases: `tensorflow.agents.agent.Agent`

Base class for learning agents, using as model a subclass of MemoryModel and DistributionModel.

```
__init__ (states, actions, network, update_mode, memory, optimizer, batched_observe=True,  
          batching_capacity=1000, scope='learning-agent', device=None, saver=None, summa-  
          rizer=None, execution=None, variable_noise=None, states_preprocessing=None, ac-  
          tions_exploration=None, reward_preprocessing=None, discount=0.99, distributions=None,  
          entropy_regularization=None)
```

Initializes the learning agent.

Parameters **update_mode** – Update mode specification, with the following attributes (required):

Parameters

- **memory** (*spec*) – Memory specification, see core.memories module for more information (required).

- **optimizer** (*spec*) – Optimizer specification, see `core.optimizers` module for more information (required).
- **network** (*spec*) – Network specification, usually a list of layer specifications, see `core.networks` module for more information (required).
- **scope** (*str*) – TensorFlow scope (default: name of agent).
- **device** – TensorFlow device (default: none)
- **saver** – Saver specification, with the following attributes (default: none):

Parameters **summarizer** – Summarizer specification, with the following attributes (default: none):

Parameters **execution** – Distributed specification, with the following attributes (default: none):

Parameters

- **variable_noise** (*float*) – Standard deviation of variable noise (default: none).
- **states_preprocessing** (*spec, or dict of specs*) – States preprocessing specification, see `core.preprocessors` module for more information (default: none)
- **actions_exploration** (*spec, or dict of specs*) – Actions exploration specification, see `core.explorations` module for more information (default: none).
- **reward_preprocessing** (*spec*) – Reward preprocessing specification, see `core.preprocessors` module for more information (default: none).
- **discount** (*float*) – Discount factor for future rewards (default: 0.99).
- **distributions** (*spec / dict of specs*) – Distributions specifications, see `core.distributions` module for more information (default: none).
- **entropy_regularization** (*float*) – Entropy regularization weight (default: none).

act (*states, deterministic=False, independent=False, fetch_tensors=None*)

Return action(s) for given state(s). States preprocessing and exploration are applied if configured accordingly.

Parameters

- **states** (*any*) – One state (usually a value tuple) or dict of states if multiple states are expected.
- **deterministic** (*bool*) – If true, no exploration and sampling is applied.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).
- **fetch_tensors** (*list*) – Optional String of named tensors to fetch

Returns Scalar value of the action or dict of multiple actions the agent wants to execute. (fetched_tensors) Optional dict() with named tensors fetched

close ()

from_spec (*spec, kwargs*)

Creates an agent from a specification dict.

import_experience (*experiences*)

Imports experiences.

Parameters *experiences* –

initialize_model ()

Creates the model for the respective agent based on specifications given by user. This is a separate call after constructing the agent because the agent constructor has to perform a number of checks on the specs first, sometimes adjusting them e.g. by converting to a dict.

last_observation ()

observe (*terminal*, *reward*)

Observe experience from the environment to learn from. Optionally pre-processes rewards Child classes should call super to get the processed reward EX: `terminal, reward = super(). . .`

Parameters

- **terminal** (*bool*) – boolean indicating if the episode terminated after the observation.
- **reward** (*float*) – scalar reward that resulted from executing the action.

reset ()

Reset the agent to its initial state (e.g. on experiment start). Updates the Model's internal episode and time step counter, internal states, and resets preprocessors.

restore_model (*directory=None*, *file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

save_model (*directory=None*, *append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** (*str*) – Optional checkpoint directory.
- **append_timestep** (*bool*) – Appends the current timestep to the checkpoint file if true. If this is set to True, the load path must include the checkpoint timestep suffix. For example, if stored to `models/` and set to true, the exported file will be of the form `models/model.ckpt-X` where X is the last timestep saved. The load path must precisely match this file name. If this option is turned off, the checkpoint will always overwrite the file specified in path and the model can always be loaded under this path.

Returns Checkpoint path where the model was saved.

set_normalized_actions (*actions*)

set_normalized_states (*states*)

should_stop ()

```
class tensorforce.agents.DQFDAgent (states, actions, network, batched_observe=True,
                                   batching_capacity=1000, scope='dqfd', device=None,
                                   saver=None, summarizer=None, execution=None,
                                   variable_noise=None, states_preprocessing=None,
                                   actions_exploration=None, reward_preprocessing=None,
                                   update_mode=None, memory=None, optimizer=None,
                                   discount=0.99, distributions=None,
                                   entropy_regularization=None, target_sync_frequency=10000,
                                   target_update_weight=1.0, huber_loss=None,
                                   expert_margin=0.5, supervised_weight=0.1,
                                   demo_memory_capacity=10000, demo_sampling_ratio=0.2)
```

Bases: `tensorforce.agents.learning_agent.LearningAgent`

Deep Q-learning from demonstration agent (Hester et al., 2017).

```
__init__ (states, actions, network, batched_observe=True, batching_capacity=1000, scope='dqfd',
          device=None, saver=None, summarizer=None, execution=None, variable_noise=None,
          states_preprocessing=None, actions_exploration=None, reward_preprocessing=None,
          update_mode=None, memory=None, optimizer=None, discount=0.99, distributions=None,
          entropy_regularization=None, target_sync_frequency=10000, target_update_weight=1.0,
          huber_loss=None, expert_margin=0.5, supervised_weight=0.1, demo_memory_capacity=10000,
          demo_sampling_ratio=0.2)
```

Initializes the DQFD agent.

Parameters `update_mode` – Update mode specification, with the following attributes:

Parameters

- **memory** (*spec*) – Memory specification, see `core.memories` module for more information (default: `{type='replay', include_next_states=True, capacity=1000*batch_size}`).
- **optimizer** (*spec*) – Optimizer specification, see `core.optimizers` module for more information (default: `{type='adam', learning_rate=1e-3}`).
- **target_sync_frequency** (*int*) – Target network sync frequency (default: 10000).
- **target_update_weight** (*float*) – Target network update weight (default: 1.0).
- **huber_loss** (*float*) – Huber loss clipping (default: none).
- **expert_margin** (*float*) – Enforced supervised margin between expert action Q-value and other Q-values (default: 0.5).
- **supervised_weight** (*float*) – Weight of supervised loss term (default: 0.1).
- **demo_memory_capacity** (*int*) – Capacity of expert demonstration memory (default: 10000).
- **demo_sampling_ratio** (*float*) – Runtime sampling ratio of expert data (default: 0.2).

act (*states*, *deterministic=False*, *independent=False*, *fetch_tensors=None*)

Return action(s) for given state(s). States preprocessing and exploration are applied if configured accordingly.

Parameters

- **states** (*any*) – One state (usually a value tuple) or dict of states if multiple states are expected.
- **deterministic** (*bool*) – If true, no exploration and sampling is applied.

- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).
- **fetch_tensors** (*list*) – Optional String of named tensors to fetch

Returns Scalar value of the action or dict of multiple actions the agent wants to execute.
(*fetch_tensors*) Optional dict() with named tensors fetched

close()

from_spec (*spec, kwargs*)

Creates an agent from a specification dict.

import_demonstrations (*demonstrations*)

Imports demonstrations, i.e. expert observations. Note that for large numbers of observations, `set_demonstrations` is more appropriate, which directly sets memory contents to an array an expects a different layout.

Parameters **demonstrations** – List of observation dicts

import_experience (*experiences*)

Imports experiences.

Parameters **experiences** –

initialize_model()

last_observation()

observe (*terminal, reward*)

Observe experience from the environment to learn from. Optionally pre-processes rewards Child classes should call super to get the processed reward EX: `terminal, reward = super(). . .`

Parameters

- **terminal** (*bool*) – boolean indicating if the episode terminated after the observation.
- **reward** (*float*) – scalar reward that resulted from executing the action.

pretrain (*steps*)

Computes pre-train updates.

Parameters **steps** – Number of updates to execute.

reset()

Reset the agent to its initial state (e.g. on experiment start). Updates the Model's internal episode and time step counter, internal states, and resets preprocessors.

restore_model (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

save_model (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** (*str*) – Optional checkpoint directory.

- **append_timestep** (*bool*) – Appends the current timestep to the checkpoint file if true. If this is set to True, the load path must include the checkpoint timestep suffix. For example, if stored to models/ and set to true, the exported file will be of the form models/model.ckpt-X where X is the last timestep saved. The load path must precisely match this file name. If this option is turned off, the checkpoint will always overwrite the file specified in path and the model can always be loaded under this path.

Returns Checkpoint path where the model was saved.

set_normalized_actions (*actions*)

set_normalized_states (*states*)

should_stop ()

```
class tensorflowforce.agents.DQNAgent (states, actions, network, batched_observe=True,
                                       batching_capacity=1000, scope='dqn', device=None,
                                       saver=None, summarizer=None, execution=None,
                                       variable_noise=None, states_preprocessing=None,
                                       actions_exploration=None, reward_preprocessing=None,
                                       update_mode=None, memory=None, optimizer=None,
                                       discount=0.99, distributions=None,
                                       entropy_regularization=None, target_sync_frequency=10000,
                                       target_update_weight=1.0, double_q_model=False,
                                       huber_loss=None)
```

Bases: `tensorflowforce.agents.learning_agent.LearningAgent`

Deep Q-Network agent (Mnih et al., 2015).

```
__init__ (states, actions, network, batched_observe=True, batching_capacity=1000, scope='dqn',
          device=None, saver=None, summarizer=None, execution=None, variable_noise=None,
          states_preprocessing=None, actions_exploration=None, reward_preprocessing=None,
          update_mode=None, memory=None, optimizer=None, discount=0.99, distributions=None,
          entropy_regularization=None, target_sync_frequency=10000, target_update_weight=1.0,
          double_q_model=False, huber_loss=None)
```

Initializes the DQN agent.

Parameters **update_mode** – Update mode specification, with the following attributes:

Parameters

- **memory** (*spec*) – Memory specification, see core.memories module for more information (default: {type='replay', include_next_states=true, capacity=1000*batch_size}).
- **optimizer** (*spec*) – Optimizer specification, see core.optimizers module for more information (default: {type='adam', learning_rate=1e-3}).
- **target_sync_frequency** (*int*) – Target network sync frequency (default: 10000).
- **target_update_weight** (*float*) – Target network update weight (default: 1.0).
- **double_q_model** (*bool*) – Specifies whether double DQN mode is used (default: false).
- **huber_loss** (*float*) – Huber loss clipping (default: none).

act (*states*, *deterministic=False*, *independent=False*, *fetch_tensors=None*)

Return action(s) for given state(s). States preprocessing and exploration are applied if configured accordingly.

Parameters

- **states** (*any*) – One state (usually a value tuple) or dict of states if multiple states are expected.
- **deterministic** (*bool*) – If true, no exploration and sampling is applied.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).
- **fetch_tensors** (*list*) – Optional String of named tensors to fetch

Returns Scalar value of the action or dict of multiple actions the agent wants to execute.
(*fetch_tensors*) Optional dict() with named tensors fetched

close()

from_spec (*spec, kwargs*)

Creates an agent from a specification dict.

import_experience (*experiences*)

Imports experiences.

Parameters experiences –

initialize_model()

last_observation()

observe (*terminal, reward*)

Observe experience from the environment to learn from. Optionally pre-processes rewards Child classes should call super to get the processed reward EX: `terminal, reward = super(). . .`

Parameters

- **terminal** (*bool*) – boolean indicating if the episode terminated after the observation.
- **reward** (*float*) – scalar reward that resulted from executing the action.

reset()

Reset the agent to its initial state (e.g. on experiment start). Updates the Model's internal episode and time step counter, internal states, and resets preprocessors.

restore_model (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

save_model (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** (*str*) – Optional checkpoint directory.
- **append_timestep** (*bool*) – Appends the current timestep to the checkpoint file if true. If this is set to True, the load path must include the checkpoint timestep suffix. For example, if stored to `models/` and set to true, the exported file will be of the form `models/model.ckpt-X` where X is the last timestep saved. The load path must precisely

match this file name. If this option is turned off, the checkpoint will always overwrite the file specified in path and the model can always be loaded under this path.

Returns Checkpoint path where the model was saved.

set_normalized_actions (*actions*)

set_normalized_states (*states*)

should_stop ()

```
class tensorflowforce.agents.DQNnStepAgent (states, actions, network, batched_observe=True,  
                                           batching_capacity=1000, scope='dqn-  
                                           nstep', device=None, saver=None, sum-  
                                           marizer=None, execution=None, vari-  
                                           able_noise=None, states_preprocessing=None,  
                                           actions_exploration=None, re-  
                                           ward_preprocessing=None, update_mode=None,  
                                           memory=None, optimizer=None, discount=0.99,  
                                           distributions=None, entropy_regularization=None,  
                                           target_sync_frequency=10000, tar-  
                                           get_update_weight=1.0, double_q_model=False,  
                                           huber_loss=None)
```

Bases: `tensorflowforce.agents.learning_agent.LearningAgent`

DQN n-step agent.

```
__init__ (states, actions, network, batched_observe=True, batching_capacity=1000,  
          scope='dqn-nstep', device=None, saver=None, summarizer=None, execution=None,  
          variable_noise=None, states_preprocessing=None, actions_exploration=None,  
          reward_preprocessing=None, update_mode=None, memory=None, opti-  
          mizer=None, discount=0.99, distributions=None, entropy_regularization=None, tar-  
          get_sync_frequency=10000, target_update_weight=1.0, double_q_model=False, hu-  
          ber_loss=None)
```

Initializes the DQN n-step agent.

Parameters **update_mode** – Update mode specification, with the following attributes:

Parameters

- **memory** (*spec*) – Memory specification, see `core.memories` module for more information (default: `{type='latest', include_next_states=True, capacity=1000*batch_size}`).
- **optimizer** (*spec*) – Optimizer specification, see `core.optimizers` module for more information (default: `{type='adam', learning_rate=1e-3}`).
- **target_sync_frequency** (*int*) – Target network sync frequency (default: 10000).
- **target_update_weight** (*float*) – Target network update weight (default: 1.0).
- **double_q_model** (*bool*) – Specifies whether double DQN mode is used (default: `false`).
- **huber_loss** (*float*) – Huber loss clipping (default: `none`).

act (*states, deterministic=False, independent=False, fetch_tensors=None*)

Return action(s) for given state(s). States preprocessing and exploration are applied if configured accordingly.

Parameters

- **states** (*any*) – One state (usually a value tuple) or dict of states if multiple states are expected.
- **deterministic** (*bool*) – If true, no exploration and sampling is applied.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).
- **fetch_tensors** (*list*) – Optional String of named tensors to fetch

Returns Scalar value of the action or dict of multiple actions the agent wants to execute.
(*fetch_tensors*) Optional dict() with named tensors fetched

close()

from_spec (*spec, kwargs*)

Creates an agent from a specification dict.

import_experience (*experiences*)

Imports experiences.

Parameters experiences –

initialize_model()

last_observation()

observe (*terminal, reward*)

Observe experience from the environment to learn from. Optionally pre-processes rewards Child classes should call super to get the processed reward EX: `terminal, reward = super(). . .`

Parameters

- **terminal** (*bool*) – boolean indicating if the episode terminated after the observation.
- **reward** (*float*) – scalar reward that resulted from executing the action.

reset()

Reset the agent to its initial state (e.g. on experiment start). Updates the Model's internal episode and time step counter, internal states, and resets preprocessors.

restore_model (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

save_model (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** (*str*) – Optional checkpoint directory.
- **append_timestep** (*bool*) – Appends the current timestep to the checkpoint file if true. If this is set to True, the load path must include the checkpoint timestep suffix. For example, if stored to `models/` and set to true, the exported file will be of the form `models/model.ckpt-X` where X is the last timestep saved. The load path must precisely

match this file name. If this option is turned off, the checkpoint will always overwrite the file specified in path and the model can always be loaded under this path.

Returns Checkpoint path where the model was saved.

set_normalized_actions (*actions*)

set_normalized_states (*states*)

should_stop ()

```
class tensorforce.agents.NAFAgent (states, actions, network, batched_observe=True,
    batching_capacity=1000, scope='naf', device=None,
    saver=None, summarizer=None, execution=None,
    variable_noise=None, states_preprocessing=None, ac-
    tions_exploration=None, reward_preprocessing=None,
    update_mode=None, memory=None, opti-
    mizer=None, discount=0.99, distributions=None, en-
    tropy_regularization=None, target_sync_frequency=10000,
    target_update_weight=1.0, double_q_model=False, hu-
    ber_loss=None)
```

Bases: `tensorforce.agents.learning_agent.LearningAgent`

Normalized Advantage Function agent (Gu et al., 2016).

```
__init__ (states, actions, network, batched_observe=True, batching_capacity=1000, scope='naf',
    device=None, saver=None, summarizer=None, execution=None, variable_noise=None,
    states_preprocessing=None, actions_exploration=None, reward_preprocessing=None, up-
    date_mode=None, memory=None, optimizer=None, discount=0.99, distributions=None,
    entropy_regularization=None, target_sync_frequency=10000, target_update_weight=1.0,
    double_q_model=False, huber_loss=None)
```

Initializes the NAF agent.

Parameters **update_mode** – Update mode specification, with the following attributes:

Parameters

- **memory** (*spec*) – Memory specification, see `core.memories` module for more information (default: `{type='replay', include_next_states=True, capacity=1000*batch_size}`).
- **optimizer** (*spec*) – Optimizer specification, see `core.optimizers` module for more information (default: `{type='adam', learning_rate=1e-3}`).
- **target_sync_frequency** (*int*) – Target network sync frequency (default: 10000).
- **target_update_weight** (*float*) – Target network update weight (default: 1.0).
- **double_q_model** (*bool*) – Specifies whether double DQN mode is used (default: `false`).
- **huber_loss** (*float*) – Huber loss clipping (default: `none`).

act (*states*, *deterministic=False*, *independent=False*, *fetch_tensors=None*)

Return action(s) for given state(s). States preprocessing and exploration are applied if configured accordingly.

Parameters

- **states** (*any*) – One state (usually a value tuple) or dict of states if multiple states are expected.
- **deterministic** (*bool*) – If true, no exploration and sampling is applied.

- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).
- **fetch_tensors** (*list*) – Optional String of named tensors to fetch

Returns Scalar value of the action or dict of multiple actions the agent wants to execute.
(*fetch_tensors*) Optional dict() with named tensors fetched

close()

from_spec (*spec, kwargs*)

Creates an agent from a specification dict.

import_experience (*experiences*)

Imports experiences.

Parameters *experiences* –

initialize_model()

last_observation()

observe (*terminal, reward*)

Observe experience from the environment to learn from. Optionally pre-processes rewards Child classes should call super to get the processed reward EX: `terminal, reward = super(). . .`

Parameters

- **terminal** (*bool*) – boolean indicating if the episode terminated after the observation.
- **reward** (*float*) – scalar reward that resulted from executing the action.

reset()

Reset the agent to its initial state (e.g. on experiment start). Updates the Model's internal episode and time step counter, internal states, and resets preprocessors.

restore_model (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

save_model (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** (*str*) – Optional checkpoint directory.
- **append_timestep** (*bool*) – Appends the current timestep to the checkpoint file if true. If this is set to True, the load path must include the checkpoint timestep suffix. For example, if stored to `models/` and set to true, the exported file will be of the form `models/model.ckpt-X` where X is the last timestep saved. The load path must precisely match this file name. If this option is turned off, the checkpoint will always overwrite the file specified in path and the model can always be loaded under this path.

Returns Checkpoint path were the model was saved.

set_normalized_actions (*actions*)

```
set_normalized_states (states)
```

```
should_stop ()
```

```
class tensorforce.agents.PPOAgent (states, actions, network, batched_observe=True,
    batching_capacity=1000, scope='ppo', device=None, saver=None, summarizer=None,
    execution=None, variable_noise=None, states_preprocessing=None, actions_exploration=None,
    reward_preprocessing=None, update_mode=None, memory=None, discount=0.99,
    distributions=None, entropy_regularization=None, baseline_mode=None, baseline=None,
    baseline_optimizer=None, gae_lambda=None, likelihood_ratio_clipping=0.2,
    step_optimizer=None, subsampling_fraction=0.1, optimization_steps=50)
```

Bases: `tensorforce.agents.learning_agent.LearningAgent`

Proximal Policy Optimization agent (Schulman et al., 2017).

```
__init__ (states, actions, network, batched_observe=True, batching_capacity=1000,
    scope='ppo', device=None, saver=None, summarizer=None, execution=None,
    variable_noise=None, states_preprocessing=None, actions_exploration=None,
    reward_preprocessing=None, update_mode=None, memory=None, discount=0.99,
    distributions=None, entropy_regularization=None, baseline_mode=None, baseline=None,
    baseline_optimizer=None, gae_lambda=None, likelihood_ratio_clipping=0.2,
    step_optimizer=None, subsampling_fraction=0.1, optimization_steps=50)
```

Initializes the PPO agent.

Parameters `update_mode` – Update mode specification, with the following attributes:

Parameters

- **memory** (*spec*) – Memory specification, see `core.memories` module for more information (default: `{type='latest', include_next_states=False, capacity=1000*batch_size}`).
- **optimizer** (*spec*) – PPO agent implicitly defines a multi-step subsampling optimizer.
- **baseline_mode** (*str*) – One of 'states', 'network' (default: none).
- **baseline** (*spec*) – Baseline specification, see `core.baselines` module for more information (default: none).
- **baseline_optimizer** (*spec*) – Baseline optimizer specification, see `core.optimizers` module for more information (default: none).
- **gae_lambda** (*float*) – Lambda factor for generalized advantage estimation (default: none).
- **likelihood_ratio_clipping** (*float*) – Likelihood ratio clipping for policy gradient (default: 0.2).
- **step_optimizer** (*spec*) – Step optimizer specification of implicit multi-step subsampling optimizer, see `core.optimizers` module for more information (default: `{type='adam', learning_rate=1e-3}`).
- **subsampling_fraction** (*float*) – Subsampling fraction of implicit subsampling optimizer (default: 0.1).
- **optimization_steps** (*int*) – Number of optimization steps for implicit multi-step optimizer (default: 50).

act (*states*, *deterministic=False*, *independent=False*, *fetch_tensors=None*)

Return action(s) for given state(s). States preprocessing and exploration are applied if configured accordingly.

Parameters

- **states** (*any*) – One state (usually a value tuple) or dict of states if multiple states are expected.
- **deterministic** (*bool*) – If true, no exploration and sampling is applied.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).
- **fetch_tensors** (*list*) – Optional String of named tensors to fetch

Returns Scalar value of the action or dict of multiple actions the agent wants to execute.
(*fetch_tensors*) Optional dict() with named tensors fetched

close ()

from_spec (*spec*, *kwargs*)

Creates an agent from a specification dict.

import_experience (*experiences*)

Imports experiences.

Parameters *experiences* –

initialize_model ()

last_observation ()

observe (*terminal*, *reward*)

Observe experience from the environment to learn from. Optionally pre-processes rewards Child classes should call super to get the processed reward EX: `terminal, reward = super(). . .`

Parameters

- **terminal** (*bool*) – boolean indicating if the episode terminated after the observation.
- **reward** (*float*) – scalar reward that resulted from executing the action.

reset ()

Reset the agent to its initial state (e.g. on experiment start). Updates the Model's internal episode and time step counter, internal states, and resets preprocessors.

restore_model (*directory=None*, *file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

save_model (*directory=None*, *append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** (*str*) – Optional checkpoint directory.

- **append_timestep** (*bool*) – Appends the current timestep to the checkpoint file if true. If this is set to True, the load path must include the checkpoint timestep suffix. For example, if stored to models/ and set to true, the exported file will be of the form models/model.ckpt-X where X is the last timestep saved. The load path must precisely match this file name. If this option is turned off, the checkpoint will always overwrite the file specified in path and the model can always be loaded under this path.

Returns Checkpoint path where the model was saved.

set_normalized_actions (*actions*)

set_normalized_states (*states*)

should_stop ()

```
class tensorflow.agents.TRPOAgent (states, actions, network, batched_observe=True, batch-  
ing_capacity=1000, scope='trpo', device=None,  
saver=None, summarizer=None, execution=None,  
variable_noise=None, states_preprocessing=None, ac-  
tions_exploration=None, reward_preprocessing=None,  
update_mode=None, memory=None, discount=0.99,  
distributions=None, entropy_regularization=None,  
baseline_mode=None, baseline=None, base-  
line_optimizer=None, gae_lambda=None, likeli-  
hood_ratio_clipping=None, learning_rate=0.001,  
cg_max_iterations=20, cg_damping=0.001,  
cg_unroll_loop=False, ls_max_iterations=10,  
ls_accept_ratio=0.9, ls_unroll_loop=False)
```

Bases: [tensorflow.agents.learning_agent.LearningAgent](#)

Trust Region Policy Optimization agent (Schulman et al., 2015).

```
__init__ (states, actions, network, batched_observe=True, batching_capacity=1000,  
scope='trpo', device=None, saver=None, summarizer=None, execution=None,  
variable_noise=None, states_preprocessing=None, actions_exploration=None, re-  
ward_preprocessing=None, update_mode=None, memory=None, discount=0.99, dis-  
tributions=None, entropy_regularization=None, baseline_mode=None, baseline=None,  
baseline_optimizer=None, gae_lambda=None, likelihood_ratio_clipping=None, learn-  
ing_rate=0.001, cg_max_iterations=20, cg_damping=0.001, cg_unroll_loop=False,  
ls_max_iterations=10, ls_accept_ratio=0.9, ls_unroll_loop=False)
```

Initializes the TRPO agent.

Parameters **update_mode** – Update mode specification, with the following attributes:

Parameters

- **memory** (*spec*) – Memory specification, see core.memories module for more information (default: {type='latest', include_next_states=false, capacity=1000*batch_size}).
- **optimizer** (*spec*) – TRPO agent implicitly defines a optimized-step natural-gradient optimizer.
- **baseline_mode** (*str*) – One of 'states', 'network' (default: none).
- **baseline** (*spec*) – Baseline specification, see core.baselines module for more information (default: none).
- **baseline_optimizer** (*spec*) – Baseline optimizer specification, see core.optimizers module for more information (default: none).

- **gae_lambda** (*float*) – Lambda factor for generalized advantage estimation (default: none).
- **likelihood_ratio_clipping** (*float*) – Likelihood ratio clipping for policy gradient (default: none).
- **learning_rate** (*float*) – Learning rate of natural-gradient optimizer (default: 1e-3).
- **cg_max_iterations** (*int*) – Conjugate-gradient max iterations (default: 20).
- **cg_damping** (*float*) – Conjugate-gradient damping (default: 1e-3).
- **cg_unroll_loop** (*bool*) – Conjugate-gradient unroll loop (default: false).
- **ls_max_iterations** (*int*) – Line-search max iterations (default: 10).
- **ls_accept_ratio** (*float*) – Line-search accept ratio (default: 0.9).
- **ls_unroll_loop** (*bool*) – Line-search unroll loop (default: false).

act (*states*, *deterministic=False*, *independent=False*, *fetch_tensors=None*)

Return action(s) for given state(s). States preprocessing and exploration are applied if configured accordingly.

Parameters

- **states** (*any*) – One state (usually a value tuple) or dict of states if multiple states are expected.
- **deterministic** (*bool*) – If true, no exploration and sampling is applied.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).
- **fetch_tensors** (*list*) – Optional String of named tensors to fetch

Returns Scalar value of the action or dict of multiple actions the agent wants to execute.
(*fetch_tensors*) Optional dict() with named tensors fetched

close ()

from_spec (*spec*, *kwargs*)

Creates an agent from a specification dict.

import_experience (*experiences*)

Imports experiences.

Parameters *experiences* –

initialize_model ()

last_observation ()

observe (*terminal*, *reward*)

Observe experience from the environment to learn from. Optionally pre-processes rewards Child classes should call super to get the processed reward EX: `terminal, reward = super()....`

Parameters

- **terminal** (*bool*) – boolean indicating if the episode terminated after the observation.
- **reward** (*float*) – scalar reward that resulted from executing the action.

reset ()

Reset the agent to its initial state (e.g. on experiment start). Updates the Model's internal episode and time step counter, internal states, and resets preprocessors.

restore_model (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

save_model (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** (*str*) – Optional checkpoint directory.
- **append_timestep** (*bool*) – Appends the current timestep to the checkpoint file if true. If this is set to True, the load path must include the checkpoint timestep suffix. For example, if stored to models/ and set to true, the exported file will be of the form models/model.ckpt-X where X is the last timestep saved. The load path must precisely match this file name. If this option is turned off, the checkpoint will always overwrite the file specified in path and the model can always be loaded under this path.

Returns Checkpoint path where the model was saved.

set_normalized_actions (*actions*)

set_normalized_states (*states*)

should_stop ()

```
class tensorflow.agents.VPGAgent (states, actions, network, batched_observe=True,  
                                batching_capacity=1000, scope='vpg', device=None,  
                                saver=None, summarizer=None, execution=None,  
                                variable_noise=None, states_preprocessing=None, ac-  
                                tions_exploration=None, reward_preprocessing=None,  
                                update_mode=None, memory=None, opti-  
                                mizer=None, discount=0.99, distributions=None, en-  
                                tropy_regularization=None, baseline_mode=None, base-  
                                line=None, baseline_optimizer=None, gae_lambda=None)
```

Bases: `tensorflow.agents.learning_agent.LearningAgent`

Vanilla policy gradient agent (Williams, 1992)).

```
__init__ (states, actions, network, batched_observe=True, batching_capacity=1000, scope='vpg',  
          device=None, saver=None, summarizer=None, execution=None, variable_noise=None,  
          states_preprocessing=None, actions_exploration=None, reward_preprocessing=None,  
          update_mode=None, memory=None, optimizer=None, discount=0.99, distribu-  
          tions=None, entropy_regularization=None, baseline_mode=None, baseline=None,  
          baseline_optimizer=None, gae_lambda=None)
```

Initializes the VPG agent.

Parameters **update_mode** – Update mode specification, with the following attributes:

Parameters

- **memory** (*spec*) – Memory specification, see core.memories module for more information (default: {type='latest', include_next_states=false, capacity=1000*batch_size}).

- **optimizer** (*spec*) – Optimizer specification, see `core.optimizers` module for more information (default: `{type='adam', learning_rate=1e-3}`).
- **baseline_mode** (*str*) – One of 'states', 'network' (default: none).
- **baseline** (*spec*) – Baseline specification, see `core.baselines` module for more information (default: none).
- **baseline_optimizer** (*spec*) – Baseline optimizer specification, see `core.optimizers` module for more information (default: none).
- **gae_lambda** (*float*) – Lambda factor for generalized advantage estimation (default: none).

act (*states*, *deterministic=False*, *independent=False*, *fetch_tensors=None*)

Return action(s) for given state(s). States preprocessing and exploration are applied if configured accordingly.

Parameters

- **states** (*any*) – One state (usually a value tuple) or dict of states if multiple states are expected.
- **deterministic** (*bool*) – If true, no exploration and sampling is applied.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).
- **fetch_tensors** (*list*) – Optional String of named tensors to fetch

Returns Scalar value of the action or dict of multiple actions the agent wants to execute.
(*fetch_tensors*) Optional dict() with named tensors fetched

close ()

from_spec (*spec*, *kwargs*)

Creates an agent from a specification dict.

import_experience (*experiences*)

Imports experiences.

Parameters *experiences* –

initialize_model ()

last_observation ()

observe (*terminal*, *reward*)

Observe experience from the environment to learn from. Optionally pre-processes rewards Child classes should call `super` to get the processed reward EX: `terminal, reward = super()....`

Parameters

- **terminal** (*bool*) – boolean indicating if the episode terminated after the observation.
- **reward** (*float*) – scalar reward that resulted from executing the action.

reset ()

Reset the agent to its initial state (e.g. on experiment start). Updates the Model's internal episode and time step counter, internal states, and resets preprocessors.

restore_model (*directory=None*, *file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

save_model (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** (*str*) – Optional checkpoint directory.
- **append_timestep** (*bool*) – Appends the current timestep to the checkpoint file if true. If this is set to True, the load path must include the checkpoint timestep suffix. For example, if stored to models/ and set to true, the exported file will be of the form models/model.ckpt-X where X is the last timestep saved. The load path must precisely match this file name. If this option is turned off, the checkpoint will always overwrite the file specified in path and the model can always be loaded under this path.

Returns Checkpoint path where the model was saved.

set_normalized_actions (*actions*)

set_normalized_states (*states*)

should_stop ()

```
class tensorflow.agents.DDPGAgent (states, actions, network, batched_observe=True, batching_capacity=1000, scope='ddpg', device=None, saver=None, summarizer=None, execution=None, variable_noise=None, states_preprocessing=None, actions_exploration=None, reward_preprocessing=None, update_mode=None, memory=None, optimizer=None, discount=0.99, distributions=None, entropy_regularization=None, critic_network=None, critic_optimizer=None, target_sync_frequency=10000, target_update_weight=1.0)
```

Bases: `tensorflow.agents.learning_agent.LearningAgent`

Deep Deterministic Policy Gradient agent (Lillicrap et al., 2015).

```
__init__ (states, actions, network, batched_observe=True, batching_capacity=1000, scope='ddpg', device=None, saver=None, summarizer=None, execution=None, variable_noise=None, states_preprocessing=None, actions_exploration=None, reward_preprocessing=None, update_mode=None, memory=None, optimizer=None, discount=0.99, distributions=None, entropy_regularization=None, critic_network=None, critic_optimizer=None, target_sync_frequency=10000, target_update_weight=1.0)
```

Initializes the DDPG agent.

Parameters **update_mode** – Update mode specification, with the following attributes:

Parameters

- **memory** (*spec*) – Memory specification, see core.memories module for more information (default: {type='replay', include_next_states=true, capacity=1000*batch_size}).
- **optimizer** (*spec*) – Optimizer specification, see core.optimizers module for more information (default: {type='adam', learning_rate=1e-3}).

- **critic_network** (*spec*) – Critic network specification, usually a list of layer specifications, see `core.networks` module for more information (default: `network`).
- **critic_optimizer** (*spec*) – Critic optimizer specification, see `core.optimizers` module for more information (default: `{type='adam', learning_rate=1e-3}`).
- **target_sync_frequency** (*int*) – Target network sync frequency (default: 10000).
- **target_update_weight** (*float*) – Target network update weight (default: 1.0).

act (*states*, *deterministic=False*, *independent=False*, *fetch_tensors=None*)

Return action(s) for given state(s). States preprocessing and exploration are applied if configured accordingly.

Parameters

- **states** (*any*) – One state (usually a value tuple) or dict of states if multiple states are expected.
- **deterministic** (*bool*) – If true, no exploration and sampling is applied.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).
- **fetch_tensors** (*list*) – Optional String of named tensors to fetch

Returns Scalar value of the action or dict of multiple actions the agent wants to execute. (fetched_tensors) Optional dict() with named tensors fetched

close ()

from_spec (*spec*, *kwargs*)

Creates an agent from a specification dict.

import_experience (*experiences*)

Imports experiences.

Parameters experiences –

initialize_model ()

last_observation ()

observe (*terminal*, *reward*)

Observe experience from the environment to learn from. Optionally pre-processes rewards Child classes should call super to get the processed reward EX: `terminal, reward = super()....`

Parameters

- **terminal** (*bool*) – boolean indicating if the episode terminated after the observation.
- **reward** (*float*) – scalar reward that resulted from executing the action.

reset ()

Reset the agent to its initial state (e.g. on experiment start). Updates the Model's internal episode and time step counter, internal states, and resets preprocessors.

restore_model (*directory=None*, *file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

save_model (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** (*str*) – Optional checkpoint directory.
- **append_timestep** (*bool*) – Appends the current timestep to the checkpoint file if true. If this is set to True, the load path must include the checkpoint timestep suffix. For example, if stored to models/ and set to true, the exported file will be of the form models/model.ckpt-X where X is the last timestep saved. The load path must precisely match this file name. If this option is turned off, the checkpoint will always overwrite the file specified in path and the model can always be loaded under this path.

Returns Checkpoint path where the model was saved.

set_normalized_actions (*actions*)

set_normalized_states (*states*)

should_stop ()

tensorforce.contrib package

Submodules

tensorforce.contrib.ale module

```
class tensorforce.contrib.ale.ALE(rom, frame_skip=1, repeat_action_probability=0.0,  
                                loss_of_life_termination=False, loss_of_life_reward=0,  
                                display_screen=False, seed=<mtrand.RandomState ob-  
ject>)
```

Bases: *tensorforce.environments.environment.Environment*

Arcade Learning Environment (ALE). <https://github.com/mgbellemare/Arcade-Learning-Environment>

```
__init__(rom, frame_skip=1, repeat_action_probability=0.0, loss_of_life_termination=False,  
         loss_of_life_reward=0, display_screen=False, seed=<mtrand.RandomState object>)  
Initialize ALE.
```

Parameters

- **rom** – Rom filename and directory.
- **frame_skip** – Repeat action for n frames. Default 1.
- **repeat_action_probability** – Repeats last action with given probability. Default 0.
- **loss_of_life_termination** – Signals a terminal state on loss of life. Default False.
- **loss_of_life_reward** – Reward/Penalty on loss of life (negative values are a penalty). Default 0.
- **display_screen** – Displays the emulator screen. Default False.
- **seed** – Random seed

action_names

actions**close()****current_state****execute** (*actions*)**from_spec** (*spec, kwargs*)

Creates an environment from a specification dict.

is_terminal**reset()****seed** (*seed*)

Sets the random seed of the environment to the given value (current time, if seed=None). Naturally deterministic Environments (e.g. ALE or some gym Envs) don't have to implement this method.

Parameters **seed** (*int*) – The seed to use for initializing the pseudo-random number generator (default=epoch time in sec).

Returns: The actual seed (int) used OR None if Environment did not override this method (no seeding supported).

states

tensorforce.contrib.deepmind_lab module

```
class tensorforce.contrib.deepmind_lab.DeepMindLab(level_id, repeat_action=1,
                                                    state_attribute='RGB_INTERLACED',
                                                    settings={'width': '320', 'appendCommand': '', 'fps': '60', 'height': '240'})
```

Bases: *tensorforce.environments.environment.Environment*DeepMind Lab Integration: <https://arxiv.org/abs/1612.03801> <https://github.com/deepmind/lab>

Since DeepMind lab is only available as source code, a manual install via bazel is required. Further, due to the way bazel handles external dependencies, cloning TensorForce into lab is the most convenient way to run it using the bazel BUILD file we provide. To use lab, first download and install it according to instructions <https://github.com/deepmind/lab/blob/master/docs/build.md>:

```
git clone https://github.com/deepmind/lab.git
```

Add to the lab main BUILD file:

Clone TensorForce into the lab directory, then run the TensorForce bazel runner.

Note that using any specific configuration file currently requires changing the Tensorforce BUILD file to adjust environment parameters.

```
bazel run //tensorforce:lab_runner
```

Please note that we have not tried to reproduce any lab results yet, and these instructions just explain connectivity in case someone wants to get started there.

```
__init__ (level_id, repeat_action=1, state_attribute='RGB_INTERLACED', settings={'width': '320', 'appendCommand': '', 'fps': '60', 'height': '240'})
Initialize DeepMind Lab environment.
```

Parameters

- **level_id** – string with id/descriptor of the level, e.g. ‘seekavoid_arena_01’.
- **repeat_action** – number of frames the environment is advanced, executing the given action during every frame.
- **state_attribute** – Attributes which represents the state for this environment, should adhere to the specification given in `DeepMindLabEnvironment.state_spec(level_id)`.
- **settings** – dict specifying additional settings as key-value string pairs. The following options are recognized: ‘width’ (horizontal resolution of the observation frames), ‘height’ (vertical resolution of the observation frames), ‘fps’ (frames per second) and ‘appendCommand’ (commands for the internal Quake console).

actions

close()

Closes the environment and releases the underlying Quake III Arena instance. No other method calls possible afterwards.

execute(actions)

Pass action to universe environment, return reward, next step, terminal state and additional info.

Parameters **action** – action to execute as numpy array, should have dtype `np.intc` and should adhere to the specification given in `DeepMindLabEnvironment.action_spec(level_id)`

Returns dict containing the next state, the reward, and a boolean indicating if the next state is a terminal state

fps

An advisory metric that correlates discrete environment steps (“frames”) with real (wallclock) time: the number of frames per (real) second.

from_spec(spec, kwargs)

Creates an environment from a specification dict.

num_steps

Number of frames since the last `reset()` call.

reset()

Resets the environment to its initialization state. This method needs to be called to start a new episode after the last episode ended.

Returns initial state

seed(seed)

Sets the random seed of the environment to the given value (current time, if `seed=None`). Naturally deterministic Environments (e.g. ALE or some gym Envs) don’t have to implement this method.

Parameters **seed(int)** – The seed to use for initializing the pseudo-random number generator (default=epoch time in sec).

Returns: The actual seed (int) used OR None if Environment did not override this method (no seeding supported).

states

tensorforce.contrib.maze_explorer module

class `tensorforce.contrib.maze_explorer.MazeExplorer(mode_id=0, visible=True)`

Bases: `tensorforce.environments.environment.Environment`

MazeExplorer Integration: https://github.com/mryellow/maze_explorer.


```
__init__ (mode_id=0, visible=True)
```

Initialize MazeExplorer.

Parameters

- **mode_id** – Game mode ID. See https://github.com/mryellow/maze_explorer
- **visible** – Show output window

```
actions
```

```
close ()
```

```
execute (actions)
```

```
from_spec (spec, kwargs)
```

Creates an environment from a specification dict.

```
reset ()
```

```
seed (seed)
```

Sets the random seed of the environment to the given value (current time, if seed=None). Naturally deterministic Environments (e.g. ALE or some gym Envs) don't have to implement this method.

Parameters **seed** (*int*) – The seed to use for initializing the pseudo-random number generator (default=epoch time in sec).

Returns: The actual seed (int) used OR None if Environment did not override this method (no seeding supported).

```
states
```

tensorforce.contrib.openai_gym module

OpenAI Gym Integration: <https://gym.openai.com/>.

```
class tensorforce.contrib.openai_gym.OpenAIGym (gym_id, monitor=None, monitor_safe=False, monitor_video=0, visualize=False)
```

Bases: *tensorforce.environments.environment.Environment*

```
__init__ (gym_id, monitor=None, monitor_safe=False, monitor_video=0, visualize=False)
```

Initialize OpenAI Gym.

Parameters

- **gym_id** – OpenAI Gym environment ID. See <https://gym.openai.com/envs>
- **monitor** – Output directory. Setting this to None disables monitoring.
- **monitor_safe** – Setting this to True prevents existing log files to be overwritten. Default False.
- **monitor_video** – Save a video every monitor_video steps. Setting this to 0 disables recording of videos.
- **visualize** – If set True, the program will visualize the trainings of gym's environment. Note that such visualization is probably going to slow down the training.

```
static action_from_space (space)
```

```
actions
```

```
close ()
```

execute (*actions*)

from_spec (*spec, kwargs*)

Creates an environment from a specification dict.

reset ()

seed (*seed*)

Sets the random seed of the environment to the given value (current time, if seed=None). Naturally deterministic Environments (e.g. ALE or some gym Envs) don't have to implement this method.

Parameters **seed** (*int*) – The seed to use for initializing the pseudo-random number generator (default=epoch time in sec).

Returns: The actual seed (int) used OR None if Environment did not override this method (no seeding supported).

static state_from_space (*space*)

states

tensorforce.contrib.openai_universe module

class tensorforce.contrib.openai_universe.**OpenAIUniverse** (*env_id*)

Bases: *tensorforce.environments.environment.Environment*

OpenAI Universe Integration: <https://universe.openai.com/>. Contains OpenAI Gym: <https://gym.openai.com/>.

__init__ (*env_id*)

Initialize OpenAI universe environment.

Parameters **env_id** – string with id/descriptor of the universe environment, e.g. 'HarvestDay-v0'.

actions

close ()

configure (**args, **kwargs*)

execute (*actions*)

from_spec (*spec, kwargs*)

Creates an environment from a specification dict.

render (**args, **kwargs*)

reset ()

seed (*seed*)

Sets the random seed of the environment to the given value (current time, if seed=None). Naturally deterministic Environments (e.g. ALE or some gym Envs) don't have to implement this method.

Parameters **seed** (*int*) – The seed to use for initializing the pseudo-random number generator (default=epoch time in sec).

Returns: The actual seed (int) used OR None if Environment did not override this method (no seeding supported).

states

tensorforce.contrib.remote_environment module

class `tensorforce.contrib.remote_environment.MsgPackNumpyProtocol` (*max_msg_len=8192*)

Bases: `object`

A simple protocol to communicate over tcp sockets, which can be used by RemoteEnvironment implementations. The protocol is based on msgpack-numpy encoding and decoding.

Each message has a simple 8-byte header, which encodes the length of the subsequent msgpack-numpy encoded byte-string. All messages received need to have the 'status' field set to 'ok'. If 'status' is set to 'error', the field 'message' should be populated with some error information.

Examples: client sends: "[8-byte header]msgpack-encoded({"cmd": "seed", "value": 200})" server responds: "[8-byte header]msgpack-encoded({"status": "ok", "value": 200})"

client sends: "[8-byte header]msgpack-encoded({"cmd": "reset"})" server responds: "[8-byte header]msgpack-encoded({"status": "ok"})"

client sends: "[8-byte header]msgpack-encoded({"cmd": "step", "action": 5})" server responds: "[8-byte header]msgpack-encoded({"status": "ok", "obs_dict": {... some observations}, "reward": -10.0, "is_terminal": False})"

__init__ (*max_msg_len=8192*)

Parameters **max_msg_len** (*int*) – The maximum number of bytes to read from the socket.

recv (*socket_*)

Receives a message as msgpack-numpy encoded byte-string from the given socket object. Blocks until something was received.

Parameters **socket** – The python socket object to use.

Returns: The decoded (as dict) message received.

send (*message, socket_*)

Sends a message (dict) to the socket. Message consists of a 8-byte len header followed by a msgpack-numpy encoded dict.

Parameters

- **message** – The message dict (e.g. {"cmd": "reset"})
- **socket** – The python socket object to use.

class `tensorforce.contrib.remote_environment.RemoteEnvironment` (*host='localhost', port=6025*)

Bases: `tensorforce.environments.environment.Environment`

__init__ (*host='localhost', port=6025*)

A remote Environment that one can connect to through tcp. Implements a simple msgpack protocol to get the step/reset/etc.. commands to the remote server and simply waits (blocks) for a response.

Parameters

- **host** (*str*) – The hostname to connect to.
- **port** (*int*) – The port to connect to.

actions

Return the action space. Might include subdicts if multiple actions are available simultaneously.

Returns: dict of action properties (continuous, number of actions)

close()

Same as disconnect method.

connect (*timeout=600*)

Starts the server tcp connection on the given host:port.

Parameters **timeout** (*int*) – The time (in seconds) for which we will attempt a connection to the remote (every 5sec). After that (or if timeout is None or 0), an error is raised.

current_state

disconnect()

Ends our server tcp connection.

execute (*actions*)

Executes action, observes next state(s) and reward.

Parameters **actions** – Actions to execute.

Returns (Dict of) next state(s), boolean indicating terminal, and reward signal.

from_spec (*spec, kwargs*)

Creates an environment from a specification dict.

reset()

Reset environment and setup for new episode.

Returns initial state of reset environment.

seed (*seed*)

Sets the random seed of the environment to the given value (current time, if seed=None). Naturally deterministic Environments (e.g. ALE or some gym Envs) don't have to implement this method.

Parameters **seed** (*int*) – The seed to use for initializing the pseudo-random number generator (default=epoch time in sec).

Returns: The actual seed (int) used OR None if Environment did not override this method (no seeding supported).

states

Return the state space. Might include subdicts if multiple states are available simultaneously.

Returns: dict of state properties (shape and type).

tensorforce.contrib.state_settable_environment module

class tensorforce.contrib.state_settable_environment.StateSettableEnvironment

Bases: *tensorforce.environments.environment.Environment*

An Environment that implements the set_state method to set the current state to some new state using setter instructions.

__init__

x.init(...) initializes x; see help(type(x)) for signature

actions

Return the action space. Might include subdicts if multiple actions are available simultaneously.

Returns: dict of action properties (continuous, number of actions)

close()

Close environment. No other method calls possible afterwards.

execute (*actions*)

Executes action, observes next state(s) and reward.

Parameters *actions* – Actions to execute.

Returns (Dict of) next state(s), boolean indicating terminal, and reward signal.

from_spec (*spec, kwargs*)

Creates an environment from a specification dict.

reset ()

Reset environment and setup for new episode.

Returns initial state of reset environment.

seed (*seed*)

Sets the random seed of the environment to the given value (current time, if seed=None). Naturally deterministic Environments (e.g. ALE or some gym Envs) don't have to implement this method.

Parameters *seed* (*int*) – The seed to use for initializing the pseudo-random number generator (default=epoch time in sec).

Returns: The actual seed (int) used OR None if Environment did not override this method (no seeding supported).

set_state (***kwargs*)

Sets the current state of the environment manually to some other state and returns a new observation.

Parameters ***kwargs* –

The set instruction(s) to be executed by the environment. A single set instruction usually set a single property of the state/observation vector to some new value.

Returns: The observation dictionary of the Environment after(!) setting it to the new state.

states

Return the state space. Might include subdicts if multiple states are available simultaneously.

Returns: dict of state properties (shape and type).

tensorforce.contrib.unreal_engine module

```
class tensorforce.contrib.unreal_engine.UE4Environment (host='localhost',
                                                    port=6025, connect=True,
                                                    discretize_actions=False,
                                                    delta_time=0,
                                                    num_ticks=4)
```

Bases: `tensorforce.contrib.remote_environment.RemoteEnvironment`, `tensorforce.contrib.state_settable_environment.StateSettableEnvironment`

A special RemoteEnvironment for UE4 game connections. Communicates with the remote to receive information on the definitions of action- and observation spaces. Sends UE4 Action- and Axis-mappings as RL-actions and receives observations back defined by MLObserver objects placed in the Game (these could be camera pixels or other observations, e.g. a x/y/z position of some game actor).

```
__init__ (host='localhost', port=6025, connect=True, discretize_actions=False, delta_time=0,
          num_ticks=4)
```

Parameters

- **host** (*str*) – The hostname to connect to.

- **port** (*int*) – The port to connect to.
- **connect** (*bool*) – Whether to connect already in this c'tor.
- **discretize_actions** (*bool*) – Whether to treat axis-mappings defined in UE4 game as discrete actions. This would be necessary e.g. for agents that use q-networks where the output are q-values per discrete state-action pair.
- **delta_time** (*float*) – The fake delta time to use for each single game tick.
- **num_ticks** (*int*) – The number of ticks to be executed in a single act call (each tick will repeat the same given actions).

actions ()

close ()

Same as disconnect method.

connect (*timeout=600*)

current_state

disconnect ()

Ends our server tcp connection.

discretize_action_space_desc ()

Creates a list of discrete action(-combinations) in case we want to learn with a discrete set of actions, but only have action-combinations (maybe even continuous) available from the env. E.g. the UE4 game has the following action/axis-mappings:

```
{
  'Fire':
    {'type': 'action', 'keys': ('SpaceBar',)},
  'MoveRight':
    {'type': 'axis', 'keys': (('Right', 1.0), ('Left', -1.0), ('A', -1.0), ('D', 1.0))},
}
```

-> this method will discretize them into the following 6 discrete actions:

```
[
  [(Right, 0.0), (SpaceBar, False)],
  [(Right, 0.0), (SpaceBar, True)],
  [(Right, -1.0), (SpaceBar, False)],
  [(Right, -1.0), (SpaceBar, True)],
  [(Right, 1.0), (SpaceBar, False)],
  [(Right, 1.0), (SpaceBar, True)],
]
```

execute (*actions*)

Executes a single step in the UE4 game. This step may be comprised of one or more actual game ticks for all of which the same given action- and axis-inputs (or action number in case of discretized actions) are repeated. UE4 distinguishes between action-mappings, which are boolean actions (e.g. jump or dont-jump) and axis-mappings, which are continuous actions like MoveForward with values between -1.0 (run backwards) and 1.0 (run forwards), 0.0 would mean: stop.

static extract_observation (*message*)

from_spec (*spec, kwargs*)

Creates an environment from a specification dict.

reset ()

same as step (no kwargs to pass), but needs to block and return observation_dict

- stores the received observation in self.last_observation

seed (*seed=None*)

set_state (*setters, **kwargs*)

states ()

translate_abstract_actions_to_keys (*abstract*)

Translates a list of tuples ([pretty mapping], [value]) to a list of tuples ([some key], [translated value]) each single item in abstract will undergo the following translation:

Example1: we want: “MoveRight”: 5.0 possible keys for the action are: (“Right”, 1.0), (“Left”, -1.0)
result: “Right”: 5.0 * 1.0 = 5.0

Example2: we want: “MoveRight”: -0.5 possible keys for the action are: (“Left”, -1.0), (“Right”, 1.0)
result: “Left”: -0.5 * -1.0 = 0.5 (same as “Right”: -0.5)

Module contents

tensorforce.core package

Subpackages

tensorforce.core.baselines package

Submodules

tensorforce.core.baselines.aggregated_baseline module

```
class tensorforce.core.baselines.aggregated_baseline.AggregatedBaseline (baselines,  
                                                                    scope='aggregated-  
                                                                    baseline',  
                                                                    sum-  
                                                                    mary_labels=())
```

Bases: *tensorforce.core.baselines.baseline.Baseline*

Baseline which aggregates per-state baselines.

```
__init__ (baselines, scope='aggregated-baseline', summary_labels=())  
    Aggregated baseline.
```

Parameters **baselines** – Dict of per-state baseline specification dicts

```
from_spec (spec, kwargs=None)  
    Creates a baseline from a specification dict.
```

```
get_summaries ()
```

```
get_variables (include_nontrainable=False)
```

```
tf_loss (states, internals, reward, update, reference=None)  
    Creates the TensorFlow operations for calculating the L2 loss between predicted state values and actual rewards.
```

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor

tf_predict (*states, internals, update*)

tf_reference (*states, internals, reward, update*)

Creates the TensorFlow operations for obtaining the reference tensor(s), in case of a comparative loss.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Reference tensor(s).

tf_regularization_loss ()

tensorforce.core.baselines.baseline module

class tensorforce.core.baselines.baseline.**Baseline** (*scope='baseline', summary_labels=None*) *sum-*

Bases: object

Base class for baseline value functions.

__init__ (*scope='baseline', summary_labels=None*)
Baseline.

static from_spec (*spec, kwargs=None*)
Creates a baseline from a specification dict.

get_summaries ()
Returns the TensorFlow summaries reported by the baseline

Returns List of summaries

get_variables (*include_nontrainable=False*)
Returns the TensorFlow variables used by the baseline.

Returns List of variables

tf_loss (*states, internals, reward, update, reference=None*)
Creates the TensorFlow operations for calculating the L2 loss between predicted state values and actual rewards.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.

- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor

tf_predict (*states, internals, update*)

Creates the TensorFlow operations for predicting the value function of given states. :param states: Dict of state tensors. :param internals: List of prior internal state tensors. :param update: Boolean tensor indicating whether this call happens during an update.

Returns State value tensor

tf_reference (*states, internals, reward, update*)

Creates the TensorFlow operations for obtaining the reference tensor(s), in case of a comparative loss.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Reference tensor(s).

tf_regularization_loss ()

Creates the TensorFlow operations for the baseline regularization loss/

Returns Regularization loss tensor

tensorforce.core.baselines.cnn_baseline module

```
class tensorforce.core.baselines.cnn_baseline.CNNBaseline (conv_sizes, dense_sizes,
                                                         scope='cnn-baseline',
                                                         summary_labels=())
```

Bases: [tensorforce.core.baselines.network_baseline.NetworkBaseline](#)

CNN baseline (single-state) consisting of convolutional layers followed by dense layers.

```
__init__ (conv_sizes, dense_sizes, scope='cnn-baseline', summary_labels=())
CNN baseline.
```

Parameters

- **conv_sizes** – List of convolutional layer sizes
- **dense_sizes** – List of dense layer sizes

```
from_spec (spec, kwargs=None)
```

Creates a baseline from a specification dict.

```
get_summaries ()
```

```
get_variables (include_nontrainable=False)
```

```
tf_loss (states, internals, reward, update, reference=None)
```

Creates the TensorFlow operations for calculating the L2 loss between predicted state values and actual rewards.

Parameters

- **states** – Dict of state tensors.

- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor

tf_predict (*states, internals, update*)

tf_reference (*states, internals, reward, update*)

Creates the TensorFlow operations for obtaining the reference tensor(s), in case of a comparative loss.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Reference tensor(s).

tf_regularization_loss ()

tensorforce.core.baselines.mlp_baseline module

```
class tensorforce.core.baselines.mlp_baseline.MLPBaseline (sizes, scope='mlp-  
baseline', summary_labels=())
```

Bases: [tensorforce.core.baselines.network_baseline.NetworkBaseline](#)

Multi-layer perceptron baseline (single-state) consisting of dense layers.

```
__init__ (sizes, scope='mlp-baseline', summary_labels=())  
Multi-layer perceptron baseline.
```

Parameters **sizes** – List of dense layer sizes

```
from_spec (spec, kwargs=None)  
Creates a baseline from a specification dict.
```

```
get_summaries ()
```

```
get_variables (include_nontrainable=False)
```

```
tf_loss (states, internals, reward, update, reference=None)
```

Creates the TensorFlow operations for calculating the L2 loss between predicted state values and actual rewards.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor

tf_predict (*states, internals, update*)

tf_reference (*states, internals, reward, update*)

Creates the TensorFlow operations for obtaining the reference tensor(s), in case of a comparative loss.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Reference tensor(s).

tf_regularization_loss ()

tensorforce.core.baselines.network_baseline module

```
class tensorforce.core.baselines.network_baseline.NetworkBaseline(network,
                                                                scope='network-
                                                                baseline',
                                                                sum-
                                                                mary_labels=())
```

Bases: *tensorforce.core.baselines.baseline.Baseline*

Baseline based on a TensorForce network, used when parameters are shared between the value function and the baseline.

__init__ (*network, scope='network-baseline', summary_labels=()*)

Network baseline.

Parameters **network_spec** – Network specification dict

from_spec (*spec, kwargs=None*)

Creates a baseline from a specification dict.

get_summaries ()

get_variables (*include_nontrainable=False*)

tf_loss (*states, internals, reward, update, reference=None*)

Creates the TensorFlow operations for calculating the L2 loss between predicted state values and actual rewards.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor

tf_predict (*states, internals, update*)

tf_reference (*states, internals, reward, update*)

Creates the TensorFlow operations for obtaining the reference tensor(s), in case of a comparative loss.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Reference tensor(s).

tf_regularization_loss ()

Module contents

class `tensorforce.core.baselines.Baseline` (*scope='baseline', summary_labels=None*)

Bases: `object`

Base class for baseline value functions.

__init__ (*scope='baseline', summary_labels=None*)
Baseline.

static from_spec (*spec, kwargs=None*)
Creates a baseline from a specification dict.

get_summaries ()
Returns the TensorFlow summaries reported by the baseline

Returns List of summaries

get_variables (*include_nontrainable=False*)
Returns the TensorFlow variables used by the baseline.

Returns List of variables

tf_loss (*states, internals, reward, update, reference=None*)
Creates the TensorFlow operations for calculating the L2 loss between predicted state values and actual rewards.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor

tf_predict (*states, internals, update*)
Creates the TensorFlow operations for predicting the value function of given states. :param states: Dict of state tensors. :param internals: List of prior internal state tensors. :param update: Boolean tensor indicating whether this call happens during an update.

Returns State value tensor

tf_reference (*states, internals, reward, update*)
Creates the TensorFlow operations for obtaining the reference tensor(s), in case of a comparative loss.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Reference tensor(s).

tf_regularization_loss ()

Creates the TensorFlow operations for the baseline regularization loss/

Returns Regularization loss tensor

class tensorflow.core.baselines.**AggregatedBaseline** (*baselines*, *scope='aggregated-baseline'*, *summary_labels=()*)

Bases: *tensorflow.core.baselines.baseline.Baseline*

Baseline which aggregates per-state baselines.

__init__ (*baselines*, *scope='aggregated-baseline'*, *summary_labels=()*)

Aggregated baseline.

Parameters **baselines** – Dict of per-state baseline specification dicts

from_spec (*spec*, *kwargs=None*)

Creates a baseline from a specification dict.

get_summaries ()

get_variables (*include_nontrainable=False*)

tf_loss (*states*, *internals*, *reward*, *update*, *reference=None*)

Creates the TensorFlow operations for calculating the L2 loss between predicted state values and actual rewards.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor

tf_predict (*states*, *internals*, *update*)

tf_reference (*states*, *internals*, *reward*, *update*)

Creates the TensorFlow operations for obtaining the reference tensor(s), in case of a comparative loss.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Reference tensor(s).

```
tf_regularization_loss()
```

```
class tensorflow.core.baselines.NetworkBaseline(network, scope='network-baseline',  
                                              summary_labels=())
```

Bases: [tensorflow.core.baselines.baseline.Baseline](#)

Baseline based on a TensorFlow network, used when parameters are shared between the value function and the baseline.

```
__init__(network, scope='network-baseline', summary_labels=())  
    Network baseline.
```

Parameters **network_spec** – Network specification dict

```
from_spec(spec, kwargs=None)  
    Creates a baseline from a specification dict.
```

```
get_summaries()
```

```
get_variables(include_nontrainable=False)
```

```
tf_loss(states, internals, reward, update, reference=None)  
    Creates the TensorFlow operations for calculating the L2 loss between predicted state values and actual rewards.
```

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor

```
tf_predict(states, internals, update)
```

```
tf_reference(states, internals, reward, update)  
    Creates the TensorFlow operations for obtaining the reference tensor(s), in case of a comparative loss.
```

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Reference tensor(s).

```
tf_regularization_loss()
```

```
class tensorflow.core.baselines.MLPBaseline(sizes, scope='mlp-baseline', summary_labels=())
```

Bases: [tensorflow.core.baselines.network_baseline.NetworkBaseline](#)

Multi-layer perceptron baseline (single-state) consisting of dense layers.

```
__init__(sizes, scope='mlp-baseline', summary_labels=())  
    Multi-layer perceptron baseline.
```

Parameters **sizes** – List of dense layer sizes

from_spec (*spec*, *kwargs=None*)
Creates a baseline from a specification dict.

get_summaries ()

get_variables (*include_nontrainable=False*)

tf_loss (*states*, *internals*, *reward*, *update*, *reference=None*)
Creates the TensorFlow operations for calculating the L2 loss between predicted state values and actual rewards.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor

tf_predict (*states*, *internals*, *update*)

tf_reference (*states*, *internals*, *reward*, *update*)
Creates the TensorFlow operations for obtaining the reference tensor(s), in case of a comparative loss.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Reference tensor(s).

tf_regularization_loss ()

class `tensorforce.core.baselines.CNNBaseline` (*conv_sizes*, *dense_sizes*, *scope='cnn-baseline'*, *summary_labels=()*)

Bases: `tensorforce.core.baselines.network_baseline.NetworkBaseline`

CNN baseline (single-state) consisting of convolutional layers followed by dense layers.

__init__ (*conv_sizes*, *dense_sizes*, *scope='cnn-baseline'*, *summary_labels=()*)
CNN baseline.

Parameters

- **conv_sizes** – List of convolutional layer sizes
- **dense_sizes** – List of dense layer sizes

from_spec (*spec*, *kwargs=None*)
Creates a baseline from a specification dict.

get_summaries ()

get_variables (*include_nontrainable=False*)

tf_loss (*states*, *internals*, *reward*, *update*, *reference=None*)
Creates the TensorFlow operations for calculating the L2 loss between predicted state values and actual rewards.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor

tf_predict (*states, internals, update*)

tf_reference (*states, internals, reward, update*)

Creates the TensorFlow operations for obtaining the reference tensor(s), in case of a comparative loss.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Reference tensor(s).

tf_regularization_loss ()

tensorflow.core.distributions package**Submodules****tensorflow.core.distributions.bernoulli module**

```
class tensorflow.core.distributions.bernoulli.Bernoulli (shape, probability=0.5,  
                                                    scope='bernoulli',  
                                                    summary_labels=())
```

Bases: *tensorflow.core.distributions.distribution.Distribution*

Bernoulli distribution, for binary boolean actions.

```
__init__ (shape, probability=0.5, scope='bernoulli', summary_labels=())  
Bernoulli distribution.
```

Parameters

- **shape** – Action shape.
- **probability** – Optional distribution bias.

```
from_spec (spec, kwargs=None)
```

Creates a distribution from a specification dict.

```
get_summaries ()
```

```
get_variables (include_nontrainable=False)
```

```
state_action_value (distr_params, action=None)
```

```
state_value (distr_params)
```



```

tf_entropy (distr_params)
tf_kl_divergence (distr_params1, distr_params2)
tf_log_probability (distr_params, action)
tf_parameterize (x)
tf_regularization_loss ()
tf_sample (distr_params, deterministic)

```

tensorflow.core.distributions.beta module

```

class tensorflow.core.distributions.beta.Beta (shape, min_value, max_value, alpha=0.0, beta=0.0, scope='beta', summary_labels=())

```

Bases: *tensorflow.core.distributions.distribution.Distribution*

Beta distribution, for bounded continuous actions.

```

__init__ (shape, min_value, max_value, alpha=0.0, beta=0.0, scope='beta', summary_labels=())
    Beta distribution.

```

Parameters

- **shape** – Action shape.
- **min_value** – Minimum value of continuous actions.
- **max_value** – Maximum value of continuous actions.
- **alpha** – Optional distribution bias for the alpha value.
- **beta** – Optional distribution bias for the beta value.

```

from_spec (spec, kwargs=None)
    Creates a distribution from a specification dict.

```

```

get_summaries ()
get_variables (include_nontrainable=False)
tf_entropy (distr_params)
tf_kl_divergence (distr_params1, distr_params2)
tf_log_probability (distr_params, action)
tf_parameterize (x)
tf_regularization_loss ()
tf_sample (distr_params, deterministic)

```

tensorforce.core.distributions.categorical module

```
class tensorforce.core.distributions.categorical.Categorical (shape,  
num_actions, probabilities=None,  
scope='categorical',  
summary_labels=())
```

Bases: *tensorforce.core.distributions.distribution.Distribution*

Categorical distribution, for discrete actions.

```
__init__ (shape, num_actions, probabilities=None, scope='categorical', summary_labels=())  
    Categorical distribution.
```

Parameters

- **shape** – Action shape.
- **num_actions** – Number of discrete action alternatives.
- **probabilities** – Optional distribution bias.

```
from_spec (spec, kwargs=None)  
    Creates a distribution from a specification dict.  
get_summaries ()  
get_variables (include_nontrainable=False)  
state_action_value (distr_params, action=None)  
state_value (distr_params)  
tf_entropy (distr_params)  
tf_kl_divergence (distr_params1, distr_params2)  
tf_log_probability (distr_params, action)  
tf_parameterize (x)  
tf_regularization_loss ()  
tf_sample (distr_params, deterministic)
```

tensorforce.core.distributions.distribution module

```
class tensorforce.core.distributions.distribution.Distribution (shape,  
scope='distribution',  
summary_labels=None)
```

Bases: object

Base class for policy distributions.

```
__init__ (shape, scope='distribution', summary_labels=None)  
    Distribution.
```

Parameters **shape** – Action shape.

```
static from_spec (spec, kwargs=None)  
    Creates a distribution from a specification dict.
```

get_summaries ()

Returns the TensorFlow summaries reported by the distribution.

Returns List of summaries.

get_variables (*include_nontrainable=False*)

Returns the TensorFlow variables used by the distribution.

Returns List of variables.

tf_entropy (*distr_params*)

Creates the TensorFlow operations for calculating the entropy of a distribution.

Parameters **distr_params** – Tuple of distribution parameter tensors.

Returns Entropy tensor.

tf_kl_divergence (*distr_params1, distr_params2*)

Creates the TensorFlow operations for calculating the KL divergence between two distributions.

Parameters

- **distr_params1** – Tuple of parameter tensors for first distribution.
- **distr_params2** – Tuple of parameter tensors for second distribution.

Returns KL divergence tensor.

tf_log_probability (*distr_params, action*)

Creates the TensorFlow operations for calculating the log probability of an action for a distribution.

Parameters

- **distr_params** – Tuple of distribution parameter tensors.
- **action** – Action tensor.

Returns KL divergence tensor.

tf_parameterize (*x*)

Creates the TensorFlow operations for parameterizing a distribution conditioned on the given input.

Parameters **x** – Input tensor which the distribution is conditioned on.

Returns Tuple of distribution parameter tensors.

tf_regularization_loss ()

Creates the TensorFlow operations for the distribution regularization loss.

Returns Regularization loss tensor.

tf_sample (*distr_params, deterministic*)

Creates the TensorFlow operations for sampling an action based on a distribution.

Parameters

- **distr_params** – Tuple of distribution parameter tensors.
- **deterministic** – Boolean input tensor indicating whether the maximum likelihood action should be returned.

Returns Sampled action tensor.

tensorflow.core.distributions.gaussian module

```
class tensorflow.core.distributions.gaussian.Gaussian (shape, mean=0.0,  
                                                    log_stddev=0.0,  
                                                    scope='gaussian', summary_labels=())
```

Bases: *tensorflow.core.distributions.distribution.Distribution*

Gaussian distribution, for unbounded continuous actions.

```
__init__ (shape, mean=0.0, log_stddev=0.0, scope='gaussian', summary_labels=())  
    Categorical distribution.
```

Parameters

- **shape** – Action shape.
- **mean** – Optional distribution bias for the mean.
- **log_stddev** – Optional distribution bias for the standard deviation.

```
from_spec (spec, kwargs=None)  
    Creates a distribution from a specification dict.
```

```
get_summaries ()
```

```
get_variables (include_nontrainable=False)
```

```
state_action_value (distr_params, action)
```

```
state_value (distr_params)
```

```
tf_entropy (distr_params)
```

```
tf_kl_divergence (distr_params1, distr_params2)
```

```
tf_log_probability (distr_params, action)
```

```
tf_parameterize (x)
```

```
tf_regularization_loss ()
```

```
tf_sample (distr_params, deterministic)
```

Module contents

```
class tensorflow.core.distributions.Distribution (shape, scope='distribution', summary_labels=None)
```

Bases: *object*

Base class for policy distributions.

```
__init__ (shape, scope='distribution', summary_labels=None)  
    Distribution.
```

Parameters **shape** – Action shape.

```
static from_spec (spec, kwargs=None)  
    Creates a distribution from a specification dict.
```

```
get_summaries ()  
    Returns the TensorFlow summaries reported by the distribution.
```

Returns List of summaries.

get_variables (*include_nontrainable=False*)

Returns the TensorFlow variables used by the distribution.

Returns List of variables.

tf_entropy (*distr_params*)

Creates the TensorFlow operations for calculating the entropy of a distribution.

Parameters **distr_params** – Tuple of distribution parameter tensors.

Returns Entropy tensor.

tf_kl_divergence (*distr_params1, distr_params2*)

Creates the TensorFlow operations for calculating the KL divergence between two distributions.

Parameters

- **distr_params1** – Tuple of parameter tensors for first distribution.
- **distr_params2** – Tuple of parameter tensors for second distribution.

Returns KL divergence tensor.

tf_log_probability (*distr_params, action*)

Creates the TensorFlow operations for calculating the log probability of an action for a distribution.

Parameters

- **distr_params** – Tuple of distribution parameter tensors.
- **action** – Action tensor.

Returns KL divergence tensor.

tf_parameterize (*x*)

Creates the TensorFlow operations for parameterizing a distribution conditioned on the given input.

Parameters **x** – Input tensor which the distribution is conditioned on.

Returns Tuple of distribution parameter tensors.

tf_regularization_loss ()

Creates the TensorFlow operations for the distribution regularization loss.

Returns Regularization loss tensor.

tf_sample (*distr_params, deterministic*)

Creates the TensorFlow operations for sampling an action based on a distribution.

Parameters

- **distr_params** – Tuple of distribution parameter tensors.
- **deterministic** – Boolean input tensor indicating whether the maximum likelihood action should be returned.

Returns Sampled action tensor.

```
class tensorflow.core.distributions.Bernoulli (shape, probability=0.5,  
                                             scope='bernoulli', summary_labels=())
```

Bases: `tensorflow.core.distributions.distribution.Distribution`

Bernoulli distribution, for binary boolean actions.

```
__init__ (shape, probability=0.5, scope='bernoulli', summary_labels=())
```

Bernoulli distribution.

Parameters

- **shape** – Action shape.
- **probability** – Optional distribution bias.

from_spec (*spec*, *kwargs=None*)

Creates a distribution from a specification dict.

get_summaries ()

get_variables (*include_nontrainable=False*)

state_action_value (*distr_params*, *action=None*)

state_value (*distr_params*)

tf_entropy (*distr_params*)

tf_kl_divergence (*distr_params1*, *distr_params2*)

tf_log_probability (*distr_params*, *action*)

tf_parameterize (*x*)

tf_regularization_loss ()

tf_sample (*distr_params*, *deterministic*)

```
class tensorflow.core.distributions.Categorical (shape, num_actions, probabilities=None, scope='categorical', summary_labels=())
```

Bases: [tensorflow.core.distributions.distribution.Distribution](#)

Categorical distribution, for discrete actions.

__init__ (*shape*, *num_actions*, *probabilities=None*, *scope='categorical'*, *summary_labels=()*)

Categorical distribution.

Parameters

- **shape** – Action shape.
- **num_actions** – Number of discrete action alternatives.
- **probabilities** – Optional distribution bias.

from_spec (*spec*, *kwargs=None*)

Creates a distribution from a specification dict.

get_summaries ()

get_variables (*include_nontrainable=False*)

state_action_value (*distr_params*, *action=None*)

state_value (*distr_params*)

tf_entropy (*distr_params*)

tf_kl_divergence (*distr_params1*, *distr_params2*)

tf_log_probability (*distr_params*, *action*)

tf_parameterize (*x*)

tf_regularization_loss ()

tf_sample (*distr_params*, *deterministic*)

```
class tensorflowforce.core.distributions.Gaussian(shape, mean=0.0, log_stddev=0.0,
                                              scope='gaussian', summary_labels=())
    Bases: tensorflowforce.core.distributions.distribution.Distribution
```

Gaussian distribution, for unbounded continuous actions.

```
__init__ (shape, mean=0.0, log_stddev=0.0, scope='gaussian', summary_labels=())
    Categorical distribution.
```

Parameters

- **shape** – Action shape.
- **mean** – Optional distribution bias for the mean.
- **log_stddev** – Optional distribution bias for the standard deviation.

```
from_spec (spec, kwargs=None)
    Creates a distribution from a specification dict.
```

```
get_summaries ()
```

```
get_variables (include_nontrainable=False)
```

```
state_action_value (distr_params, action)
```

```
state_value (distr_params)
```

```
tf_entropy (distr_params)
```

```
tf_kl_divergence (distr_params1, distr_params2)
```

```
tf_log_probability (distr_params, action)
```

```
tf_parameterize (x)
```

```
tf_regularization_loss ()
```

```
tf_sample (distr_params, deterministic)
```

```
class tensorflowforce.core.distributions.Beta(shape, min_value, max_value, alpha=0.0,
                                              beta=0.0, scope='beta', summary_labels=())
    Bases: tensorflowforce.core.distributions.distribution.Distribution
```

Beta distribution, for bounded continuous actions.

```
__init__ (shape, min_value, max_value, alpha=0.0, beta=0.0, scope='beta', summary_labels=())
    Beta distribution.
```

Parameters

- **shape** – Action shape.
- **min_value** – Minimum value of continuous actions.
- **max_value** – Maximum value of continuous actions.
- **alpha** – Optional distribution bias for the alpha value.
- **beta** – Optional distribution bias for the beta value.

```
from_spec (spec, kwargs=None)
    Creates a distribution from a specification dict.
```

```
get_summaries ()
```

```
get_variables (include_nontrainable=False)
```

```
tf_entropy (distr_params)
```

```
tf_kl_divergence (distr_params1, distr_params2)
tf_log_probability (distr_params, action)
tf_parameterize (x)
tf_regularization_loss ()
tf_sample (distr_params, deterministic)
```

tensorflow.core.explorations package

Submodules

tensorflow.core.explorations.constant module

```
class tensorflow.core.explorations.constant.Constant (constant=0.0,
                                                    scope='constant',      sum-
                                                    mary_labels=())
Bases: tensorflow.core.explorations.exploration.Exploration
Explore via adding a constant term.
__init__ (constant=0.0, scope='constant', summary_labels=())
from_spec (spec)
    Creates an exploration object from a specification dict.
get_variables ()
    Returns exploration variables.
    Returns List of variables.
tf_explore (episode, timestep, action_spec=None)
```

tensorflow.core.explorations.epsilon_anneal module

```
class tensorflow.core.explorations.epsilon_anneal.EpsilonAnneal (initial_epsilon=1.0,
                                                                fi-
                                                                nal_epsilon=0.1,
                                                                timesteps=10000,
                                                                start_timestep=0,
                                                                scope='epsilon_anneal',
                                                                sum-
                                                                mary_labels=())
Bases: tensorflow.core.explorations.exploration.Exploration
Annealing epsilon parameter based on ratio of current timestep to total timesteps.
__init__ (initial_epsilon=1.0,      final_epsilon=0.1,      timesteps=10000,      start_timestep=0,
          scope='epsilon_anneal', summary_labels=())
from_spec (spec)
    Creates an exploration object from a specification dict.
get_variables ()
    Returns exploration variables.
    Returns List of variables.
```


tf_explore (*episode*, *timestep*, *action_spec=None*)

tensorflow.core.explorations.epsilon_decay module

class tensorflow.core.explorations.epsilon_decay.**EpsilonDecay** (*initial_epsilon=1.0*,
final_epsilon=0.1,
timesteps=10000,
start_timestep=0,
half_lives=10,
scope='epsilon_anneal',
summary_labels=())

Bases: *tensorflow.core.explorations.exploration.Exploration*

Exponentially decaying epsilon parameter based on ratio of difference between current and final epsilon to total timesteps.

__init__ (*initial_epsilon=1.0*, *final_epsilon=0.1*, *timesteps=10000*, *start_timestep=0*, *half_lives=10*,
scope='epsilon_anneal', *summary_labels=()*)

from_spec (*spec*)
 Creates an exploration object from a specification dict.

get_variables ()
 Returns exploration variables.

Returns List of variables.

tf_explore (*episode=0*, *timestep=0*, *action_spec=None*)

tensorflow.core.explorations.exploration module

class tensorflow.core.explorations.exploration.**Exploration** (*scope='exploration'*,
summary_labels=None)

Bases: object

Abstract exploration object.

__init__ (*scope='exploration'*, *summary_labels=None*)

static from_spec (*spec*)
 Creates an exploration object from a specification dict.

get_variables ()
 Returns exploration variables.

Returns List of variables.

tf_explore (*episode*, *timestep*, *action_spec*)
 Creates exploration value, e.g. compute an epsilon for epsilon-greedy or sample normal noise.

tensorforce.core.explorations.linear_decay module**tensorforce.core.explorations.ornstein_uhlenbeck_process module****class** tensorforce.core.explorations.ornstein_uhlenbeck_process.**OrnsteinUhlenbeckProcess** (*sign-**mu-*
the-
scop-
sum-
*man-*Bases: *tensorforce.core.explorations.exploration.Exploration*

Explores via an Ornstein-Uhlenbeck process.

__init__ (*sigma=0.3, mu=0.0, theta=0.15, scope='ornstein_uhlenbeck', summary_labels=()*)
Initializes an Ornstein-Uhlenbeck process which is a mean reverting stochastic process introducing time-correlated noise.**from_spec** (*spec*)
Creates an exploration object from a specification dict.**get_variables** ()
Returns exploration variables.**Returns** List of variables.**tf_explore** (*episode, timestep, action_spec*)**Module contents****class** tensorforce.core.explorations.**Exploration** (*scope='exploration', summary_labels=None*)

Bases: object

Abstract exploration object.

__init__ (*scope='exploration', summary_labels=None*)**static from_spec** (*spec*)
Creates an exploration object from a specification dict.**get_variables** ()
Returns exploration variables.**Returns** List of variables.**tf_explore** (*episode, timestep, action_spec*)
Creates exploration value, e.g. compute an epsilon for epsilon-greedy or sample normal noise.**class** tensorforce.core.explorations.**Constant** (*constant=0.0, scope='constant', summary_labels=()*)Bases: *tensorforce.core.explorations.exploration.Exploration*

Explore via adding a constant term.

__init__ (*constant=0.0, scope='constant', summary_labels=()*)**from_spec** (*spec*)
Creates an exploration object from a specification dict.

get_variables()

Returns exploration variables.

Returns List of variables.

tf_explore (*episode, timestep, action_spec=None*)

```
class tensorflow.core.explorations.EpsilonAnneal (initial_epsilon=1.0, final_epsilon=0.1, timesteps=10000, start_timestep=0, scope='epsilon_anneal', summary_labels=())
```

Bases: [tensorflow.core.explorations.exploration.Exploration](#)

Annealing epsilon parameter based on ratio of current timestep to total timesteps.

```
__init__ (initial_epsilon=1.0, final_epsilon=0.1, timesteps=10000, start_timestep=0, scope='epsilon_anneal', summary_labels=())
```

from_spec (*spec*)

Creates an exploration object from a specification dict.

get_variables()

Returns exploration variables.

Returns List of variables.

tf_explore (*episode, timestep, action_spec=None*)

```
class tensorflow.core.explorations.EpsilonDecay (initial_epsilon=1.0, final_epsilon=0.1, timesteps=10000, start_timestep=0, half_lives=10, scope='epsilon_anneal', summary_labels=())
```

Bases: [tensorflow.core.explorations.exploration.Exploration](#)

Exponentially decaying epsilon parameter based on ratio of difference between current and final epsilon to total timesteps.

```
__init__ (initial_epsilon=1.0, final_epsilon=0.1, timesteps=10000, start_timestep=0, half_lives=10, scope='epsilon_anneal', summary_labels=())
```

from_spec (*spec*)

Creates an exploration object from a specification dict.

get_variables()

Returns exploration variables.

Returns List of variables.

tf_explore (*episode=0, timestep=0, action_spec=None*)

```
class tensorflow.core.explorations.GaussianNoise (sigma=0.3, mu=0.0, scope='gaussian_noise', summary_labels=())
```

Bases: [tensorflow.core.explorations.exploration.Exploration](#)

Explores via gaussian noise.

```
__init__ (sigma=0.3, mu=0.0, scope='gaussian_noise', summary_labels=())
```

Initializes distribution values for gaussian noise

from_spec (*spec*)

Creates an exploration object from a specification dict.

get_variables()

Returns exploration variables.

Returns List of variables.

tf_explore(*episode, timestep, action_spec*)

```
class tensorflow.core.explorations.OrnsteinUhlenbeckProcess(sigma=0.3,  
                                                         mu=0.0,  
                                                         theta=0.15,  
                                                         scope='ornstein_uhlenbeck',  
                                                         summary_labels=())
```

Bases: `tensorflow.core.explorations.exploration.Exploration`

Explores via an Ornstein-Uhlenbeck process.

__init__(*sigma=0.3, mu=0.0, theta=0.15, scope='ornstein_uhlenbeck', summary_labels=()*)

Initializes an Ornstein-Uhlenbeck process which is a mean reverting stochastic process introducing time-correlated noise.

from_spec(*spec*)

Creates an exploration object from a specification dict.

get_variables()

Returns exploration variables.

Returns List of variables.

tf_explore(*episode, timestep, action_spec*)

tensorflow.core.memories package

Submodules

tensorflow.core.memories.memory module

```
class tensorflow.core.memories.memory.Memory(states, internals, actions, include_next_states,  
                                              scope='memory', summary_labels=None)
```

Bases: `object`

Base class for memories.

__init__(*states, internals, actions, include_next_states, scope='memory', summary_labels=None*)

Memory.

Parameters

- **states** – States specification.
- **internals** – Internal states specification.
- **actions** – Actions specification.
- **include_next_states** – Include subsequent state if true.

static from_spec(*spec, kwargs=None*)

Creates a memory from a specification dict.

get_summaries()

Returns the TensorFlow summaries reported by the memory.

Returns List of summaries.

get_variables()

Returns the TensorFlow variables used by the memory.

Returns List of variables.

tf_initialize()

Initializes memory.

tf_retrieve_episodes(*n*)

Retrieves a given number of episodes from the stored experiences.

Parameters *n* – Number of episodes to retrieve.

Returns Dicts containing the retrieved experiences.

tf_retrieve_sequences(*n*, *sequence_length*)

Retrieves a given number of temporally consistent timestep sequences from the stored experiences.

Parameters

- *n* – Number of sequences to retrieve.
- *sequence_length* – Length of timestep sequences.

Returns Dicts containing the retrieved experiences.

tf_retrieve_timesteps(*n*)

Retrieves a given number of timesteps from the stored experiences.

Parameters *n* – Number of timesteps to retrieve.

Returns Dicts containing the retrieved experiences.

tf_store(*states*, *internals*, *actions*, *terminal*, *reward*)

” Stores experiences, i.e. a batch of timesteps.

Parameters

- *states* – Dict of state tensors.
- *internals* – List of prior internal state tensors.
- *actions* – Dict of action tensors.
- *terminal* – Terminal boolean tensor.
- *reward* – Reward tensor.

tf_update_batch(*loss_per_instance*)

Updates the internal information of the latest batch instances based on their loss.

Parameters *loss_per_instance* – Loss per instance tensor.

tensorforce.core.memories.naive_prioritized_replay module**tensorforce.core.memories.prioritized_replay module**

```
class tensorforce.core.memories.prioritized_replay.PrioritizedReplay(states,  
                                                                    inter-  
                                                                    nals,  
                                                                    ac-  
                                                                    tions,  
                                                                    in-  
                                                                    clude_next_states,  
                                                                    capac-  
                                                                    ity,  
                                                                    prior-  
                                                                    itiza-  
                                                                    tion_weight=1.0,  
                                                                    buffer_size=100,  
                                                                    scope='queue',  
                                                                    sum-  
                                                                    mary_labels=None)
```

Bases: `tensorforce.core.memories.memory.Memory`

Memory organized as a priority queue, which randomly retrieves experiences sampled according their priority values.

```
__init__(states, internals, actions, include_next_states, capacity, prioritization_weight=1.0,  
         buffer_size=100, scope='queue', summary_labels=None)  
    Prioritized experience replay.
```

Parameters

- **states** – States specification.
- **internals** – Internal states specification.
- **actions** – Actions specification.
- **include_next_states** – Include subsequent state if true.
- **capacity** – Memory capacity.
- **prioritization_weight** – Prioritization weight.
- **buffer_size** – Buffer size. The buffer is used to insert experiences before experiences have been computed via updates.

```
from_spec(spec, kwargs=None)  
    Creates a memory from a specification dict.
```

```
get_summaries()  
    Returns the TensorFlow summaries reported by the memory.
```

Returns List of summaries.

```
get_variables()  
    Returns the TensorFlow variables used by the memory.
```

Returns List of variables.

```
tf_initialize()
```

```
tf_retrieve_episodes(n)
```

tf_retrieve_indices (*buffer_elements, priority_indices*)

Fetches experiences for given indices by combining entries from buffer which have no priorities, and entries from priority memory.

Parameters

- **buffer_elements** – Number of buffer elements to retrieve
- **priority_indices** – Index tensor for priority memory

Returns: Batch of experiences

tf_retrieve_sequences (*n, sequence_length*)

tf_retrieve_timesteps (*n*)

tf_store (*states, internals, actions, terminal, reward*)

tf_update_batch (*loss_per_instance*)

Updates priority memory by performing the following steps:

1. Use saved indices from prior retrieval to reconstruct the batch elements which will have their priorities updated.
2. Compute priorities for these elements.
3. Insert buffer elements to memory, potentially overwriting existing elements.
4. Update priorities of existing memory elements
5. Resort memory.
6. Update buffer insertion index.

Note that this implementation could be made more efficient by maintaining a sorted version via sum trees.

Parameters **loss_per_instance** – Losses from recent batch to perform priority update

tensorforce.core.memories.replay module

class tensorforce.core.memories.replay.**Replay** (*states, internals, actions, include_next_states, capacity, scope='replay', summary_labels=None*)

Bases: tensorforce.core.memories.queue.Queue

Memory which randomly retrieves experiences.

__init__ (*states, internals, actions, include_next_states, capacity, scope='replay', summary_labels=None*)

Replay memory.

Parameters

- **states** – States specification.
- **internals** – Internal states specification.
- **actions** – Actions specification.
- **include_next_states** – Include subsequent state if true.
- **capacity** – Memory capacity.

from_spec (*spec, kwargs=None*)

Creates a memory from a specification dict.

get_summaries()
Returns the TensorFlow summaries reported by the memory.
Returns List of summaries.

get_variables()
Returns the TensorFlow variables used by the memory.
Returns List of variables.

tf_initialize()

tf_retrieve_episodes(n)

tf_retrieve_indices(indices)
Fetches experiences for given indices.
Parameters **indices** – Index tensor
Returns: Batch of experiences

tf_retrieve_sequences(n, sequence_length)

tf_retrieve_timesteps(n)

tf_store(states, internals, actions, terminal, reward)

tf_update_batch(loss_per_instance)
Updates the internal information of the latest batch instances based on their loss.
Parameters **loss_per_instance** – Loss per instance tensor.

Module contents

class `tensorforce.core.memories.Memory`(*states, internals, actions, include_next_states, scope='memory', summary_labels=None*)

Bases: `object`

Base class for memories.

__init__(*states, internals, actions, include_next_states, scope='memory', summary_labels=None*)
Memory.

Parameters

- **states** – States specification.
- **internals** – Internal states specification.
- **actions** – Actions specification.
- **include_next_states** – Include subsequent state if true.

static from_spec(*spec, kwargs=None*)
Creates a memory from a specification dict.

get_summaries()
Returns the TensorFlow summaries reported by the memory.
Returns List of summaries.

get_variables()
Returns the TensorFlow variables used by the memory.
Returns List of variables.

tf_initialize()

Initializes memory.

tf_retrieve_episodes(*n*)

Retrieves a given number of episodes from the stored experiences.

Parameters *n* – Number of episodes to retrieve.

Returns Dicts containing the retrieved experiences.

tf_retrieve_sequences(*n*, *sequence_length*)

Retrieves a given number of temporally consistent timestep sequences from the stored experiences.

Parameters

- *n* – Number of sequences to retrieve.
- *sequence_length* – Length of timestep sequences.

Returns Dicts containing the retrieved experiences.

tf_retrieve_timesteps(*n*)

Retrieves a given number of timesteps from the stored experiences.

Parameters *n* – Number of timesteps to retrieve.

Returns Dicts containing the retrieved experiences.

tf_store(*states*, *internals*, *actions*, *terminal*, *reward*)

” Stores experiences, i.e. a batch of timesteps.

Parameters

- *states* – Dict of state tensors.
- *internals* – List of prior internal state tensors.
- *actions* – Dict of action tensors.
- *terminal* – Terminal boolean tensor.
- *reward* – Reward tensor.

tf_update_batch(*loss_per_instance*)

Updates the internal information of the latest batch instances based on their loss.

Parameters *loss_per_instance* – Loss per instance tensor.

class `tensorforce.core.memories.Queue`(*states*, *internals*, *actions*, *include_next_states*, *capacity*, *scope*=*'queue'*, *summary_labels*=None)

Bases: `tensorforce.core.memories.memory.Memory`

Base class for memories organized as a queue (FIFO).

__init__(*states*, *internals*, *actions*, *include_next_states*, *capacity*, *scope*=*'queue'*, *summary_labels*=None)

Queue memory.

Parameters

- *states* – States specification.
- *internals* – Internal states specification.
- *actions* – Actions specification.
- *include_next_states* – Include subsequent state if true.
- *capacity* – Memory capacity.

from_spec (*spec*, *kwargs=None*)

Creates a memory from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the memory.

Returns List of summaries.

get_variables ()

Returns the TensorFlow variables used by the memory.

Returns List of variables.

tf_initialize ()

tf_retrieve_episodes (*n*)

Retrieves a given number of episodes from the stored experiences.

Parameters *n* – Number of episodes to retrieve.

Returns Dicts containing the retrieved experiences.

tf_retrieve_indices (*indices*)

Fetches experiences for given indices.

Parameters *indices* – Index tensor

Returns: Batch of experiences

tf_retrieve_sequences (*n*, *sequence_length*)

Retrieves a given number of temporally consistent timestep sequences from the stored experiences.

Parameters

- *n* – Number of sequences to retrieve.
- *sequence_length* – Length of timestep sequences.

Returns Dicts containing the retrieved experiences.

tf_retrieve_timesteps (*n*)

Retrieves a given number of timesteps from the stored experiences.

Parameters *n* – Number of timesteps to retrieve.

Returns Dicts containing the retrieved experiences.

tf_store (*states*, *internals*, *actions*, *terminal*, *reward*)

tf_update_batch (*loss_per_instance*)

Updates the internal information of the latest batch instances based on their loss.

Parameters *loss_per_instance* – Loss per instance tensor.

class tensorforce.core.memories.**Latest** (*states*, *internals*, *actions*, *include_next_states*, *capacity*, *scope='latest'*, *summary_labels=None*)

Bases: tensorforce.core.memories.queue.Queue

Memory which always retrieves most recent experiences.

__init__ (*states*, *internals*, *actions*, *include_next_states*, *capacity*, *scope='latest'*, *summary_labels=None*)

Latest memory.

Parameters

- *states* – States specification.

- **internals** – Internal states specification.
- **actions** – Actions specification.
- **include_next_states** – Include subsequent state if true.
- **capacity** – Memory capacity.

from_spec (*spec*, *kwargs=None*)

Creates a memory from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the memory.

Returns List of summaries.

get_variables ()

Returns the TensorFlow variables used by the memory.

Returns List of variables.

tf_initialize ()

tf_retrieve_episodes (*n*)

tf_retrieve_indices (*indices*)

Fetches experiences for given indices.

Parameters **indices** – Index tensor

Returns: Batch of experiences

tf_retrieve_sequences (*n*, *sequence_length*)

tf_retrieve_timesteps (*n*)

tf_store (*states*, *internals*, *actions*, *terminal*, *reward*)

tf_update_batch (*loss_per_instance*)

Updates the internal information of the latest batch instances based on their loss.

Parameters **loss_per_instance** – Loss per instance tensor.

class `tensorforce.core.memories.Replay` (*states*, *internals*, *actions*, *include_next_states*, *capacity*, *scope='replay'*, *summary_labels=None*)

Bases: `tensorforce.core.memories.queue.Queue`

Memory which randomly retrieves experiences.

__init__ (*states*, *internals*, *actions*, *include_next_states*, *capacity*, *scope='replay'*, *summary_labels=None*)

Replay memory.

Parameters

- **states** – States specification.
- **internals** – Internal states specification.
- **actions** – Actions specification.
- **include_next_states** – Include subsequent state if true.
- **capacity** – Memory capacity.

from_spec (*spec*, *kwargs=None*)

Creates a memory from a specification dict.

get_summaries()

Returns the TensorFlow summaries reported by the memory.

Returns List of summaries.

get_variables()

Returns the TensorFlow variables used by the memory.

Returns List of variables.

tf_initialize()

tf_retrieve_episodes(*n*)

tf_retrieve_indices(*indices*)

Fetches experiences for given indices.

Parameters *indices* – Index tensor

Returns: Batch of experiences

tf_retrieve_sequences(*n*, *sequence_length*)

tf_retrieve_timesteps(*n*)

tf_store(*states*, *internals*, *actions*, *terminal*, *reward*)

tf_update_batch(*loss_per_instance*)

Updates the internal information of the latest batch instances based on their loss.

Parameters *loss_per_instance* – Loss per instance tensor.

```
class tensorflow.core.memories.PrioritizedReplay(states, internals, actions, include_next_states, capacity,  
                                              prioritization_weight=1.0,  
                                              buffer_size=100, scope='queue',  
                                              summary_labels=None)
```

Bases: [*tensorflow.core.memories.memory.Memory*](#)

Memory organized as a priority queue, which randomly retrieves experiences sampled according their priority values.

```
__init__(states, internals, actions, include_next_states, capacity, prioritization_weight=1.0,  
        buffer_size=100, scope='queue', summary_labels=None)
```

Prioritized experience replay.

Parameters

- **states** – States specification.
- **internals** – Internal states specification.
- **actions** – Actions specification.
- **include_next_states** – Include subsequent state if true.
- **capacity** – Memory capacity.
- **prioritization_weight** – Prioritization weight.
- **buffer_size** – Buffer size. The buffer is used to insert experiences before experiences have been computed via updates.

```
from_spec(spec, kwargs=None)
```

Creates a memory from a specification dict.

get_summaries()

Returns the TensorFlow summaries reported by the memory.

Returns List of summaries.

get_variables()

Returns the TensorFlow variables used by the memory.

Returns List of variables.

tf_initialize()

tf_retrieve_episodes (*n*)

tf_retrieve_indices (*buffer_elements*, *priority_indices*)

Fetches experiences for given indices by combining entries from buffer which have no priorities, and entries from priority memory.

Parameters

- **buffer_elements** – Number of buffer elements to retrieve
- **priority_indices** – Index tensor for priority memory

Returns: Batch of experiences

tf_retrieve_sequences (*n*, *sequence_length*)

tf_retrieve_timesteps (*n*)

tf_store (*states*, *internals*, *actions*, *terminal*, *reward*)

tf_update_batch (*loss_per_instance*)

Updates priority memory by performing the following steps:

1. Use saved indices from prior retrieval to reconstruct the batch elements which will have their priorities updated.
2. Compute priorities for these elements.
3. Insert buffer elements to memory, potentially overwriting existing elements.
4. Update priorities of existing memory elements
5. Resort memory.
6. Update buffer insertion index.

Note that this implementation could be made more efficient by maintaining a sorted version via sum trees.

Parameters **loss_per_instance** – Losses from recent batch to perform priority update

tensorflow.core.networks package

Submodules

tensorflow.core.networks.complex_network module

```
class tensorflow.core.networks.complex_network.ComplexLayeredNetwork (complex_layers_spec,  
                                                                    scope='layered-  
                                                                    network',  
                                                                    sum-  
                                                                    mary_labels=())
```

Bases: *tensorflow.core.networks.network.LayerBasedNetwork*

Complex Network consisting of a sequence of layers, which can be created from a specification dict.

```
__init__ (complex_layers_spec, scope='layered-network', summary_labels=())  
    Complex Layered network.
```

Parameters **complex_layers_spec** – List of layer specification dicts

```
add_layer (layer)
```

```
static from_json (filename)  
    Creates a complex_layered_network_builder from a JSON.
```

Parameters **filename** – Path to configuration

Returns: A ComplexLayeredNetwork class with layers generated from the JSON

```
from_spec (spec, kwargs=None)  
    Creates a network from a specification dict.
```

```
get_list_of_named_tensor ()  
    Returns a list of the names of tensors available.
```

Returns List of the names of tensors available.

```
get_named_tensor (name)  
    Returns a named tensor if available.
```

Returns True if named tensor found, False otherwise tensor: If valid, will be a tensor, otherwise None

Return type valid

```
get_summaries ()
```

```
get_variables (include_nontrainable=False)
```

```
internals_spec ()
```

```
set_named_tensor (name, tensor)  
    Returns the TensorFlow summaries reported by the network.
```

Returns None

```
tf_apply (x, internals, update, return_internals=False)
```

```
tf_regularization_loss ()
```

```
class tensorflow.core.networks.complex_network.Input (inputs, axis=1,  
                                                         scope='merge_inputs',  
                                                         summary_labels=())
```

Bases: *tensorflow.core.networks.layer.Layer*

Input layer. Used for ComplexLayerNetwork's to collect data together as a form of output to the next layer. Allows for multiple inputs to merge into a single import for next layer.

`__init__` (*inputs*, *axis*=1, *scope*='merge_inputs', *summary_labels*=())

Input layer.

Parameters

- **inputs** – A list of strings that name the inputs to merge
- **axis** – Axis to merge the inputs

`from_spec` (*spec*, *kwargs*=None)

Creates a layer from a specification dict.

`get_summaries` ()

Returns the TensorFlow summaries reported by the layer.

Returns List of summaries.

`get_variables` (*include_nontrainable*=False)

Returns the TensorFlow variables used by the layer.

Returns List of variables.

`internals_spec` ()

Returns the internal states specification.

Returns Internal states specification

`tf_apply` (*x*, *update*)

`tf_regularization_loss` ()

Creates the TensorFlow operations for the layer regularization loss.

Returns Regularization loss tensor.

`tf_tensors` (*named_tensors*)

Attaches the named_tensors dictionary to the layer for examination and update.

Parameters **named_tensors** – Dictionary of named tensors to be used as Input's or recorded from outputs

Returns NA

class `tensorforce.core.networks.complex_network.Output` (*output*, *scope*='output', *summary_labels*=())

Bases: `tensorforce.core.networks.layer.Layer`

Output layer. Used for ComplexLayerNetwork's to capture the tensor under and name for use with Input layers. Acts as a input to output passthrough.

`__init__` (*output*, *scope*='output', *summary_labels*=())

Output layer.

Parameters **output** – A string that names the tensor, will be added to available inputs

`from_spec` (*spec*, *kwargs*=None)

Creates a layer from a specification dict.

`get_summaries` ()

Returns the TensorFlow summaries reported by the layer.

Returns List of summaries.

`get_variables` (*include_nontrainable*=False)

Returns the TensorFlow variables used by the layer.

Returns List of variables.

internals_spec()

Returns the internal states specification.

Returns Internal states specification

tf_apply(*x, update*)

tf_regularization_loss()

Creates the TensorFlow operations for the layer regularization loss.

Returns Regularization loss tensor.

tf_tensors(*named_tensors*)

Attaches the *named_tensors* dictionary to the layer for examination and update.

Parameters **named_tensors** – Dictionary of named tensors to be used as Input's or recorded from outputs

Returns NA

tensorforce.core.networks.layer module

Collection of custom layer implementations.

```
class tensorforce.core.networks.layer.Conv1d(size,           window=3,           stride=1,
                                           padding='SAME',  bias=True,   activation='relu',
                                           l2_regularization=0.0,
                                           l1_regularization=0.0,  scope='conv1d',
                                           summary_labels=())
```

Bases: [tensorforce.core.networks.layer.Layer](#)

1-dimensional convolutional layer.

```
__init__(size,   window=3,   stride=1,   padding='SAME',   bias=True,   activation='relu',
          l2_regularization=0.0, l1_regularization=0.0, scope='conv1d', summary_labels=())
```

1D convolutional layer.

Parameters

- **size** – Number of filters
- **window** – Convolution window size
- **stride** – Convolution stride
- **padding** – Convolution padding, one of 'VALID' or 'SAME'
- **bias** – If true, a bias is added
- **activation** – Type of nonlinearity, or dict with name & arguments
- **l2_regularization** – L2 regularization weight
- **l1_regularization** – L1 regularization weight

from_spec(*spec, kwargs=None*)

Creates a layer from a specification dict.

get_summaries()

get_variables(*include_nontrainable=False*)

internals_spec()

Returns the internal states specification.

Returns Internal states specification

tf_apply(*x, update*)

tf_regularization_loss()

tf_tensors(*named_tensors*)

Attaches the *named_tensors* dictionary to the layer for examination and update.

Parameters **named_tensors** – Dictionary of named tensors to be used as Input's or recorded from outputs

Returns NA

```
class tensorflow.core.networks.layer.Conv2d(size,           window=3,           stride=1,
                                           padding='SAME', bias=True,  activation='relu',
                                           l2_regularization=0.0,
                                           l1_regularization=0.0, scope='conv2d',
                                           summary_labels=())
```

Bases: [tensorflow.core.networks.layer.Layer](#)

2-dimensional convolutional layer.

```
__init__(size,    window=3,    stride=1,    padding='SAME',    bias=True,    activation='relu',
          l2_regularization=0.0, l1_regularization=0.0, scope='conv2d', summary_labels=())
```

2D convolutional layer.

Parameters

- **size** – Number of filters
- **window** – Convolution window size, either an integer or pair of integers.
- **stride** – Convolution stride, either an integer or pair of integers.
- **padding** – Convolution padding, one of 'VALID' or 'SAME'
- **bias** – If true, a bias is added
- **activation** – Type of nonlinearity, or dict with name & arguments
- **l2_regularization** – L2 regularization weight
- **l1_regularization** – L1 regularization weight

from_spec(*spec, kwargs=None*)

Creates a layer from a specification dict.

get_summaries()

get_variables(*include_nontrainable=False*)

internals_spec()

Returns the internal states specification.

Returns Internal states specification

tf_apply(*x, update*)

tf_regularization_loss()

tf_tensors(*named_tensors*)

Attaches the *named_tensors* dictionary to the layer for examination and update.

Parameters `named_tensors` – Dictionary of named tensors to be used as Input's or recorded from outputs

Returns NA

```
class tensorflowforce.core.networks.layer.Dense (size=None, weights=None, bias=True, activation='relu', l2_regularization=0.0, l1_regularization=0.0, skip=False, scope='dense', summary_labels=())
```

Bases: `tensorflowforce.core.networks.layer.Layer`

Dense layer, i.e. linear fully connected layer with subsequent non-linearity.

```
__init__ (size=None, weights=None, bias=True, activation='relu', l2_regularization=0.0, l1_regularization=0.0, skip=False, scope='dense', summary_labels=())
```

Dense layer.

Parameters

- **size** – Layer size, if None than input size matches the output size of the layer
- **weights** – Weight initialization, random if None.
- **bias** – If true, bias is added.
- **activation** – Type of nonlinearity, or dict with name & arguments
- **l2_regularization** – L2 regularization weight.
- **l1_regularization** – L1 regularization weight.
- **skip** – Add skip connection like ResNet (<https://arxiv.org/pdf/1512.03385.pdf>), doubles layers and ShortCut from Input to output

```
from_spec (spec, kwargs=None)
```

Creates a layer from a specification dict.

```
get_summaries ()
```

```
get_variables (include_nontrainable=False)
```

```
internals_spec ()
```

Returns the internal states specification.

Returns Internal states specification

```
tf_apply (x, update)
```

```
tf_regularization_loss ()
```

```
tf_tensors (named_tensors)
```

Attaches the `named_tensors` dictionary to the layer for examination and update.

Parameters `named_tensors` – Dictionary of named tensors to be used as Input's or recorded from outputs

Returns NA

```
class tensorflowforce.core.networks.layer.Dropout (rate=0.0, scope='dropout', summary_labels=())
```

Bases: `tensorflowforce.core.networks.layer.Layer`

Dropout layer. If using dropout, add this layer after inputs and after dense layers. For LSTM, dropout is handled independently as an argument. Not available for Conv2d yet.

```
__init__ (rate=0.0, scope='dropout', summary_labels=())
```

from_spec (*spec*, *kwargs=None*)
Creates a layer from a specification dict.

get_summaries ()
Returns the TensorFlow summaries reported by the layer.

Returns List of summaries.

get_variables (*include_nontrainable=False*)
Returns the TensorFlow variables used by the layer.

Returns List of variables.

internals_spec ()
Returns the internal states specification.

Returns Internal states specification

tf_apply (*x*, *update*)

tf_regularization_loss ()
Creates the TensorFlow operations for the layer regularization loss.

Returns Regularization loss tensor.

tf_tensors (*named_tensors*)
Attaches the *named_tensors* dictionary to the layer for examination and update.

Parameters **named_tensors** – Dictionary of named tensors to be used as Input's or recorded from outputs

Returns NA

class `tensorforce.core.networks.layer.Dueling` (*size*, *bias=False*, *activation='none'*, *l2_regularization=0.0*, *l1_regularization=0.0*, *output=None*, *scope='dueling'*, *summary_labels=()*)

Bases: `tensorforce.core.networks.layer.Layer`

Dueling layer, i.e. Duel pipelines for Exp & Adv to help with stability

__init__ (*size*, *bias=False*, *activation='none'*, *l2_regularization=0.0*, *l1_regularization=0.0*, *output=None*, *scope='dueling'*, *summary_labels=()*)
Dueling layer.

[Dueling Networks] (<https://arxiv.org/pdf/1511.06581.pdf>) Implement $Y = \text{Expectation}[x] + (\text{Advantage}[x] - \text{Mean}(\text{Advantage}[x]))$

Parameters

- **size** – Layer size.
- **bias** – If true, bias is added.
- **activation** – Type of nonlinearity, or dict with name & arguments
- **l2_regularization** – L2 regularization weight.
- **l1_regularization** – L1 regularization weight.
- **output** – None or tuple of output names for ('expectation', 'advantage', 'mean_advantage')

from_spec (*spec*, *kwargs=None*)
Creates a layer from a specification dict.

get_summaries ()

get_variables (*include_nontrainable=False*)

internals_spec ()

Returns the internal states specification.

Returns Internal states specification

tf_apply (*x, update*)

tf_regularization_loss ()

tf_tensors (*named_tensors*)

Attaches the named_tensors dictionary to the layer for examination and update.

Parameters **named_tensors** – Dictionary of named tensors to be used as Input's or recorded from outputs

Returns NA

class tensorflow.core.networks.layer.**Embedding** (*indices, size, l2_regularization=0.0, l1_regularization=0.0, scope='embedding', summary_labels=()*)

Bases: [tensorflow.core.networks.layer.Layer](#)

Embedding layer.

__init__ (*indices, size, l2_regularization=0.0, l1_regularization=0.0, scope='embedding', summary_labels=()*)

Embedding layer.

Parameters

- **indices** – Number of embedding indices.
- **size** – Embedding size.
- **l2_regularization** – L2 regularization weight.
- **l1_regularization** – L1 regularization weight.

from_spec (*spec, kwargs=None*)

Creates a layer from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the layer.

Returns List of summaries.

get_variables (*include_nontrainable=False*)

Returns the TensorFlow variables used by the layer.

Returns List of variables.

internals_spec ()

Returns the internal states specification.

Returns Internal states specification

tf_apply (*x, update*)

tf_regularization_loss ()

tf_tensors (*named_tensors*)

Attaches the named_tensors dictionary to the layer for examination and update.

Parameters `named_tensors` – Dictionary of named tensors to be used as Input's or recorded from outputs

Returns NA

class `tensorforce.core.networks.layer.Flatten` (*scope='flatten', summary_labels=()*)

Bases: `tensorforce.core.networks.layer.Layer`

Flatten layer reshaping the input.

__init__ (*scope='flatten', summary_labels=()*)

from_spec (*spec, kwargs=None*)

Creates a layer from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the layer.

Returns List of summaries.

get_variables (*include_nontrainable=False*)

Returns the TensorFlow variables used by the layer.

Returns List of variables.

internals_spec ()

Returns the internal states specification.

Returns Internal states specification

tf_apply (*x, update*)

tf_regularization_loss ()

Creates the TensorFlow operations for the layer regularization loss.

Returns Regularization loss tensor.

tf_tensors (*named_tensors*)

Attaches the `named_tensors` dictionary to the layer for examination and update.

Parameters `named_tensors` – Dictionary of named tensors to be used as Input's or recorded from outputs

Returns NA

class `tensorforce.core.networks.layer.InternalLstm` (*size, dropout=None, lstmcell_args={}, scope='internal_lstm', summary_labels=()*)

Bases: `tensorforce.core.networks.layer.Layer`

Long short-term memory layer for internal state management.

__init__ (*size, dropout=None, lstmcell_args={}, scope='internal_lstm', summary_labels=()*)

LSTM layer.

Parameters

- **size** – LSTM size.
- **dropout** – Dropout rate.

from_spec (*spec, kwargs=None*)

Creates a layer from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the layer.

Returns List of summaries.

get_variables (*include_nontrainable=False*)

Returns the TensorFlow variables used by the layer.

Returns List of variables.

internals_spec ()

tf_apply (*x, update, state*)

tf_regularization_loss ()

Creates the TensorFlow operations for the layer regularization loss.

Returns Regularization loss tensor.

tf_tensors (*named_tensors*)

Attaches the named_tensors dictionary to the layer for examination and update.

Parameters **named_tensors** – Dictionary of named tensors to be used as Input's or recorded from outputs

Returns NA

class tensorflow.core.networks.layer.**Layer** (*scope='layer', summary_labels=None*)

Bases: object

Base class for network layers.

__init__ (*scope='layer', summary_labels=None*)

Layer.

static from_spec (*spec, kwargs=None*)

Creates a layer from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the layer.

Returns List of summaries.

get_variables (*include_nontrainable=False*)

Returns the TensorFlow variables used by the layer.

Returns List of variables.

internals_spec ()

Returns the internal states specification.

Returns Internal states specification

tf_apply (*x, update*)

Creates the TensorFlow operations for applying the layer to the given input.

Parameters

- **x** – Layer input tensor.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Layer output tensor.

tf_regularization_loss ()

Creates the TensorFlow operations for the layer regularization loss.

Returns Regularization loss tensor.

tf_tensors (*named_tensors*)

Attaches the `named_tensors` dictionary to the layer for examination and update.

Parameters **named_tensors** – Dictionary of named tensors to be used as Input's or recorded from outputs

Returns NA

```
class tensorflow.core.networks.layer.Linear (size, weights=None, bias=True,
                                             l2_regularization=0.0,
                                             l1_regularization=0.0, scope='linear',
                                             summary_labels=())
```

Bases: `tensorflow.core.networks.layer.Layer`

Linear fully-connected layer.

```
__init__ (size, weights=None, bias=True, l2_regularization=0.0, l1_regularization=0.0,
          scope='linear', summary_labels=())
```

Linear layer.

Parameters

- **size** – Layer size.
- **weights** – Weight initialization, random if None.
- **bias** – Bias initialization, random if True, no bias added if False.
- **l2_regularization** – L2 regularization weight.
- **l1_regularization** – L1 regularization weight.

```
from_spec (spec, kwargs=None)
```

Creates a layer from a specification dict.

```
get_summaries ()
```

Returns the TensorFlow summaries reported by the layer.

Returns List of summaries.

```
get_variables (include_nontrainable=False)
```

Returns the TensorFlow variables used by the layer.

Returns List of variables.

```
internals_spec ()
```

Returns the internal states specification.

Returns Internal states specification

```
tf_apply (x, update=False)
```

```
tf_regularization_loss ()
```

```
tf_tensors (named_tensors)
```

Attaches the `named_tensors` dictionary to the layer for examination and update.

Parameters **named_tensors** – Dictionary of named tensors to be used as Input's or recorded from outputs

Returns NA

```
class tensorflow.core.networks.layer.Lstm (size, dropout=None, scope='lstm', summary_labels=(),
                                           return_final_state=True)
```

Bases: `tensorflow.core.networks.layer.Layer`

`__init__` (*size*, *dropout=None*, *scope='lstm'*, *summary_labels=()*, *return_final_state=True*)
LSTM layer.

Parameters

- **size** – LSTM size.
- **dropout** – Dropout rate.

`from_spec` (*spec*, *kwargs=None*)
Creates a layer from a specification dict.

`get_summaries` ()
Returns the TensorFlow summaries reported by the layer.

Returns List of summaries.

`get_variables` (*include_nontrainable=False*)
Returns the TensorFlow variables used by the layer.

Returns List of variables.

`internals_spec` ()
Returns the internal states specification.

Returns Internal states specification

`tf_apply` (*x*, *update*, *sequence_length=None*)

`tf_regularization_loss` ()
Creates the TensorFlow operations for the layer regularization loss.

Returns Regularization loss tensor.

`tf_tensors` (*named_tensors*)
Attaches the *named_tensors* dictionary to the layer for examination and update.

Parameters **named_tensors** – Dictionary of named tensors to be used as Input's or recorded from outputs

Returns NA

```
class tensorflow.core.networks.layer.Nonlinearity (name='relu',          alpha=None,
                                                  beta=1.0, max=None, min=None,
                                                  scope='nonlinearity',      sum-
                                                  mary_labels=())
```

Bases: `tensorflow.core.networks.layer.Layer`

Non-linearity layer applying a non-linear transformation.

`__init__` (*name='relu'*, *alpha=None*, *beta=1.0*, *max=None*, *min=None*, *scope='nonlinearity'*, *summary_labels=()*)
Non-linearity activation layer.

Parameters

- **name** – Non-linearity name, one of 'elu', 'relu', 'selu', 'sigmoid', 'swish', 'softmax', 'leaky_relu' (or 'lrelu'), 'crelu', 'softmax', 'softplus', 'softsign', 'tanh' or 'none'.
- **alpha** – (float) Alpha value for leaky Relu
- **beta** – (float) Beta value or 'learn' to train value (default 1.0)
- **max** – (float) maximum (beta * input) value passed to non-linearity function
- **min** – (float) minimum (beta * input) value passed to non-linearity function

- **summary_labels** – Requested summary labels for tensorboard export, add ‘beta’ to watch beta learning

from_spec (*spec*, *kwargs=None*)

Creates a layer from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the layer.

Returns List of summaries.

get_variables (*include_nontrainable=False*)

Returns the TensorFlow variables used by the layer.

Returns List of variables.

internals_spec ()

Returns the internal states specification.

Returns Internal states specification

tf_apply (*x*, *update*)

tf_regularization_loss ()

Creates the TensorFlow operations for the layer regularization loss.

Returns Regularization loss tensor.

tf_tensors (*named_tensors*)

Attaches the named_tensors dictionary to the layer for examination and update.

Parameters **named_tensors** – Dictionary of named tensors to be used as Input’s or recorded from outputs

Returns NA

class `tensorforce.core.networks.layer.Pool2d` (*pooling_type='max'*, *window=2*, *stride=2*, *padding='SAME'*, *scope='pool2d'*, *summary_labels=()*)

Bases: `tensorforce.core.networks.layer.Layer`

2-dimensional pooling layer.

__init__ (*pooling_type='max'*, *window=2*, *stride=2*, *padding='SAME'*, *scope='pool2d'*, *summary_labels=()*)

2-dimensional pooling layer.

Parameters

- **pooling_type** – Either ‘max’ or ‘average’.
- **window** – Pooling window size, either an integer or pair of integers.
- **stride** – Pooling stride, either an integer or pair of integers.
- **padding** – Pooling padding, one of ‘VALID’ or ‘SAME’.

from_spec (*spec*, *kwargs=None*)

Creates a layer from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the layer.

Returns List of summaries.

get_variables (*include_nontrainable=False*)

Returns the TensorFlow variables used by the layer.

Returns List of variables.

internals_spec ()

Returns the internal states specification.

Returns Internal states specification

tf_apply (*x*, *update*)

tf_regularization_loss ()

Creates the TensorFlow operations for the layer regularization loss.

Returns Regularization loss tensor.

tf_tensors (*named_tensors*)

Attaches the *named_tensors* dictionary to the layer for examination and update.

Parameters **named_tensors** – Dictionary of named tensors to be used as Input's or recorded from outputs

Returns NA

class `tensorforce.core.networks.layer.TFLayer` (*layer*, *scope*='tf-layer', *summary_labels*=(), ***kwargs*)

Bases: `tensorforce.core.networks.layer.Layer`

Wrapper class for TensorFlow layers.

__init__ (*layer*, *scope*='tf-layer', *summary_labels*=(), ***kwargs*)

Creates a new layer instance of a TensorFlow layer.

Parameters

- **name** – The name of the layer, one of 'dense'.
- ****kwargs** – Additional arguments passed on to the TensorFlow layer constructor.

from_spec (*spec*, *kwargs*=None)

Creates a layer from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the layer.

Returns List of summaries.

get_variables (*include_nontrainable*=False)

Returns the TensorFlow variables used by the layer.

Returns List of variables.

internals_spec ()

Returns the internal states specification.

Returns Internal states specification

tf_apply (*x*, *update*)

tf_layers = {'conv1d': <sphinx.ext.autodoc._MockObject object>, 'dropout': <sphinx.e

tf_regularization_loss ()

tf_tensors (*named_tensors*)

Attaches the *named_tensors* dictionary to the layer for examination and update.

Parameters **named_tensors** – Dictionary of named tensors to be used as Input's or recorded from outputs

Returns NA

tensorflow.core.networks.network module

```
class tensorflow.core.networks.network.LayerBasedNetwork (scope='layerbased-
                                                    network', summary-
                                                    labels=())
```

Bases: *tensorflow.core.networks.network.Network*

Base class for networks using TensorForce layers.

```
__init__ (scope='layerbased-network', summary_labels=())
    Layer-based network.
```

```
add_layer (layer)
```

```
from_spec (spec, kwargs=None)
    Creates a network from a specification dict.
```

```
get_list_of_named_tensor ()
    Returns a list of the names of tensors available.

    Returns List of the names of tensors available.
```

```
get_named_tensor (name)
    Returns a named tensor if available.

    Returns True if named tensor found, False otherwise tensor: If valid, will be a tensor, otherwise
    None

    Return type valid
```

```
get_summaries ()
```

```
get_variables (include_nontrainable=False)
```

```
internals_spec ()
```

```
set_named_tensor (name, tensor)
    Returns the TensorFlow summaries reported by the network.

    Returns None
```

```
tf_apply (x, internals, update, return_internals=False)
    Creates the TensorFlow operations for applying the network to the given input.

    Parameters
    • x – Network input tensor or dict of input tensors.
    • internals – List of prior internal state tensors
    • update – Boolean tensor indicating whether this call happens during an update.
    • return_internals – If true, also returns posterior internal state tensors

    Returns Network output tensor, plus optionally list of posterior internal state tensors
```

```
tf_regularization_loss ()
```

```
class tensorflow.core.networks.network.LayeredNetwork (layers, scope='layered-
                                                    network', summary-
                                                    labels=())
```

Bases: *tensorflow.core.networks.network.LayerBasedNetwork*

Network consisting of a sequence of layers, which can be created from a specification dict.

__init__ (*layers*, *scope*='layered-network', *summary_labels*=())
Single-stack layered network.

Parameters **layers** – List of layer specification dicts.

add_layer (*layer*)

static from_json (*filename*)
Creates a `layer_networkd_builder` from a JSON.

Parameters **filename** – Path to configuration

Returns: A `layered_network_builder` function with layers generated from the JSON

from_spec (*spec*, *kwargs*=None)
Creates a network from a specification dict.

get_list_of_named_tensor ()
Returns a list of the names of tensors available.

Returns List of the names of tensors available.

get_named_tensor (*name*)
Returns a named tensor if available.

Returns True if named tensor found, False otherwise tensor: If valid, will be a tensor, otherwise None

Return type valid

get_summaries ()

get_variables (*include_nontrainable*=False)

internals_spec ()

set_named_tensor (*name*, *tensor*)
Returns the TensorFlow summaries reported by the network.

Returns None

tf_apply (*x*, *internals*, *update*, *return_internals*=False)

tf_regularization_loss ()

class `tensorforce.core.networks.network.Network` (*scope*='network', *summary_labels*=None) *sum-*

Bases: object

Base class for neural networks.

__init__ (*scope*='network', *summary_labels*=None)
Neural network.

static from_spec (*spec*, *kwargs*=None)
Creates a network from a specification dict.

get_list_of_named_tensor ()
Returns a list of the names of tensors available.

Returns List of the names of tensors available.

get_named_tensor (*name*)
Returns a named tensor if available.

Returns True if named tensor found, False otherwise tensor: If valid, will be a tensor, otherwise None

Return type valid

get_summaries ()

Returns the TensorFlow summaries reported by the network.

Returns List of summaries

get_variables (*include_nontrainable=False*)

Returns the TensorFlow variables used by the network.

Returns List of variables

internals_spec ()

Returns the internal states specification.

Returns Internal states specification

set_named_tensor (*name, tensor*)

Returns the TensorFlow summaries reported by the network.

Returns None

tf_apply (*x, internals, update, return_internals=False*)

Creates the TensorFlow operations for applying the network to the given input.

Parameters

- **x** – Network input tensor or dict of input tensors.
- **internals** – List of prior internal state tensors
- **update** – Boolean tensor indicating whether this call happens during an update.
- **return_internals** – If true, also returns posterior internal state tensors

Returns Network output tensor, plus optionally list of posterior internal state tensors

tf_regularization_loss ()

Creates the TensorFlow operations for the network regularization loss.

Returns Regularization loss tensor

Module contents

class `tensorforce.core.networks.Layer` (*scope='layer', summary_labels=None*)

Bases: `object`

Base class for network layers.

__init__ (*scope='layer', summary_labels=None*)

Layer.

static from_spec (*spec, kwargs=None*)

Creates a layer from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the layer.

Returns List of summaries.

get_variables (*include_nontrainable=False*)

Returns the TensorFlow variables used by the layer.

Returns List of variables.

internals_spec()

Returns the internal states specification.

Returns Internal states specification

tf_apply(*x, update*)

Creates the TensorFlow operations for applying the layer to the given input.

Parameters

- **x** – Layer input tensor.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Layer output tensor.

tf_regularization_loss()

Creates the TensorFlow operations for the layer regularization loss.

Returns Regularization loss tensor.

tf_tensors(*named_tensors*)

Attaches the named_tensors dictionary to the layer for examination and update.

Parameters **named_tensors** – Dictionary of named tensors to be used as Input's or recorded from outputs

Returns NA

class tensorflow.core.networks.**TFLayer**(*layer, scope='tf-layer', summary_labels=(), **kwargs*)

Bases: [tensorflow.core.networks.layer.Layer](#)

Wrapper class for TensorFlow layers.

__init__(*layer, scope='tf-layer', summary_labels=(), **kwargs*)

Creates a new layer instance of a TensorFlow layer.

Parameters

- **name** – The name of the layer, one of 'dense'.
- ****kwargs** – Additional arguments passed on to the TensorFlow layer constructor.

from_spec(*spec, kwargs=None*)

Creates a layer from a specification dict.

get_summaries()

Returns the TensorFlow summaries reported by the layer.

Returns List of summaries.

get_variables(*include_nontrainable=False*)

Returns the TensorFlow variables used by the layer.

Returns List of variables.

internals_spec()

Returns the internal states specification.

Returns Internal states specification

tf_apply(*x, update*)

tf_layers = {'conv1d': <sphinx.ext.autodoc._MockObject object>, 'dropout': <sphinx.e

tf_regularization_loss()

tf_tensors (*named_tensors*)

Attaches the `named_tensors` dictionary to the layer for examination and update.

Parameters **named_tensors** – Dictionary of named tensors to be used as Input’s or recorded from outputs

Returns NA

```
class tensorflow.core.networks.Nonlinearity (name='relu', alpha=None,  
                                              beta=1.0, max=None, min=None,  
                                              scope='nonlinearity', summary_labels=())
```

Bases: `tensorflow.core.networks.layer.Layer`

Non-linearity layer applying a non-linear transformation.

```
__init__ (name='relu', alpha=None, beta=1.0, max=None, min=None, scope='nonlinearity', summary_labels=())
```

Non-linearity activation layer.

Parameters

- **name** – Non-linearity name, one of ‘elu’, ‘relu’, ‘selu’, ‘sigmoid’, ‘swish’, ‘softmax’, ‘leaky_relu’ (or ‘lrelu’), ‘crelu’, ‘softmax’, ‘softplus’, ‘softsign’, ‘tanh’ or ‘none’.
- **alpha** – (float) Alpha value for leaky Relu
- **beta** – (float) Beta value or ‘learn’ to train value (default 1.0)
- **max** – (float) maximum (beta * input) value passed to non-linearity function
- **min** – (float) minimum (beta * input) value passed to non-linearity function
- **summary_labels** – Requested summary labels for tensorboard export, add ‘beta’ to watch beta learning

```
from_spec (spec, kwargs=None)
```

Creates a layer from a specification dict.

```
get_summaries ()
```

Returns the TensorFlow summaries reported by the layer.

Returns List of summaries.

```
get_variables (include_nontrainable=False)
```

Returns the TensorFlow variables used by the layer.

Returns List of variables.

```
internals_spec ()
```

Returns the internal states specification.

Returns Internal states specification

```
tf_apply (x, update)
```

```
tf_regularization_loss ()
```

Creates the TensorFlow operations for the layer regularization loss.

Returns Regularization loss tensor.

tf_tensors (*named_tensors*)

Attaches the `named_tensors` dictionary to the layer for examination and update.

Parameters **named_tensors** – Dictionary of named tensors to be used as Input’s or recorded from outputs

Returns NA

class tensorflow.core.networks.**Dropout** (*rate=0.0, scope='dropout', summary_labels=()*)

Bases: *tensorflow.core.networks.layer.Layer*

Dropout layer. If using dropout, add this layer after inputs and after dense layers. For LSTM, dropout is handled independently as an argument. Not available for Conv2d yet.

__init__ (*rate=0.0, scope='dropout', summary_labels=()*)

from_spec (*spec, kwargs=None*)

Creates a layer from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the layer.

Returns List of summaries.

get_variables (*include_nontrainable=False*)

Returns the TensorFlow variables used by the layer.

Returns List of variables.

internals_spec ()

Returns the internal states specification.

Returns Internal states specification

tf_apply (*x, update*)

tf_regularization_loss ()

Creates the TensorFlow operations for the layer regularization loss.

Returns Regularization loss tensor.

tf_tensors (*named_tensors*)

Attaches the named_tensors dictionary to the layer for examination and update.

Parameters **named_tensors** – Dictionary of named tensors to be used as Input's or recorded from outputs

Returns NA

class tensorflow.core.networks.**Flatten** (*scope='flatten', summary_labels=()*)

Bases: *tensorflow.core.networks.layer.Layer*

Flatten layer reshaping the input.

__init__ (*scope='flatten', summary_labels=()*)

from_spec (*spec, kwargs=None*)

Creates a layer from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the layer.

Returns List of summaries.

get_variables (*include_nontrainable=False*)

Returns the TensorFlow variables used by the layer.

Returns List of variables.

internals_spec ()

Returns the internal states specification.

Returns Internal states specification

tf_apply (*x, update*)

tf_regularization_loss()

Creates the TensorFlow operations for the layer regularization loss.

Returns Regularization loss tensor.

tf_tensors (*named_tensors*)

Attaches the *named_tensors* dictionary to the layer for examination and update.

Parameters **named_tensors** – Dictionary of named tensors to be used as Input's or recorded from outputs

Returns NA

class `tensorforce.core.networks.Pool2d` (*pooling_type='max', window=2, stride=2, padding='SAME', scope='pool2d', summary_labels=()*)

Bases: `tensorforce.core.networks.layer.Layer`

2-dimensional pooling layer.

__init__ (*pooling_type='max', window=2, stride=2, padding='SAME', scope='pool2d', summary_labels=()*)

2-dimensional pooling layer.

Parameters

- **pooling_type** – Either 'max' or 'average'.
- **window** – Pooling window size, either an integer or pair of integers.
- **stride** – Pooling stride, either an integer or pair of integers.
- **padding** – Pooling padding, one of 'VALID' or 'SAME'.

from_spec (*spec, kwargs=None*)

Creates a layer from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the layer.

Returns List of summaries.

get_variables (*include_nontrainable=False*)

Returns the TensorFlow variables used by the layer.

Returns List of variables.

internals_spec ()

Returns the internal states specification.

Returns Internal states specification

tf_apply (*x, update*)

tf_regularization_loss ()

Creates the TensorFlow operations for the layer regularization loss.

Returns Regularization loss tensor.

tf_tensors (*named_tensors*)

Attaches the *named_tensors* dictionary to the layer for examination and update.

Parameters **named_tensors** – Dictionary of named tensors to be used as Input's or recorded from outputs

Returns NA

```
class tensorflow.core.networks.Embedding(indices, size, l2_regularization=0.0,
                                         l1_regularization=0.0, scope='embedding',
                                         summary_labels=())
```

Bases: [tensorflow.core.networks.layer.Layer](#)

Embedding layer.

```
__init__(indices, size, l2_regularization=0.0, l1_regularization=0.0, scope='embedding',
          summary_labels=())
Embedding layer.
```

Parameters

- **indices** – Number of embedding indices.
- **size** – Embedding size.
- **l2_regularization** – L2 regularization weight.
- **l1_regularization** – L1 regularization weight.

```
from_spec(spec, kwargs=None)
Creates a layer from a specification dict.
```

```
get_summaries()
Returns the TensorFlow summaries reported by the layer.
```

Returns List of summaries.

```
get_variables(include_nontrainable=False)
Returns the TensorFlow variables used by the layer.
```

Returns List of variables.

```
internals_spec()
Returns the internal states specification.
```

Returns Internal states specification

```
tf_apply(x, update)
```

```
tf_regularization_loss()
```

```
tf_tensors(named_tensors)
Attaches the named_tensors dictionary to the layer for examination and update.
```

Parameters **named_tensors** – Dictionary of named tensors to be used as Input's or recorded from outputs

Returns NA

```
class tensorflow.core.networks.Linear(size, weights=None, bias=True,
                                       l2_regularization=0.0, l1_regularization=0.0,
                                       scope='linear', summary_labels=())
```

Bases: [tensorflow.core.networks.layer.Layer](#)

Linear fully-connected layer.

```
__init__(size, weights=None, bias=True, l2_regularization=0.0, l1_regularization=0.0,
          scope='linear', summary_labels=())
Linear layer.
```

Parameters

- **size** – Layer size.
- **weights** – Weight initialization, random if None.

- **bias** – Bias initialization, random if True, no bias added if False.
- **l2_regularization** – L2 regularization weight.
- **l1_regularization** – L1 regularization weight.

from_spec (*spec*, *kwargs=None*)

Creates a layer from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the layer.

Returns List of summaries.

get_variables (*include_nontrainable=False*)

Returns the TensorFlow variables used by the layer.

Returns List of variables.

internals_spec ()

Returns the internal states specification.

Returns Internal states specification

tf_apply (*x*, *update=False*)

tf_regularization_loss ()

tf_tensors (*named_tensors*)

Attaches the *named_tensors* dictionary to the layer for examination and update.

Parameters **named_tensors** – Dictionary of named tensors to be used as Input's or recorded from outputs

Returns NA

```
class tensorforce.core.networks.Dense (size=None, weights=None, bias=True,
                                     activation='relu', l2_regularization=0.0,
                                     l1_regularization=0.0, skip=False, scope='dense',
                                     summary_labels=())
```

Bases: `tensorforce.core.networks.layer.Layer`

Dense layer, i.e. linear fully connected layer with subsequent non-linearity.

```
__init__ (size=None, weights=None, bias=True, activation='relu', l2_regularization=0.0,
          l1_regularization=0.0, skip=False, scope='dense', summary_labels=())
```

Dense layer.

Parameters

- **size** – Layer size, if None than input size matches the output size of the layer
- **weights** – Weight initialization, random if None.
- **bias** – If true, bias is added.
- **activation** – Type of nonlinearity, or dict with name & arguments
- **l2_regularization** – L2 regularization weight.
- **l1_regularization** – L1 regularization weight.
- **skip** – Add skip connection like ResNet (<https://arxiv.org/pdf/1512.03385.pdf>), doubles layers and ShortCut from Input to output

from_spec (*spec*, *kwargs=None*)

Creates a layer from a specification dict.

get_summaries ()

get_variables (*include_nontrainable=False*)

internals_spec ()

Returns the internal states specification.

Returns Internal states specification

tf_apply (*x, update*)

tf_regularization_loss ()

tf_tensors (*named_tensors*)

Attaches the *named_tensors* dictionary to the layer for examination and update.

Parameters **named_tensors** – Dictionary of named tensors to be used as Input's or recorded from outputs

Returns NA

class `tensorforce.core.networks.Dueling` (*size*, *bias=False*, *activation='none'*,
l2_regularization=0.0, *l1_regularization=0.0*, *output=None*, *scope='dueling'*, *summary_labels=()*)

Bases: `tensorforce.core.networks.layer.Layer`

Dueling layer, i.e. Duel pipelines for Exp & Adv to help with stability

__init__ (*size*, *bias=False*, *activation='none'*, *l2_regularization=0.0*, *l1_regularization=0.0*, *output=None*, *scope='dueling'*, *summary_labels=()*)

Dueling layer.

[Dueling Networks] (<https://arxiv.org/pdf/1511.06581.pdf>) Implement $Y = \text{Expectation}[x] + (\text{Advantage}[x] - \text{Mean}(\text{Advantage}[x]))$

Parameters

- **size** – Layer size.
- **bias** – If true, bias is added.
- **activation** – Type of nonlinearity, or dict with name & arguments
- **l2_regularization** – L2 regularization weight.
- **l1_regularization** – L1 regularization weight.
- **output** – None or tuple of output names for ('expectation', 'advantage', 'mean_advantage')

from_spec (*spec*, *kwargs=None*)

Creates a layer from a specification dict.

get_summaries ()

get_variables (*include_nontrainable=False*)

internals_spec ()

Returns the internal states specification.

Returns Internal states specification

tf_apply (*x, update*)

tf_regularization_loss ()

tf_tensors (*named_tensors*)

Attaches the *named_tensors* dictionary to the layer for examination and update.

Parameters `named_tensors` – Dictionary of named tensors to be used as Input’s or recorded from outputs

Returns NA

```
class tensorflow.core.networks.Conv1d(size, window=3, stride=1, padding='SAME',
                                     bias=True, activation='relu', l2_regularization=0.0,
                                     l1_regularization=0.0, scope='conv1d', summary_labels=())
```

Bases: `tensorflow.core.networks.layer.Layer`

1-dimensional convolutional layer.

```
__init__(size, window=3, stride=1, padding='SAME', bias=True, activation='relu',
         l2_regularization=0.0, l1_regularization=0.0, scope='conv1d', summary_labels=())
```

1D convolutional layer.

Parameters

- **size** – Number of filters
- **window** – Convolution window size
- **stride** – Convolution stride
- **padding** – Convolution padding, one of ‘VALID’ or ‘SAME’
- **bias** – If true, a bias is added
- **activation** – Type of nonlinearity, or dict with name & arguments
- **l2_regularization** – L2 regularization weight
- **l1_regularization** – L1 regularization weight

```
from_spec(spec, kwargs=None)
```

Creates a layer from a specification dict.

```
get_summaries()
```

```
get_variables(include_nontrainable=False)
```

```
internals_spec()
```

Returns the internal states specification.

Returns Internal states specification

```
tf_apply(x, update)
```

```
tf_regularization_loss()
```

```
tf_tensors(named_tensors)
```

Attaches the `named_tensors` dictionary to the layer for examination and update.

Parameters `named_tensors` – Dictionary of named tensors to be used as Input’s or recorded from outputs

Returns NA

```
class tensorflow.core.networks.Conv2d(size, window=3, stride=1, padding='SAME',
                                     bias=True, activation='relu', l2_regularization=0.0,
                                     l1_regularization=0.0, scope='conv2d', summary_labels=())
```

Bases: `tensorflow.core.networks.layer.Layer`

2-dimensional convolutional layer.

```
__init__ (size, window=3, stride=1, padding='SAME', bias=True, activation='relu',  
          l2_regularization=0.0, l1_regularization=0.0, scope='conv2d', summary_labels=())  
2D convolutional layer.
```

Parameters

- **size** – Number of filters
- **window** – Convolution window size, either an integer or pair of integers.
- **stride** – Convolution stride, either an integer or pair of integers.
- **padding** – Convolution padding, one of 'VALID' or 'SAME'
- **bias** – If true, a bias is added
- **activation** – Type of nonlinearity, or dict with name & arguments
- **l2_regularization** – L2 regularization weight
- **l1_regularization** – L1 regularization weight

```
from_spec (spec, kwargs=None)  
Creates a layer from a specification dict.
```

```
get_summaries ()
```

```
get_variables (include_nontrainable=False)
```

```
internals_spec ()  
Returns the internal states specification.
```

Returns Internal states specification

```
tf_apply (x, update)
```

```
tf_regularization_loss ()
```

```
tf_tensors (named_tensors)  
Attaches the named_tensors dictionary to the layer for examination and update.
```

Parameters **named_tensors** – Dictionary of named tensors to be used as Input's or recorded from outputs

Returns NA

```
class tensorflow.core.networks.InternalLstm (size, dropout=None, lstmcell_args={},  
                                              scope='internal_lstm', summary_labels=())
```

Bases: [tensorflow.core.networks.layer.Layer](#)

Long short-term memory layer for internal state management.

```
__init__ (size, dropout=None, lstmcell_args={}, scope='internal_lstm', summary_labels=())  
LSTM layer.
```

Parameters

- **size** – LSTM size.
- **dropout** – Dropout rate.

```
from_spec (spec, kwargs=None)  
Creates a layer from a specification dict.
```

```
get_summaries ()  
Returns the TensorFlow summaries reported by the layer.
```

Returns List of summaries.

get_variables (*include_nontrainable=False*)

Returns the TensorFlow variables used by the layer.

Returns List of variables.

internals_spec ()

tf_apply (*x, update, state*)

tf_regularization_loss ()

Creates the TensorFlow operations for the layer regularization loss.

Returns Regularization loss tensor.

tf_tensors (*named_tensors*)

Attaches the named_tensors dictionary to the layer for examination and update.

Parameters **named_tensors** – Dictionary of named tensors to be used as Input's or recorded from outputs

Returns NA

class `tensorforce.core.networks.Lstm` (*size, dropout=None, scope='lstm', summary_labels=(), return_final_state=True*)

Bases: `tensorforce.core.networks.layer.Layer`

__init__ (*size, dropout=None, scope='lstm', summary_labels=(), return_final_state=True*)
LSTM layer.

Parameters

- **size** – LSTM size.
- **dropout** – Dropout rate.

from_spec (*spec, kwargs=None*)

Creates a layer from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the layer.

Returns List of summaries.

get_variables (*include_nontrainable=False*)

Returns the TensorFlow variables used by the layer.

Returns List of variables.

internals_spec ()

Returns the internal states specification.

Returns Internal states specification

tf_apply (*x, update, sequence_length=None*)

tf_regularization_loss ()

Creates the TensorFlow operations for the layer regularization loss.

Returns Regularization loss tensor.

tf_tensors (*named_tensors*)

Attaches the named_tensors dictionary to the layer for examination and update.

Parameters **named_tensors** – Dictionary of named tensors to be used as Input's or recorded from outputs

Returns NA

class tensorflow.core.networks.**Network** (*scope='network', summary_labels=None*)

Bases: object

Base class for neural networks.

__init__ (*scope='network', summary_labels=None*)

Neural network.

static from_spec (*spec, kwargs=None*)

Creates a network from a specification dict.

get_list_of_named_tensor ()

Returns a list of the names of tensors available.

Returns List of the names of tensors available.

get_named_tensor (*name*)

Returns a named tensor if available.

Returns True if named tensor found, False otherwise
tensor: If valid, will be a tensor, otherwise None

Return type valid

get_summaries ()

Returns the TensorFlow summaries reported by the network.

Returns List of summaries

get_variables (*include_nontrainable=False*)

Returns the TensorFlow variables used by the network.

Returns List of variables

internals_spec ()

Returns the internal states specification.

Returns Internal states specification

set_named_tensor (*name, tensor*)

Returns the TensorFlow summaries reported by the network.

Returns None

tf_apply (*x, internals, update, return_internals=False*)

Creates the TensorFlow operations for applying the network to the given input.

Parameters

- **x** – Network input tensor or dict of input tensors.
- **internals** – List of prior internal state tensors
- **update** – Boolean tensor indicating whether this call happens during an update.
- **return_internals** – If true, also returns posterior internal state tensors

Returns Network output tensor, plus optionally list of posterior internal state tensors

tf_regularization_loss ()

Creates the TensorFlow operations for the network regularization loss.

Returns Regularization loss tensor


```

class tensorflow.core.networks.LayerBasedNetwork (scope='layerbased-network', summary_labels=())
    Bases: tensorflow.core.networks.network.Network
    Base class for networks using TensorForce layers.
    __init__ (scope='layerbased-network', summary_labels=())
        Layer-based network.
    add_layer (layer)
    from_spec (spec, kwargs=None)
        Creates a network from a specification dict.
    get_list_of_named_tensor ()
        Returns a list of the names of tensors available.
        Returns List of the names of tensors available.
    get_named_tensor (name)
        Returns a named tensor if available.
        Returns True if named tensor found, False otherwise tensor: If valid, will be a tensor, otherwise
        None
        Return type valid
    get_summaries ()
    get_variables (include_nontrainable=False)
    internals_spec ()
    set_named_tensor (name, tensor)
        Returns the TensorFlow summaries reported by the network.
        Returns None
    tf_apply (x, internals, update, return_internals=False)
        Creates the TensorFlow operations for applying the network to the given input.
        Parameters
        • x – Network input tensor or dict of input tensors.
        • internals – List of prior internal state tensors
        • update – Boolean tensor indicating whether this call happens during an update.
        • return_internals – If true, also returns posterior internal state tensors
        Returns Network output tensor, plus optionally list of posterior internal state tensors
    tf_regularization_loss ()
class tensorflow.core.networks.LayeredNetwork (layers, scope='layered-network', summary_labels=())
    Bases: tensorflow.core.networks.network.LayerBasedNetwork
    Network consisting of a sequence of layers, which can be created from a specification dict.
    __init__ (layers, scope='layered-network', summary_labels=())
        Single-stack layered network.
        Parameters layers – List of layer specification dicts.
    add_layer (layer)

```

static from_json (*filename*)

Creates a layer_networkd_builder from a JSON.

Parameters *filename* – Path to configuration

Returns: A layered_network_builder function with layers generated from the JSON

from_spec (*spec*, *kwargs=None*)

Creates a network from a specification dict.

get_list_of_named_tensor ()

Returns a list of the names of tensors available.

Returns List of the names of tensors available.

get_named_tensor (*name*)

Returns a named tensor if available.

Returns True if named tensor found, False otherwise
tensor: If valid, will be a tensor, otherwise None

Return type valid

get_summaries ()

get_variables (*include_nontrainable=False*)

internals_spec ()

set_named_tensor (*name*, *tensor*)

Returns the TensorFlow summaries reported by the network.

Returns None

tf_apply (*x*, *internals*, *update*, *return_internals=False*)

tf_regularization_loss ()

tensorforce.core.optimizers package

Subpackages

tensorforce.core.optimizers.solvers package

Submodules

tensorforce.core.optimizers.solvers.conjugate_gradient module

class tensorforce.core.optimizers.solvers.conjugate_gradient.**ConjugateGradient** (*max_iterations*, *damping*, *unroll_loop=False*)

Bases: *tensorforce.core.optimizers.solvers.iterative.Iterative*

Conjugate gradient algorithm which iteratively finds a solution \mathbf{x} for a system of linear equations of the form $\mathbf{A} \mathbf{x} = \mathbf{b}$, where $\mathbf{A} \mathbf{x}$ could be, for instance, a locally linear approximation of a high-dimensional function.

See below pseudo-code taken from [Wikipedia](#):

```
def conjgrad(A, b, x_0):
    r_0 := b - A * x_0
    c_0 := r_0
    r_0^2 := r^T * r

    for t in 0, ..., max_iterations - 1:
        Ac := A * c_t
        cAc := c_t^T * Ac
        lpha := r_t^2 / cAc
        x_{t+1} := x_t + lpha * c_t
        r_{t+1} := r_t - lpha * Ac
        r_{t+1}^2 := r_{t+1}^T * r_{t+1}
        if r_{t+1} < \epsilon:
            break
        eta = r_{t+1}^2 / r_t^2
        c_{t+1} := r_{t+1} + eta * c_t

    return x_{t+1}
```

__init__ (*max_iterations, damping, unroll_loop=False*)
Creates a new conjugate gradient solver instance.

Parameters

- **max_iterations** – Maximum number of iterations before termination.
- **damping** – Damping factor.
- **unroll_loop** – Unrolls the TensorFlow while loop if true.

from_config (*config, kwargs=None*)
Creates a solver from a specification dict.

tf_initialize (*x_init, b*)
Initialization step preparing the arguments for the first iteration of the loop body: $x_0, 0, p_0, r_0, r_0^2$.

Parameters

- **x_init** – Initial solution guess x_0 , zero vector if None.
- **b** – The right-hand side b of the system of linear equations.

Returns Initial arguments for `tf_step`.

tf_next_step (*x, iteration, conjugate, residual, squared_residual*)
Termination condition: max number of iterations, or residual sufficiently small.

Parameters

- **x** – Current solution estimate x_t .
- **iteration** – Current iteration counter t .
- **conjugate** – Current conjugate c_t .
- **residual** – Current residual r_t .
- **squared_residual** – Current squared residual r_t^2 .

Returns True if another iteration should be performed.

tf_solve (*fn_x, x_init, b*)
Iteratively solves the system of linear equations $Ax = b$.

Parameters

- **fn_x** – A callable returning the left-hand side Ax of the system of linear equations.
- **x_init** – Initial solution guess x_0 , zero vector if None.
- **b** – The right-hand side b of the system of linear equations.

Returns A solution x to the problem as given by the solver.

tf_step (*x, iteration, conjugate, residual, squared_residual*)
Iteration loop body of the conjugate gradient algorithm.

Parameters

- **x** – Current solution estimate x_t .
- **iteration** – Current iteration counter t .
- **conjugate** – Current conjugate c_t .
- **residual** – Current residual r_t .
- **squared_residual** – Current squared residual r_t^2 .

Returns Updated arguments for next iteration.

tensorforce.core.optimizers.solvers.iterative module

```
class tensorforce.core.optimizers.solvers.iterative.Iterative (max_iterations,  
                                                                un-  
                                                                roll_loop=False)
```

Bases: `tensorforce.core.optimizers.solvers.solver.Solver`

Generic solver which *iteratively* solves an equation/optimization problem. Involves an initialization step, the iteration loop body and the termination condition.

__init__ (*max_iterations, unroll_loop=False*)
Creates a new iterative solver instance.

Parameters

- **max_iterations** – Maximum number of iterations before termination.
- **unroll_loop** – Unrolls the TensorFlow while loop if true.

from_config (*config, kwargs=None*)
Creates a solver from a specification dict.

tf_initialize (*x_init, *args*)
Initialization step preparing the arguments for the first iteration of the loop body (default: initial solution guess and iteration counter).

Parameters

- **x_init** – Initial solution guess x_0 .
- ***args** – Additional solver-specific arguments.

Returns Initial arguments for `tf_step`.

tf_next_step (*x, iteration, *args*)
Termination condition (default: max number of iterations).

Parameters

- **x** – Current solution estimate.

- **iteration** – Current iteration counter.
- ***args** – Additional solver-specific arguments.

Returns True if another iteration should be performed.

tf_solve (*fn_x, x_init, *args*)

Iteratively solves an equation/optimization for x involving an expression $f(x)$.

Parameters

- **fn_x** – A callable returning an expression $f(x)$ given x .
- **x_init** – Initial solution guess x_0 .
- ***args** – Additional solver-specific arguments.

Returns A solution x to the problem as given by the solver.

tf_step (*x, iteration, *args*)

Iteration loop body of the iterative solver (default: increment iteration step). The first two loop arguments have to be the current solution estimate and the iteration step.

Parameters

- **x** – Current solution estimate.
- **iteration** – Current iteration counter.
- ***args** – Additional solver-specific arguments.

Returns Updated arguments for next iteration.

tensorforce.core.optimizers.solvers.line_search module

```
class tensorforce.core.optimizers.solvers.line_search.LineSearch (max_iterations,  
                                                                accept_ratio,  
                                                                mode, pa-  
                                                                rameter, un-  
                                                                roll_loop=False)
```

Bases: *tensorforce.core.optimizers.solvers.iterative.Iterative*

Line search algorithm which iteratively optimizes the value $f(x)$ for x on the line between x' and x_0 by optimistically taking the first acceptable x starting from x_0 and moving towards x' .

__init__ (*max_iterations, accept_ratio, mode, parameter, unroll_loop=False*)

Creates a new line search solver instance.

Parameters

- **max_iterations** – Maximum number of iterations before termination.
- **accept_ratio** – Lower limit of what improvement ratio over $x = x'$ is acceptable (based either on a given estimated improvement or with respect to the value at $x = x'$).
- **mode** – Mode of movement between x_0 and x' , either 'linear' or 'exponential'.
- **parameter** – Movement mode parameter, additive or multiplicative, respectively.
- **unroll_loop** – Unrolls the TensorFlow while loop if true.

from_config (*config, kwargs=None*)

Creates a solver from a specification dict.

tf_initialize (*x_init*, *base_value*, *target_value*, *estimated_improvement*)

Initialization step preparing the arguments for the first iteration of the loop body.

Parameters

- **x_init** – Initial solution guess x_0 .
- **base_value** – Value $f(x')$ at $x = x'$.
- **target_value** – Value $f(x_0)$ at $x = x_0$.
- **estimated_improvement** – Estimated value at $x = x_0$, $f(x')$ if None.

Returns Initial arguments for `tf_step`.

tf_next_step (*x*, *iteration*, *deltas*, *improvement*, *last_improvement*, *estimated_improvement*)

Termination condition: max number of iterations, or no improvement for last step, or improvement less than acceptable ratio, or estimated value not positive.

Parameters

- **x** – Current solution estimate x_t .
- **iteration** – Current iteration counter t .
- **deltas** – Current difference $x_t - x'$.
- **improvement** – Current improvement $(f(x_t) - f(x')) / v'$.
- **last*improvement** – Last improvement $(f(x_{t-1}) - f(x')) / v'$.
- **estimated_improvement** – Current estimated value v' .

Returns True if another iteration should be performed.

tf_solve (*fn_x*, *x_init*, *base_value*, *target_value*, *estimated_improvement=None*)

Iteratively optimizes $f(x)$ for x on the line between x' and x_0 .

Parameters

- **fn_x** – A callable returning the value $f(x)$ at x .
- **x_init** – Initial solution guess x_0 .
- **base_value** – Value $f(x')$ at $x = x'$.
- **target_value** – Value $f(x_0)$ at $x = x_0$.
- **estimated_improvement** – Estimated improvement for $x = x_0$, $f(x')$ if None.

Returns A solution x to the problem as given by the solver.

tf_step (*x*, *iteration*, *deltas*, *improvement*, *last_improvement*, *estimated_improvement*)

Iteration loop body of the line search algorithm.

Parameters

- **x** – Current solution estimate x_t .
- **iteration** – Current iteration counter t .
- **deltas** – Current difference $x_t - x'$.
- **improvement** – Current improvement $(f(x_t) - f(x')) / v'$.
- **last*improvement** – Last improvement $(f(x_{t-1}) - f(x')) / v'$.
- **estimated_improvement** – Current estimated value v' .

Returns Updated arguments for next iteration.

tensorflow.core.optimizers.solvers.solver module**class** tensorflow.core.optimizers.solvers.solver.Solver

Bases: object

Generic TensorFlow-based solver which solves a not yet further specified equation/optimization problem.

__init__ ()

Creates a new solver instance.

static from_config (config, kwargs=None)

Creates a solver from a specification dict.

tf_solve (fn_x, *args)Solves an equation/optimization for x involving an expression $f(x)$.**Parameters**

- **fn_x** – A callable returning an expression $f(x)$ given x .
- ***args** – Additional solver-specific arguments.

Returns A solution x to the problem as given by the solver.**Module contents****class** tensorflow.core.optimizers.solvers.Solver

Bases: object

Generic TensorFlow-based solver which solves a not yet further specified equation/optimization problem.

__init__ ()

Creates a new solver instance.

static from_config (config, kwargs=None)

Creates a solver from a specification dict.

tf_solve (fn_x, *args)Solves an equation/optimization for x involving an expression $f(x)$.**Parameters**

- **fn_x** – A callable returning an expression $f(x)$ given x .
- ***args** – Additional solver-specific arguments.

Returns A solution x to the problem as given by the solver.**class** tensorflow.core.optimizers.solvers.Iterative(*max_iterations*, *unroll_loop=False*)Bases: *tensorflow.core.optimizers.solvers.solver.Solver*Generic solver which *iteratively* solves an equation/optimization problem. Involves an initialization step, the iteration loop body and the termination condition.**__init__** (*max_iterations*, *unroll_loop=False*)

Creates a new iterative solver instance.

Parameters

- **max_iterations** – Maximum number of iterations before termination.
- **unroll_loop** – Unrolls the TensorFlow while loop if true.


```
def conjgrad(A, b, x_0):
    r_0 := b - A * x_0
    c_0 := r_0
    r_0^2 := r^T * r

    for t in 0, ..., max_iterations - 1:
        Ac := A * c_t
        cAc := c_t^T * Ac
        lpha := r_t^2 / cAc
        x_{t+1} := x_t + lpha * c_t
        r_{t+1} := r_t - lpha * Ac
        r_{t+1}^2 := r_{t+1}^T * r_{t+1}
        if r_{t+1} < \epsilon:
            break
        eta = r_{t+1}^2 / r_t^2
        c_{t+1} := r_{t+1} + eta * c_t

    return x_{t+1}
```

__init__ (*max_iterations, damping, unroll_loop=False*)
Creates a new conjugate gradient solver instance.

Parameters

- **max_iterations** – Maximum number of iterations before termination.
- **damping** – Damping factor.
- **unroll_loop** – Unrolls the TensorFlow while loop if true.

from_config (*config, kwargs=None*)
Creates a solver from a specification dict.

tf_initialize (*x_init, b*)
Initialization step preparing the arguments for the first iteration of the loop body: $x_0, 0, p_0, r_0, r_0^2$.

Parameters

- **x_init** – Initial solution guess x_0 , zero vector if None.
- **b** – The right-hand side b of the system of linear equations.

Returns Initial arguments for `tf_step`.

tf_next_step (*x, iteration, conjugate, residual, squared_residual*)
Termination condition: max number of iterations, or residual sufficiently small.

Parameters

- **x** – Current solution estimate x_t .
- **iteration** – Current iteration counter t .
- **conjugate** – Current conjugate c_t .
- **residual** – Current residual r_t .
- **squared_residual** – Current squared residual r_t^2 .

Returns True if another iteration should be performed.

tf_solve (*fn_x, x_init, b*)
Iteratively solves the system of linear equations $Ax = b$.

Parameters

- **fn_x** – A callable returning the left-hand side Ax of the system of linear equations.
- **x_init** – Initial solution guess x_0 , zero vector if None.
- **b** – The right-hand side b of the system of linear equations.

Returns A solution x to the problem as given by the solver.

tf_step (*x, iteration, conjugate, residual, squared_residual*)
Iteration loop body of the conjugate gradient algorithm.

Parameters

- **x** – Current solution estimate x_t .
- **iteration** – Current iteration counter t .
- **conjugate** – Current conjugate c_t .
- **residual** – Current residual r_t .
- **squared_residual** – Current squared residual r_t^2 .

Returns Updated arguments for next iteration.

class `tensorforce.core.optimizers.solvers.LineSearch` (*max_iterations, accept_ratio, mode, parameter, unroll_loop=False*)

Bases: `tensorforce.core.optimizers.solvers.iterative.Iterative`

Line search algorithm which iteratively optimizes the value $f(x)$ for x on the line between x' and x_0 by optimistically taking the first acceptable x starting from x_0 and moving towards x' .

__init__ (*max_iterations, accept_ratio, mode, parameter, unroll_loop=False*)
Creates a new line search solver instance.

Parameters

- **max_iterations** – Maximum number of iterations before termination.
- **accept_ratio** – Lower limit of what improvement ratio over $x = x'$ is acceptable (based either on a given estimated improvement or with respect to the value at $x = x'$).
- **mode** – Mode of movement between x_0 and x' , either 'linear' or 'exponential'.
- **parameter** – Movement mode parameter, additive or multiplicative, respectively.
- **unroll_loop** – Unrolls the TensorFlow while loop if true.

from_config (*config, kwargs=None*)
Creates a solver from a specification dict.

tf_initialize (*x_init, base_value, target_value, estimated_improvement*)
Initialization step preparing the arguments for the first iteration of the loop body.

Parameters

- **x_init** – Initial solution guess x_0 .
- **base_value** – Value $f(x')$ at $x = x'$.
- **target_value** – Value $f(x_0)$ at $x = x_0$.
- **estimated_improvement** – Estimated value at $x = x_0$, $f(x')$ if None.

Returns Initial arguments for `tf_step`.

tf_next_step (*x, iteration, deltas, improvement, last_improvement, estimated_improvement*)

Termination condition: max number of iterations, or no improvement for last step, or improvement less than acceptable ratio, or estimated value not positive.

Parameters

- **x** – Current solution estimate x_t .
- **iteration** – Current iteration counter t .
- **deltas** – Current difference $x_t - x'$.
- **improvement** – Current improvement $(f(x_t) - f(x')) / v'$.
- **last*improvement** – Last improvement $(f(x_{t-1}) - f(x')) / v'$.
- **estimated_improvement** – Current estimated value v' .

Returns True if another iteration should be performed.

tf_solve (*fn_x, x_init, base_value, target_value, estimated_improvement=None*)

Iteratively optimizes $f(x)$ for x on the line between x' and x_0 .

Parameters

- **fn_x** – A callable returning the value $f(x)$ at x .
- **x_init** – Initial solution guess x_0 .
- **base_value** – Value $f(x')$ at $x = x'$.
- **target_value** – Value $f(x_0)$ at $x = x_0$.
- **estimated_improvement** – Estimated improvement for $x = x_0$, $f(x')$ if None.

Returns A solution x to the problem as given by the solver.

tf_step (*x, iteration, deltas, improvement, last_improvement, estimated_improvement*)

Iteration loop body of the line search algorithm.

Parameters

- **x** – Current solution estimate x_t .
- **iteration** – Current iteration counter t .
- **deltas** – Current difference $x_t - x'$.
- **improvement** – Current improvement $(f(x_t) - f(x')) / v'$.
- **last*improvement** – Last improvement $(f(x_{t-1}) - f(x')) / v'$.
- **estimated_improvement** – Current estimated value v' .

Returns Updated arguments for next iteration.

Submodules

tensorforce.core.optimizers.clipped_step module

```
class tensorforce.core.optimizers.clipped_step.ClippedStep(optimizer, clip-
                                                           ping_value,
                                                           scope='clipped-step',
                                                           summary_labels=())
Bases: tensorforce.core.optimizers.meta_optimizer.MetaOptimizer
```

The clipped-shep meta optimizer clips the values of the optimization step proposed by another optimizer.

__init__ (*optimizer, clipping_value, scope='clipped-step', summary_labels=()*)
Creates a new multi-step meta optimizer instance.

Parameters

- **optimizer** – The optimizer which is modified by this meta optimizer.
- **clipping_value** – Clip deltas at this value.

apply_step (*variables, deltas*)
Applies step deltas to variable values.

Parameters

- **variables** – List of variables.
- **deltas** – List of deltas of same length.

Returns The step-applied operation.

from_spec (*spec, kwargs=None*)
Creates an optimizer from a specification dict.

get_summaries ()
Returns the TensorFlow summaries reported by the optimizer.

Returns List of summaries.

get_variables ()

minimize (*time, variables, **kwargs*)
Performs an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- ****kwargs** – Additional optimizer-specific arguments. The following arguments are used by some optimizers:
- **arguments** (–) – Dict of arguments for callables, like `fn_loss`.
- **fn_loss** (–) – A callable returning the loss of the current model.
- **fn_reference** (–) – A callable returning the reference values, in case of a comparative loss.
- **fn_kl_divergence** (–) – A callable returning the KL-divergence relative to the current model.
- **return_estimated_improvement** (–) – Returns the estimated improvement resulting from the natural gradient calculation if true.
- **source_variables** (–) – List of source variables to synchronize with.
- **global_variables** (–) – List of global variables to apply the proposed optimization step to.

Returns The optimization operation.

tf_step (*time, variables, **kwargs*)
Creates the TensorFlow operations for performing an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- ****kwargs** – Additional arguments passed on to the internal optimizer.

Returns List of delta tensors corresponding to the updates for each optimized variable.

tensorforce.core.optimizers.evolutionary module

```
class tensorforce.core.optimizers.evolutionary.Evolutionary(learning_rate,
                                                           num_samples=1,
                                                           unroll_loop=False,
                                                           scope='evolutionary',
                                                           summary_labels=())
```

Bases: *tensorforce.core.optimizers.optimizer.Optimizer*

Evolutionary optimizer which samples random perturbations and applies them either positively or negatively, depending on their improvement of the loss.

```
__init__(learning_rate, num_samples=1, unroll_loop=False, scope='evolutionary', summary_labels=())
```

Creates a new evolutionary optimizer instance.

Parameters

- **learning_rate** – Learning rate.
- **num_samples** – Number of sampled perturbations.

```
apply_step(variables, deltas)
```

Applies step deltas to variable values.

Parameters

- **variables** – List of variables.
- **deltas** – List of deltas of same length.

Returns The step-applied operation.

```
from_spec(spec, kwargs=None)
```

Creates an optimizer from a specification dict.

```
get_summaries()
```

Returns the TensorFlow summaries reported by the optimizer.

Returns List of summaries.

```
get_variables()
```

Returns the TensorFlow variables used by the optimizer.

Returns List of variables.

```
minimize(time, variables, **kwargs)
```

Performs an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.

- ****kwargs** – Additional optimizer-specific arguments. The following arguments are used by some optimizers:
- **arguments** (-) – Dict of arguments for callables, like `fn_loss`.
- **fn_loss** (-) – A callable returning the loss of the current model.
- **fn_reference** (-) – A callable returning the reference values, in case of a comparative loss.
- **fn_kl_divergence** (-) – A callable returning the KL-divergence relative to the current model.
- **return_estimated_improvement** (-) – Returns the estimated improvement resulting from the natural gradient calculation if true.
- **source_variables** (-) – List of source variables to synchronize with.
- **global_variables** (-) – List of global variables to apply the proposed optimization step to.

Returns The optimization operation.

tf_step (*time, variables, arguments, fn_loss, **kwargs*)

Creates the TensorFlow operations for performing an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- **arguments** – Dict of arguments for callables, like `fn_loss`.
- **fn_loss** – A callable returning the loss of the current model.
- ****kwargs** – Additional arguments, not used.

Returns List of delta tensors corresponding to the updates for each optimized variable.

tensorflow.core.optimizers.global_optimizer module

```
class tensorflow.core.optimizers.global_optimizer.GlobalOptimizer(optimizer,
                                                                scope='global-optimizer',
                                                                summary_labels=())
```

Bases: `tensorflow.core.optimizers.meta_optimizer.MetaOptimizer`

The global optimizer applies an optimizer to the local variables. In addition, it also applies the update to a corresponding set of global variables and subsequently updates the local variables to the value of these global variables. Note: This is used for the current distributed mode, and will likely change with the next major version update.

```
__init__(optimizer, scope='global-optimizer', summary_labels=())
```

Creates a new global optimizer instance.

Parameters **optimizer** – The optimizer which is modified by this meta optimizer.

apply_step (*variables, deltas*)

Applies step deltas to variable values.

Parameters

- **variables** – List of variables.
- **deltas** – List of deltas of same length.

Returns The step-applied operation.

from_spec (*spec*, *kwargs=None*)

Creates an optimizer from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the optimizer.

Returns List of summaries.

get_variables ()

minimize (*time*, *variables*, ***kwargs*)

Performs an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- ****kwargs** – Additional optimizer-specific arguments. The following arguments are used by some optimizers:
- **arguments** (-) – Dict of arguments for callables, like `fn_loss`.
- **fn_loss** (-) – A callable returning the loss of the current model.
- **fn_reference** (-) – A callable returning the reference values, in case of a comparative loss.
- **fn_kl_divergence** (-) – A callable returning the KL-divergence relative to the current model.
- **return_estimated_improvement** (-) – Returns the estimated improvement resulting from the natural gradient calculation if true.
- **source_variables** (-) – List of source variables to synchronize with.
- **global_variables** (-) – List of global variables to apply the proposed optimization step to.

Returns The optimization operation.

tf_step (*time*, *variables*, *global_variables*, ***kwargs*)

Creates the TensorFlow operations for performing an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- **global_variables** – List of global variables to apply the proposed optimization step to.
- ****kwargs** – ??? coming soon

Returns List of delta tensors corresponding to the updates for each optimized variable.

tensorforce.core.optimizers.meta_optimizer module

```
class tensorforce.core.optimizers.meta_optimizer.MetaOptimizer (optimizer,  
                                                             scope='meta-  
optimizer', summary_labels=(),  
                                                             **kwargs)
```

Bases: *tensorforce.core.optimizers.optimizer.Optimizer*

A meta optimizer takes the optimization implemented by another optimizer and modifies/optimizes its proposed result. For example, line search might be applied to find a more optimal step size.

```
__init__ (optimizer, scope='meta-optimizer', summary_labels=(), **kwargs)
```

Creates a new meta optimizer instance.

Parameters **optimizer** – The optimizer which is modified by this meta optimizer.

```
apply_step (variables, deltas)
```

Applies step deltas to variable values.

Parameters

- **variables** – List of variables.
- **deltas** – List of deltas of same length.

Returns The step-applied operation.

```
from_spec (spec, kwargs=None)
```

Creates an optimizer from a specification dict.

```
get_summaries ()
```

Returns the TensorFlow summaries reported by the optimizer.

Returns List of summaries.

```
get_variables ()
```

```
minimize (time, variables, **kwargs)
```

Performs an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- ****kwargs** – Additional optimizer-specific arguments. The following arguments are used by some optimizers:
- **arguments** (–) – Dict of arguments for callables, like `fn_loss`.
- **fn_loss** (–) – A callable returning the loss of the current model.
- **fn_reference** (–) – A callable returning the reference values, in case of a comparative loss.
- **fn_kl_divergence** (–) – A callable returning the KL-divergence relative to the current model.
- **return_estimated_improvement** (–) – Returns the estimated improvement resulting from the natural gradient calculation if true.
- **source_variables** (–) – List of source variables to synchronize with.

- **global_variables** (-) – List of global variables to apply the proposed optimization step to.

Returns The optimization operation.

tf_step (*time*, *variables*, ***kwargs*)

Creates the TensorFlow operations for performing an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- ****kwargs** – Additional arguments depending on the specific optimizer implementation. For instance, often includes `fn_loss` if a loss function is optimized.

Returns List of delta tensors corresponding to the updates for each optimized variable.

tensorforce.core.optimizers.multi_step module

```
class tensorforce.core.optimizers.multi_step.MultiStep(optimizer, num_steps=10,
                                                         unroll_loop=False,
                                                         scope='multi-step', summary_labels=())
```

Bases: `tensorforce.core.optimizers.meta_optimizer.MetaOptimizer`

The multi-step meta optimizer repeatedly applies the optimization step proposed by another optimizer a number of times.

__init__ (*optimizer*, *num_steps=10*, *unroll_loop=False*, *scope='multi-step'*, *summary_labels=()*)

Creates a new multi-step meta optimizer instance.

Parameters

- **optimizer** – The optimizer which is modified by this meta optimizer.
- **num_steps** – Number of optimization steps to perform.

apply_step (*variables*, *deltas*)

Applies step deltas to variable values.

Parameters

- **variables** – List of variables.
- **deltas** – List of deltas of same length.

Returns The step-applied operation.

from_spec (*spec*, *kwargs=None*)

Creates an optimizer from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the optimizer.

Returns List of summaries.

get_variables ()

minimize (*time*, *variables*, ***kwargs*)

Performs an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- ****kwargs** – Additional optimizer-specific arguments. The following arguments are used by some optimizers:
- **arguments** (-) – Dict of arguments for callables, like `fn_loss`.
- **fn_loss** (-) – A callable returning the loss of the current model.
- **fn_reference** (-) – A callable returning the reference values, in case of a comparative loss.
- **fn_kl_divergence** (-) – A callable returning the KL-divergence relative to the current model.
- **return_estimated_improvement** (-) – Returns the estimated improvement resulting from the natural gradient calculation if true.
- **source_variables** (-) – List of source variables to synchronize with.
- **global_variables** (-) – List of global variables to apply the proposed optimization step to.

Returns The optimization operation.

tf_step (*time, variables, arguments, fn_reference=None, **kwargs*)
Creates the TensorFlow operations for performing an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- **arguments** – Dict of arguments for callables, like `fn_loss`.
- **fn_reference** – A callable returning the reference values, in case of a comparative loss.
- ****kwargs** – Additional arguments passed on to the internal optimizer.

Returns List of delta tensors corresponding to the updates for each optimized variable.

tensorforce.core.optimizers.natural_gradient module

```
class tensorforce.core.optimizers.natural_gradient.NaturalGradient (learning_rate,  
                                                                    cg_max_iterations=20,  
                                                                    cg_damping=0.001,  
                                                                    cg_unroll_loop=False,  
                                                                    scope='natural-  
                                                                    gradient',  
                                                                    sum-  
                                                                    mary_labels=())
```

Bases: `tensorforce.core.optimizers.optimizer.Optimizer`

Natural gradient optimizer.

```
__init__ (learning_rate, cg_max_iterations=20, cg_damping=0.001, cg_unroll_loop=False,  
          scope='natural-gradient', summary_labels=())  
Creates a new natural gradient optimizer instance.
```

Parameters

- **learning_rate** – Learning rate, i.e. KL-divergence of distributions between optimization steps.
- **cg_max_iterations** – Conjugate gradient solver max iterations.
- **cg_damping** – Conjugate gradient solver damping factor.
- **cg_unroll_loop** – Unroll conjugate gradient loop if true.

apply_step (*variables, deltas*)

Applies step deltas to variable values.

Parameters

- **variables** – List of variables.
- **deltas** – List of deltas of same length.

Returns The step-applied operation.

from_spec (*spec, kwargs=None*)

Creates an optimizer from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the optimizer.

Returns List of summaries.

get_variables ()

Returns the TensorFlow variables used by the optimizer.

Returns List of variables.

minimize (*time, variables, **kwargs*)

Performs an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- ****kwargs** – Additional optimizer-specific arguments. The following arguments are used by some optimizers:
- **arguments** (–) – Dict of arguments for callables, like `fn_loss`.
- **fn_loss** (–) – A callable returning the loss of the current model.
- **fn_reference** (–) – A callable returning the reference values, in case of a comparative loss.
- **fn_kl_divergence** (–) – A callable returning the KL-divergence relative to the current model.
- **return_estimated_improvement** (–) – Returns the estimated improvement resulting from the natural gradient calculation if true.
- **source_variables** (–) – List of source variables to synchronize with.
- **global_variables** (–) – List of global variables to apply the proposed optimization step to.

Returns The optimization operation.

tf_step (*time, variables, arguments, fn_loss, fn_kl_divergence, return_estimated_improvement=False, **kwargs*)

Creates the TensorFlow operations for performing an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- **arguments** – Dict of arguments for callables, like `fn_loss`.
- **fn_loss** – A callable returning the loss of the current model.
- **fn_kl_divergence** – A callable returning the KL-divergence relative to the current model.
- **return_estimated_improvement** – Returns the estimated improvement resulting from the natural gradient calculation if true.
- ****kwargs** – Additional arguments, not used.

Returns List of delta tensors corresponding to the updates for each optimized variable.

tensorforce.core.optimizers.optimized_step module

```
class tensorforce.core.optimizers.optimized_step.OptimizedStep(optimizer,
                                                                ls_max_iterations=10,
                                                                ls_accept_ratio=0.9,
                                                                ls_mode='exponential',
                                                                ls_parameter=0.5,
                                                                ls_unroll_loop=False,
                                                                scope='optimized-
step', summary_labels=())
```

Bases: `tensorforce.core.optimizers.meta_optimizer.MetaOptimizer`

The optimized-step meta optimizer applies line search to the proposed optimization step of another optimizer to find a more optimal step size.

```
__init__(optimizer, ls_max_iterations=10, ls_accept_ratio=0.9, ls_mode='exponential',
          ls_parameter=0.5, ls_unroll_loop=False, scope='optimized-step', summary_labels=())
```

Creates a new optimized step meta optimizer instance.

Parameters

- **optimizer** – The optimizer which is modified by this meta optimizer.
- **ls_max_iterations** – Maximum number of line search iterations.
- **ls_accept_ratio** – Line search acceptance ratio.
- **ls_mode** – Line search mode, see `LineSearch` solver.
- **ls_parameter** – Line search parameter, see `LineSearch` solver.
- **ls_unroll_loop** – Unroll line search loop if true.

```
apply_step(variables, deltas)
```

Applies step deltas to variable values.

Parameters

- **variables** – List of variables.
- **deltas** – List of deltas of same length.

Returns The step-applied operation.

from_spec (*spec*, *kwargs*=None)

Creates an optimizer from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the optimizer.

Returns List of summaries.

get_variables ()

minimize (*time*, *variables*, ***kwargs*)

Performs an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- ****kwargs** – Additional optimizer-specific arguments. The following arguments are used by some optimizers:
- **arguments** (-) – Dict of arguments for callables, like `fn_loss`.
- **fn_loss** (-) – A callable returning the loss of the current model.
- **fn_reference** (-) – A callable returning the reference values, in case of a comparative loss.
- **fn_kl_divergence** (-) – A callable returning the KL-divergence relative to the current model.
- **return_estimated_improvement** (-) – Returns the estimated improvement resulting from the natural gradient calculation if true.
- **source_variables** (-) – List of source variables to synchronize with.
- **global_variables** (-) – List of global variables to apply the proposed optimization step to.

Returns The optimization operation.

tf_step (*time*, *variables*, *arguments*, *fn_loss*, *fn_reference*, ***kwargs*)

Creates the TensorFlow operations for performing an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- **arguments** – Dict of arguments for callables, like `fn_loss`.
- **fn_loss** – A callable returning the loss of the current model.
- **fn_reference** – A callable returning the reference values, in case of a comparative loss.
- ****kwargs** – Additional arguments passed on to the internal optimizer.

Returns List of delta tensors corresponding to the updates for each optimized variable.

tensorflow.core.optimizers.optimizer module

```
class tensorflow.core.optimizers.optimizer.Optimizer (scope='optimizer',      sum-  
mary_labels=None)
```

Bases: object

Base class for optimizers which minimize a not yet further specified expression, usually some kind of loss function. More generally, an optimizer can be considered as some method of updating a set of variables.

```
__init__ (scope='optimizer', summary_labels=None)  
Creates a new optimizer instance.
```

```
apply_step (variables, deltas)  
Applies step deltas to variable values.
```

Parameters

- **variables** – List of variables.
- **deltas** – List of deltas of same length.

Returns The step-applied operation.

```
static from_spec (spec, kwargs=None)  
Creates an optimizer from a specification dict.
```

```
get_summaries ()  
Returns the TensorFlow summaries reported by the optimizer.
```

Returns List of summaries.

```
get_variables ()  
Returns the TensorFlow variables used by the optimizer.
```

Returns List of variables.

```
minimize (time, variables, **kwargs)  
Performs an optimization step.
```

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- ****kwargs** – Additional optimizer-specific arguments. The following arguments are used by some optimizers:
- **arguments** (-) – Dict of arguments for callables, like `fn_loss`.
- **fn_loss** (-) – A callable returning the loss of the current model.
- **fn_reference** (-) – A callable returning the reference values, in case of a comparative loss.
- **fn_kl_divergence** (-) – A callable returning the KL-divergence relative to the current model.
- **return_estimated_improvement** (-) – Returns the estimated improvement resulting from the natural gradient calculation if true.
- **source_variables** (-) – List of source variables to synchronize with.
- **global_variables** (-) – List of global variables to apply the proposed optimization step to.

Returns The optimization operation.

tf_step (*time*, *variables*, ***kwargs*)

Creates the TensorFlow operations for performing an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- ****kwargs** – Additional arguments depending on the specific optimizer implementation. For instance, often includes `fn_loss` if a loss function is optimized.

Returns List of delta tensors corresponding to the updates for each optimized variable.

tensorforce.core.optimizers.synchronization module

```
class tensorforce.core.optimizers.synchronization.Synchronization (sync_frequency=1,
                                                                update_weight=1.0,
                                                                scope='synchronization',
                                                                summary_labels=())
```

Bases: `tensorforce.core.optimizers.optimizer.Optimizer`

The synchronization optimizer updates variables periodically to the value of a corresponding set of source variables.

__init__ (*sync_frequency=1*, *update_weight=1.0*, *scope='synchronization'*, *summary_labels=()*)

Creates a new synchronization optimizer instance.

Parameters

- **sync_frequency** – The interval between optimization calls actually performing a
- **step.** (*synchronization*) –
- **update_weight** – The update weight, 1.0 meaning a full assignment of the source
- **values.** (*variables*) –

apply_step (*variables*, *deltas*)

Applies step deltas to variable values.

Parameters

- **variables** – List of variables.
- **deltas** – List of deltas of same length.

Returns The step-applied operation.

from_spec (*spec*, *kwargs=None*)

Creates an optimizer from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the optimizer.

Returns List of summaries.

get_variables ()

Returns the TensorFlow variables used by the optimizer.

Returns List of variables.

minimize (*time*, *variables*, ***kwargs*)

Performs an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- ****kwargs** – Additional optimizer-specific arguments. The following arguments are used by some optimizers:
- **arguments** (-) – Dict of arguments for callables, like `fn_loss`.
- **fn_loss** (-) – A callable returning the loss of the current model.
- **fn_reference** (-) – A callable returning the reference values, in case of a comparative loss.
- **fn_kl_divergence** (-) – A callable returning the KL-divergence relative to the current model.
- **return_estimated_improvement** (-) – Returns the estimated improvement resulting from the natural gradient calculation if true.
- **source_variables** (-) – List of source variables to synchronize with.
- **global_variables** (-) – List of global variables to apply the proposed optimization step to.

Returns The optimization operation.

tf_step (*time*, *variables*, *source_variables*, ***kwargs*)

Creates the TensorFlow operations for performing an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- **source_variables** – List of source variables to synchronize with.
- ****kwargs** – Additional arguments, not used.

Returns List of delta tensors corresponding to the updates for each optimized variable.

tensorflow.core.optimizers.tf_optimizer module

```
class tensorflow.core.optimizers.tf_optimizer.TFOptimizer (optimizer,  
                                                         scope=None, summary_labels=(),  
                                                         **kwargs)
```

Bases: `tensorflow.core.optimizers.optimizer.Optimizer`

Wrapper class for TensorFlow optimizers.

```
__init__ (optimizer, scope=None, summary_labels=(), **kwargs)
```

Creates a new optimizer instance of a TensorFlow optimizer.

Parameters

- **optimizer** – The name of the optimizer, one of ‘adadelta’, ‘adagrad’, ‘adam’, ‘nadam’, ‘momentum’, ‘rmsprop’. (‘gradient_descent’,) –

- ****kwargs** – Additional arguments passed on to the TensorFlow optimizer constructor.

apply_step (*variables, deltas*)

Applies step deltas to variable values.

Parameters

- **variables** – List of variables.
- **deltas** – List of deltas of same length.

Returns The step-applied operation.

from_spec (*spec, kwargs=None*)

Creates an optimizer from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the optimizer.

Returns List of summaries.

get_variables ()

static get_wrapper (*optimizer*)

Returns a TFOptimizer constructor callable for the given optimizer name.

Parameters

- **optimizer** – The name of the optimizer, one of ‘adadelata’, ‘adagrad’, ‘adam’, ‘nadam’, ‘momentum’, ‘rmsprop’. (‘gradient_descent’,) –

Returns The TFOptimizer constructor callable.

minimize (*time, variables, **kwargs*)

Performs an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- ****kwargs** – Additional optimizer-specific arguments. The following arguments are used by some optimizers:
- **arguments** (–) – Dict of arguments for callables, like `fn_loss`.
- **fn_loss** (–) – A callable returning the loss of the current model.
- **fn_reference** (–) – A callable returning the reference values, in case of a comparative loss.
- **fn_kl_divergence** (–) – A callable returning the KL-divergence relative to the current model.
- **return_estimated_improvement** (–) – Returns the estimated improvement resulting from the natural gradient calculation if true.
- **source_variables** (–) – List of source variables to synchronize with.
- **global_variables** (–) – List of global variables to apply the proposed optimization step to.

Returns The optimization operation.

`tf_optimizers = {'nadam': <sphinx.ext.autodoc._MockObject object>, 'adam': <sphinx.e`

tf_step (*time, variables, arguments, fn_loss, **kwargs*)

Creates the TensorFlow operations for performing an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- **arguments** – Dict of arguments for callables, like `fn_loss`.
- **fn_loss** – A callable returning the loss of the current model.
- ****kwargs** – Additional arguments, not used.

Returns List of delta tensors corresponding to the updates for each optimized variable.

Module contents

class `tensorforce.core.optimizers.Optimizer` (*scope='optimizer', summary_labels=None*)

Bases: `object`

Base class for optimizers which minimize a not yet further specified expression, usually some kind of loss function. More generally, an optimizer can be considered as some method of updating a set of variables.

__init__ (*scope='optimizer', summary_labels=None*)

Creates a new optimizer instance.

apply_step (*variables, deltas*)

Applies step deltas to variable values.

Parameters

- **variables** – List of variables.
- **deltas** – List of deltas of same length.

Returns The step-applied operation.

static from_spec (*spec, kwargs=None*)

Creates an optimizer from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the optimizer.

Returns List of summaries.

get_variables ()

Returns the TensorFlow variables used by the optimizer.

Returns List of variables.

minimize (*time, variables, **kwargs*)

Performs an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- ****kwargs** – Additional optimizer-specific arguments. The following arguments are used by some optimizers:
- **arguments** (–) – Dict of arguments for callables, like `fn_loss`.

- **fn_loss** (-) – A callable returning the loss of the current model.
- **fn_reference** (-) – A callable returning the reference values, in case of a comparative loss.
- **fn_kl_divergence** (-) – A callable returning the KL-divergence relative to the current model.
- **return_estimated_improvement** (-) – Returns the estimated improvement resulting from the natural gradient calculation if true.
- **source_variables** (-) – List of source variables to synchronize with.
- **global_variables** (-) – List of global variables to apply the proposed optimization step to.

Returns The optimization operation.

tf_step (*time, variables, **kwargs*)

Creates the TensorFlow operations for performing an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- ****kwargs** – Additional arguments depending on the specific optimizer implementation. For instance, often includes `fn_loss` if a loss function is optimized.

Returns List of delta tensors corresponding to the updates for each optimized variable.

class `tensorforce.core.optimizers.MetaOptimizer` (*optimizer, scope='meta-optimizer', summary_labels=(), **kwargs*)

Bases: `tensorforce.core.optimizers.optimizer.Optimizer`

A meta optimizer takes the optimization implemented by another optimizer and modifies/optimizes its proposed result. For example, line search might be applied to find a more optimal step size.

__init__ (*optimizer, scope='meta-optimizer', summary_labels=(), **kwargs*)

Creates a new meta optimizer instance.

Parameters **optimizer** – The optimizer which is modified by this meta optimizer.

apply_step (*variables, deltas*)

Applies step deltas to variable values.

Parameters

- **variables** – List of variables.
- **deltas** – List of deltas of same length.

Returns The step-applied operation.

from_spec (*spec, kwargs=None*)

Creates an optimizer from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the optimizer.

Returns List of summaries.

get_variables ()

minimize (*time, variables, **kwargs*)

Performs an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- ****kwargs** – Additional optimizer-specific arguments. The following arguments are used by some optimizers:
- **arguments** (-) – Dict of arguments for callables, like `fn_loss`.
- **fn_loss** (-) – A callable returning the loss of the current model.
- **fn_reference** (-) – A callable returning the reference values, in case of a comparative loss.
- **fn_kl_divergence** (-) – A callable returning the KL-divergence relative to the current model.
- **return_estimated_improvement** (-) – Returns the estimated improvement resulting from the natural gradient calculation if true.
- **source_variables** (-) – List of source variables to synchronize with.
- **global_variables** (-) – List of global variables to apply the proposed optimization step to.

Returns The optimization operation.

tf_step (*time, variables, **kwargs*)

Creates the TensorFlow operations for performing an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- ****kwargs** – Additional arguments depending on the specific optimizer implementation. For instance, often includes `fn_loss` if a loss function is optimized.

Returns List of delta tensors corresponding to the updates for each optimized variable.

```
class tensorflow.core.optimizers.GlobalOptimizer (optimizer, scope='global-optimizer', summary_labels=())
```

Bases: `tensorflow.core.optimizers.meta_optimizer.MetaOptimizer`

The global optimizer applies an optimizer to the local variables. In addition, it also applies the update to a corresponding set of global variables and subsequently updates the local variables to the value of these global variables. Note: This is used for the current distributed mode, and will likely change with the next major version update.

```
__init__ (optimizer, scope='global-optimizer', summary_labels=())
```

Creates a new global optimizer instance.

Parameters **optimizer** – The optimizer which is modified by this meta optimizer.

apply_step (*variables, deltas*)

Applies step deltas to variable values.

Parameters

- **variables** – List of variables.
- **deltas** – List of deltas of same length.

Returns The step-applied operation.

from_spec (*spec*, *kwargs=None*)

Creates an optimizer from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the optimizer.

Returns List of summaries.

get_variables ()

minimize (*time*, *variables*, ***kwargs*)

Performs an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- ****kwargs** – Additional optimizer-specific arguments. The following arguments are used by some optimizers:
- **arguments** (-) – Dict of arguments for callables, like `fn_loss`.
- **fn_loss** (-) – A callable returning the loss of the current model.
- **fn_reference** (-) – A callable returning the reference values, in case of a comparative loss.
- **fn_kl_divergence** (-) – A callable returning the KL-divergence relative to the current model.
- **return_estimated_improvement** (-) – Returns the estimated improvement resulting from the natural gradient calculation if true.
- **source_variables** (-) – List of source variables to synchronize with.
- **global_variables** (-) – List of global variables to apply the proposed optimization step to.

Returns The optimization operation.

tf_step (*time*, *variables*, *global_variables*, ***kwargs*)

Creates the TensorFlow operations for performing an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- **global_variables** – List of global variables to apply the proposed optimization step to.
- ****kwargs** – ??? coming soon

Returns List of delta tensors corresponding to the updates for each optimized variable.

class `tensorforce.core.optimizers.TFOptimizer` (*optimizer*, *scope=None*, *summary_labels=()*, ***kwargs*)

Bases: `tensorforce.core.optimizers.optimizer.Optimizer`

Wrapper class for TensorFlow optimizers.

__init__ (*optimizer*, *scope=None*, *summary_labels=()*, ***kwargs*)

Creates a new optimizer instance of a TensorFlow optimizer.

Parameters

- **optimizer** – The name of the optimizer, one of ‘adadelat’, ‘adagrad’, ‘adam’, ‘nadam’,
- ‘momentum’, ‘rmsprop’. (*‘gradient_descent’*,) –
- ****kwargs** – Additional arguments passed on to the TensorFlow optimizer constructor.

apply_step (*variables, deltas*)

Applies step deltas to variable values.

Parameters

- **variables** – List of variables.
- **deltas** – List of deltas of same length.

Returns The step-applied operation.

from_spec (*spec, kwargs=None*)

Creates an optimizer from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the optimizer.

Returns List of summaries.

get_variables ()

static get_wrapper (*optimizer*)

Returns a TFOptimizer constructor callable for the given optimizer name.

Parameters

- **optimizer** – The name of the optimizer, one of ‘adadelat’, ‘adagrad’, ‘adam’, ‘nadam’,
- ‘momentum’, ‘rmsprop’. (*‘gradient_descent’*,) –

Returns The TFOptimizer constructor callable.

minimize (*time, variables, **kwargs*)

Performs an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- ****kwargs** – Additional optimizer-specific arguments. The following arguments are used by some optimizers:
- **arguments** (–) – Dict of arguments for callables, like `fn_loss`.
- **fn_loss** (–) – A callable returning the loss of the current model.
- **fn_reference** (–) – A callable returning the reference values, in case of a comparative loss.
- **fn_kl_divergence** (–) – A callable returning the KL-divergence relative to the current model.
- **return_estimated_improvement** (–) – Returns the estimated improvement resulting from the natural gradient calculation if true.
- **source_variables** (–) – List of source variables to synchronize with.
- **global_variables** (–) – List of global variables to apply the proposed optimization step to.

Returns The optimization operation.

tf_optimizers = {'nadam': <sphinx.ext.autodoc._MockObject object>, 'adam': <sphinx.e

tf_step (*time, variables, arguments, fn_loss, **kwargs*)

Creates the TensorFlow operations for performing an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- **arguments** – Dict of arguments for callables, like `fn_loss`.
- **fn_loss** – A callable returning the loss of the current model.
- ****kwargs** – Additional arguments, not used.

Returns List of delta tensors corresponding to the updates for each optimized variable.

class `tensorforce.core.optimizers.Evolutionary` (*learning_rate, num_samples=1, unroll_loop=False, scope='evolutionary', summary_labels=()*)

Bases: `tensorforce.core.optimizers.optimizer.Optimizer`

Evolutionary optimizer which samples random perturbations and applies them either positively or negatively, depending on their improvement of the loss.

__init__ (*learning_rate, num_samples=1, unroll_loop=False, scope='evolutionary', summary_labels=()*)

Creates a new evolutionary optimizer instance.

Parameters

- **learning_rate** – Learning rate.
- **num_samples** – Number of sampled perturbations.

apply_step (*variables, deltas*)

Applies step deltas to variable values.

Parameters

- **variables** – List of variables.
- **deltas** – List of deltas of same length.

Returns The step-applied operation.

from_spec (*spec, kwargs=None*)

Creates an optimizer from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the optimizer.

Returns List of summaries.

get_variables ()

Returns the TensorFlow variables used by the optimizer.

Returns List of variables.

minimize (*time, variables, **kwargs*)

Performs an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- ****kwargs** – Additional optimizer-specific arguments. The following arguments are used by some optimizers:
- **arguments** (-) – Dict of arguments for callables, like `fn_loss`.
- **fn_loss** (-) – A callable returning the loss of the current model.
- **fn_reference** (-) – A callable returning the reference values, in case of a comparative loss.
- **fn_kl_divergence** (-) – A callable returning the KL-divergence relative to the current model.
- **return_estimated_improvement** (-) – Returns the estimated improvement resulting from the natural gradient calculation if true.
- **source_variables** (-) – List of source variables to synchronize with.
- **global_variables** (-) – List of global variables to apply the proposed optimization step to.

Returns The optimization operation.

tf_step (*time, variables, arguments, fn_loss, **kwargs*)

Creates the TensorFlow operations for performing an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- **arguments** – Dict of arguments for callables, like `fn_loss`.
- **fn_loss** – A callable returning the loss of the current model.
- ****kwargs** – Additional arguments, not used.

Returns List of delta tensors corresponding to the updates for each optimized variable.

```
class tensorflow.core.optimizers.NaturalGradient (learning_rate,  
                                                cg_max_iterations=20,  
                                                cg_damping=0.001,  
                                                cg_unroll_loop=False,  
                                                scope='natural-gradient',    summary_labels=())
```

Bases: `tensorflow.core.optimizers.optimizer.Optimizer`

Natural gradient optimizer.

```
__init__ (learning_rate,    cg_max_iterations=20,    cg_damping=0.001,    cg_unroll_loop=False,  
          scope='natural-gradient',    summary_labels=())
```

Creates a new natural gradient optimizer instance.

Parameters

- **learning_rate** – Learning rate, i.e. KL-divergence of distributions between optimization steps.
- **cg_max_iterations** – Conjugate gradient solver max iterations.
- **cg_damping** – Conjugate gradient solver damping factor.

- **cg_unroll_loop** – Unroll conjugate gradient loop if true.

apply_step (*variables*, *deltas*)

Applies step deltas to variable values.

Parameters

- **variables** – List of variables.
- **deltas** – List of deltas of same length.

Returns The step-applied operation.

from_spec (*spec*, *kwargs*=None)

Creates an optimizer from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the optimizer.

Returns List of summaries.

get_variables ()

Returns the TensorFlow variables used by the optimizer.

Returns List of variables.

minimize (*time*, *variables*, ***kwargs*)

Performs an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- ****kwargs** – Additional optimizer-specific arguments. The following arguments are used by some optimizers:
- **arguments** (–) – Dict of arguments for callables, like `fn_loss`.
- **fn_loss** (–) – A callable returning the loss of the current model.
- **fn_reference** (–) – A callable returning the reference values, in case of a comparative loss.
- **fn_kl_divergence** (–) – A callable returning the KL-divergence relative to the current model.
- **return_estimated_improvement** (–) – Returns the estimated improvement resulting from the natural gradient calculation if true.
- **source_variables** (–) – List of source variables to synchronize with.
- **global_variables** (–) – List of global variables to apply the proposed optimization step to.

Returns The optimization operation.

tf_step (*time*, *variables*, *arguments*, *fn_loss*, *fn_kl_divergence*, *return_estimated_improvement*=False, ***kwargs*)

Creates the TensorFlow operations for performing an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.

- **arguments** – Dict of arguments for callables, like `fn_loss`.
- **fn_loss** – A callable returning the loss of the current model.
- **fn_kl_divergence** – A callable returning the KL-divergence relative to the current model.
- **return_estimated_improvement** – Returns the estimated improvement resulting from the natural gradient calculation if true.
- ****kwargs** – Additional arguments, not used.

Returns List of delta tensors corresponding to the updates for each optimized variable.

```
class tensorflow.core.optimizers.ClippedStep(optimizer, clipping_value,
                                             scope='clipped-step', summary_labels=())
```

Bases: `tensorflow.core.optimizers.meta_optimizer.MetaOptimizer`

The clipped-shep meta optimizer clips the values of the optimization step proposed by another optimizer.

```
__init__(optimizer, clipping_value, scope='clipped-step', summary_labels=())
    Creates a new multi-step meta optimizer instance.
```

Parameters

- **optimizer** – The optimizer which is modified by this meta optimizer.
- **clipping_value** – Clip deltas at this value.

```
apply_step(variables, deltas)
    Applies step deltas to variable values.
```

Parameters

- **variables** – List of variables.
- **deltas** – List of deltas of same length.

Returns The step-applied operation.

```
from_spec(spec, kwargs=None)
    Creates an optimizer from a specification dict.
```

```
get_summaries()
    Returns the TensorFlow summaries reported by the optimizer.
```

Returns List of summaries.

```
get_variables()
```

```
minimize(time, variables, **kwargs)
    Performs an optimization step.
```

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- ****kwargs** – Additional optimizer-specific arguments. The following arguments are used by some optimizers:
- **arguments** (–) – Dict of arguments for callables, like `fn_loss`.
- **fn_loss** (–) – A callable returning the loss of the current model.

- **fn_reference** (-) – A callable returning the reference values, in case of a comparative loss.
- **fn_kl_divergence** (-) – A callable returning the KL-divergence relative to the current model.
- **return_estimated_improvement** (-) – Returns the estimated improvement resulting from the natural gradient calculation if true.
- **source_variables** (-) – List of source variables to synchronize with.
- **global_variables** (-) – List of global variables to apply the proposed optimization step to.

Returns The optimization operation.

tf_step (*time, variables, **kwargs*)

Creates the TensorFlow operations for performing an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- ****kwargs** – Additional arguments passed on to the internal optimizer.

Returns List of delta tensors corresponding to the updates for each optimized variable.

```
class tensorflow.core.optimizers.MultiStep (optimizer, num_steps=10, unroll_loop=False, scope='multi-step', summary_labels=())
```

Bases: *tensorflow.core.optimizers.meta_optimizer.MetaOptimizer*

The multi-step meta optimizer repeatedly applies the optimization step proposed by another optimizer a number of times.

__init__ (*optimizer, num_steps=10, unroll_loop=False, scope='multi-step', summary_labels=()*)

Creates a new multi-step meta optimizer instance.

Parameters

- **optimizer** – The optimizer which is modified by this meta optimizer.
- **num_steps** – Number of optimization steps to perform.

apply_step (*variables, deltas*)

Applies step deltas to variable values.

Parameters

- **variables** – List of variables.
- **deltas** – List of deltas of same length.

Returns The step-applied operation.

from_spec (*spec, kwargs=None*)

Creates an optimizer from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the optimizer.

Returns List of summaries.

get_variables ()

minimize (*time*, *variables*, ***kwargs*)

Performs an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- ****kwargs** – Additional optimizer-specific arguments. The following arguments are used by some optimizers:
- **arguments** (-) – Dict of arguments for callables, like `fn_loss`.
- **fn_loss** (-) – A callable returning the loss of the current model.
- **fn_reference** (-) – A callable returning the reference values, in case of a comparative loss.
- **fn_kl_divergence** (-) – A callable returning the KL-divergence relative to the current model.
- **return_estimated_improvement** (-) – Returns the estimated improvement resulting from the natural gradient calculation if true.
- **source_variables** (-) – List of source variables to synchronize with.
- **global_variables** (-) – List of global variables to apply the proposed optimization step to.

Returns The optimization operation.

tf_step (*time*, *variables*, *arguments*, *fn_reference=None*, ***kwargs*)

Creates the TensorFlow operations for performing an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- **arguments** – Dict of arguments for callables, like `fn_loss`.
- **fn_reference** – A callable returning the reference values, in case of a comparative loss.
- ****kwargs** – Additional arguments passed on to the internal optimizer.

Returns List of delta tensors corresponding to the updates for each optimized variable.

```
class tensorflow.core.optimizers.OptimizedStep (optimizer, ls_max_iterations=10,
                                                ls_accept_ratio=0.9,
                                                ls_mode='exponential',
                                                ls_parameter=0.5,
                                                ls_unroll_loop=False,
                                                scope='optimized-step', summary_labels=())
```

Bases: `tensorflow.core.optimizers.meta_optimizer.MetaOptimizer`

The optimized-step meta optimizer applies line search to the proposed optimization step of another optimizer to find a more optimal step size.

```
__init__ (optimizer, ls_max_iterations=10, ls_accept_ratio=0.9, ls_mode='exponential',
          ls_parameter=0.5, ls_unroll_loop=False, scope='optimized-step', summary_labels=())
```

Creates a new optimized step meta optimizer instance.

Parameters

- **optimizer** – The optimizer which is modified by this meta optimizer.
- **ls_max_iterations** – Maximum number of line search iterations.
- **ls_accept_ratio** – Line search acceptance ratio.
- **ls_mode** – Line search mode, see LineSearch solver.
- **ls_parameter** – Line search parameter, see LineSearch solver.
- **ls_unroll_loop** – Unroll line search loop if true.

apply_step (*variables*, *deltas*)

Applies step deltas to variable values.

Parameters

- **variables** – List of variables.
- **deltas** – List of deltas of same length.

Returns The step-applied operation.

from_spec (*spec*, *kwargs*=None)

Creates an optimizer from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the optimizer.

Returns List of summaries.

get_variables ()

minimize (*time*, *variables*, ***kwargs*)

Performs an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- ****kwargs** – Additional optimizer-specific arguments. The following arguments are used by some optimizers:
- **arguments** (–) – Dict of arguments for callables, like `fn_loss`.
- **fn_loss** (–) – A callable returning the loss of the current model.
- **fn_reference** (–) – A callable returning the reference values, in case of a comparative loss.
- **fn_kl_divergence** (–) – A callable returning the KL-divergence relative to the current model.
- **return_estimated_improvement** (–) – Returns the estimated improvement resulting from the natural gradient calculation if true.
- **source_variables** (–) – List of source variables to synchronize with.
- **global_variables** (–) – List of global variables to apply the proposed optimization step to.

Returns The optimization operation.

tf_step (*time, variables, arguments, fn_loss, fn_reference, **kwargs*)
Creates the TensorFlow operations for performing an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- **arguments** – Dict of arguments for callables, like `fn_loss`.
- **fn_loss** – A callable returning the loss of the current model.
- **fn_reference** – A callable returning the reference values, in case of a comparative loss.
- ****kwargs** – Additional arguments passed on to the internal optimizer.

Returns List of delta tensors corresponding to the updates for each optimized variable.

class `tensorforce.core.optimizers.SubsamplingStep` (*optimizer, fraction=0.1, scope='subsampling-step', summary_labels=()*)

Bases: `tensorforce.core.optimizers.meta_optimizer.MetaOptimizer`

The subsampling-step meta optimizer randomly samples a subset of batch instances to calculate the optimization step of another optimizer.

__init__ (*optimizer, fraction=0.1, scope='subsampling-step', summary_labels=()*)
Creates a new subsampling-step meta optimizer instance.

Parameters

- **optimizer** – The optimizer which is modified by this meta optimizer.
- **fraction** – The fraction of instances of the batch to subsample.

apply_step (*variables, deltas*)
Applies step deltas to variable values.

Parameters

- **variables** – List of variables.
- **deltas** – List of deltas of same length.

Returns The step-applied operation.

from_spec (*spec, kwargs=None*)
Creates an optimizer from a specification dict.

get_summaries ()
Returns the TensorFlow summaries reported by the optimizer.

Returns List of summaries.

get_variables ()

minimize (*time, variables, **kwargs*)
Performs an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.

- ****kwargs** – Additional optimizer-specific arguments. The following arguments are used by some optimizers:
- **arguments** (-) – Dict of arguments for callables, like `fn_loss`.
- **fn_loss** (-) – A callable returning the loss of the current model.
- **fn_reference** (-) – A callable returning the reference values, in case of a comparative loss.
- **fn_kl_divergence** (-) – A callable returning the KL-divergence relative to the current model.
- **return_estimated_improvement** (-) – Returns the estimated improvement resulting from the natural gradient calculation if true.
- **source_variables** (-) – List of source variables to synchronize with.
- **global_variables** (-) – List of global variables to apply the proposed optimization step to.

Returns The optimization operation.

tf_step (*time, variables, arguments, **kwargs*)

Creates the TensorFlow operations for performing an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- **arguments** – Dict of arguments for callables, like `fn_loss`.
- ****kwargs** – Additional arguments passed on to the internal optimizer.

Returns List of delta tensors corresponding to the updates for each optimized variable.

```
class tensorflow.core.optimizers.Synchronization (sync_frequency=1,
                                                update_weight=1.0,
                                                scope='synchronization',      summary_labels=())
```

Bases: `tensorflow.core.optimizers.optimizer.Optimizer`

The synchronization optimizer updates variables periodically to the value of a corresponding set of source variables.

__init__ (*sync_frequency=1, update_weight=1.0, scope='synchronization', summary_labels=()*)

Creates a new synchronization optimizer instance.

Parameters

- **sync_frequency** – The interval between optimization calls actually performing a
- **step.** (*synchronization*) –
- **update_weight** – The update weight, 1.0 meaning a full assignment of the source
- **values.** (*variables*) –

apply_step (*variables, deltas*)

Applies step deltas to variable values.

Parameters

- **variables** – List of variables.
- **deltas** – List of deltas of same length.

Returns The step-applied operation.

from_spec (*spec*, *kwargs*=None)

Creates an optimizer from a specification dict.

get_summaries ()

Returns the TensorFlow summaries reported by the optimizer.

Returns List of summaries.

get_variables ()

Returns the TensorFlow variables used by the optimizer.

Returns List of variables.

minimize (*time*, *variables*, ***kwargs*)

Performs an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- ****kwargs** – Additional optimizer-specific arguments. The following arguments are used by some optimizers:
- **arguments** (-) – Dict of arguments for callables, like `fn_loss`.
- **fn_loss** (-) – A callable returning the loss of the current model.
- **fn_reference** (-) – A callable returning the reference values, in case of a comparative loss.
- **fn_kl_divergence** (-) – A callable returning the KL-divergence relative to the current model.
- **return_estimated_improvement** (-) – Returns the estimated improvement resulting from the natural gradient calculation if true.
- **source_variables** (-) – List of source variables to synchronize with.
- **global_variables** (-) – List of global variables to apply the proposed optimization step to.

Returns The optimization operation.

tf_step (*time*, *variables*, *source_variables*, ***kwargs*)

Creates the TensorFlow operations for performing an optimization step.

Parameters

- **time** – Time tensor.
- **variables** – List of variables to optimize.
- **source_variables** – List of source variables to synchronize with.
- ****kwargs** – Additional arguments, not used.

Returns List of delta tensors corresponding to the updates for each optimized variable.

tensorforce.core.preprocessing package

Submodules

tensorforce.core.preprocessing.clip module

tensorforce.core.preprocessing.divide module

tensorforce.core.preprocessing.grayscale module

tensorforce.core.preprocessing.image_resize module

tensorforce.core.preprocessing.normalize module

tensorforce.core.preprocessing.preprocessor module

tensorforce.core.preprocessing.preprocessor_stack module

tensorforce.core.preprocessing.running_standardize module

tensorforce.core.preprocessing.sequence module

tensorforce.core.preprocessing.standardize module

Module contents

Module contents

tensorforce.environments package

Submodules

tensorforce.environments.environment module

class tensorforce.environments.environment.**Environment**

Bases: object

Base environment class.

__init__

x.init(...) initializes x; see help(type(x)) for signature

actions

Return the action space. Might include subdicts if multiple actions are available simultaneously.

Returns: dict of action properties (continuous, number of actions)

close()

Close environment. No other method calls possible afterwards.

execute(actions)

Executes action, observes next state(s) and reward.

Parameters **actions** – Actions to execute.

Returns (Dict of) next state(s), boolean indicating terminal, and reward signal.

static from_spec (*spec, kwargs*)

Creates an environment from a specification dict.

reset ()

Reset environment and setup for new episode.

Returns initial state of reset environment.

seed (*seed*)

Sets the random seed of the environment to the given value (current time, if seed=None). Naturally deterministic Environments (e.g. ALE or some gym Envs) don't have to implement this method.

Parameters **seed** (*int*) – The seed to use for initializing the pseudo-random number generator (default=epoch time in sec).

Returns: The actual seed (int) used OR None if Environment did not override this method (no seeding supported).

states

Return the state space. Might include subdicts if multiple states are available simultaneously.

Returns: dict of state properties (shape and type).

tensorforce.environments.minimal_test module

Module contents

class tensorforce.environments.Environment

Bases: object

Base environment class.

__init__

x.**init**(...) initializes x; see help(type(x)) for signature

actions

Return the action space. Might include subdicts if multiple actions are available simultaneously.

Returns: dict of action properties (continuous, number of actions)

close ()

Close environment. No other method calls possible afterwards.

execute (*actions*)

Executes action, observes next state(s) and reward.

Parameters **actions** – Actions to execute.

Returns (Dict of) next state(s), boolean indicating terminal, and reward signal.

static from_spec (*spec, kwargs*)

Creates an environment from a specification dict.

reset ()

Reset environment and setup for new episode.

Returns initial state of reset environment.

seed (*seed*)

Sets the random seed of the environment to the given value (current time, if seed=None). Naturally deterministic Environments (e.g. ALE or some gym Envs) don't have to implement this method.

Parameters **seed** (*int*) – The seed to use for initializing the pseudo-random number generator (default=epoch time in sec).

Returns: The actual seed (int) used OR None if Environment did not override this method (no seeding supported).

states

Return the state space. Might include subdicts if multiple states are available simultaneously.

Returns: dict of state properties (shape and type).

class `tensorforce.environments.MinimalTest` (*specification*)

Bases: `tensorforce.environments.environment.Environment`

__init__ (*specification*)

Initializes a minimal test environment, which is used for the unit tests. Given a specification of actions types and shapes, the environment states consist of the same number of pairs (x, y). The (mean of) an action gives the next state via (1-a, a), and the 'correct' state is always (0, 1).

Parameters **specification** – Takes a dict type (keys)-> shape (values specifying the action structure of the environment. Use shape () for single scalar actions.

actions**close** ()**execute** (*actions*)**from_spec** (*spec, kwargs*)

Creates an environment from a specification dict.

reset ()**seed** (*seed*)

Sets the random seed of the environment to the given value (current time, if seed=None). Naturally deterministic Environments (e.g. ALE or some gym Envs) don't have to implement this method.

Parameters **seed** (*int*) – The seed to use for initializing the pseudo-random number generator (default=epoch time in sec).

Returns: The actual seed (int) used OR None if Environment did not override this method (no seeding supported).

states**tensorforce.execution package****Submodules****tensorforce.execution.runner module**

`tensorforce.execution.runner.DistributedTFRunner`

alias of `Runner`

class `tensorforce.execution.runner.Runner` (*agent, environment, repeat_actions=1, history=None, id_=0*)

Bases: `tensorforce.execution.base_runner.BaseRunner`

Simple runner for non-realtime single-process execution.

__init__ (*agent, environment, repeat_actions=1, history=None, id_=0*)

Initialize a single Runner object (one Agent/one Environment).

Parameters **id*** – The ID of this Runner (for distributed TF runs).

:type id*: int

close ()

episode

Deprecated property episode -> global_episode.

episode_timestep

reset (*history=None*)

Resets the Runner's internal stats counters. If history is empty, use default values in history.get().

Parameters **history** (*dict*) – A dictionary containing an already run experiment's results.

Keys should be: episode_rewards (list of rewards), episode_timesteps (lengths of episodes), episode_times (run-times)

run (*num_timesteps=None, num_episodes=None, max_episode_timesteps=None, deterministic=False, episode_finished=None, summary_report=None, summary_interval=None, timesteps=None, episodes=None*)

Parameters

- **timesteps** (*int*) – Deprecated; see num_timesteps.
- **episodes** (*int*) – Deprecated; see num_episodes.

timestep

Deprecated property timestep -> global_timestep.

tensorflow.execution.runner.**SingleRunner**

alias of *Runner*

tensorflow.execution.threaded_runner module

```
class tensorflow.execution.threaded_runner.ThreadedRunner (agent, environment,  
                                                         repeat_actions=1,  
                                                         save_path=None,  
                                                         save_episodes=None,  
                                                         save_frequency=None,  
                                                         save_frequency_unit=None,  
                                                         agents=None, envi-  
                                                         ronments=None)
```

Bases: tensorflow.execution.base_runner.BaseRunner

Runner for non-realtime threaded execution of multiple agents.

__init__ (*agent, environment, repeat_actions=1, save_path=None, save_episodes=None,*
 save_frequency=None, save_frequency_unit=None, agents=None, environments=None)

Initialize a ThreadedRunner object.

Parameters

- **save_path** (*str*) – Path where to save the shared model.
- **save_episodes** (*int*) – Deprecated: Every how many (global) episodes do we save the shared model?

- **save_frequency** (*int*) – The frequency with which to save the model (could be sec, steps, or episodes).
- **save_frequency_unit** (*str*) – “s” (sec), “t” (timesteps), “e” (episodes)
- **agents** (*List [Agent]*) – Deprecated: List of Agent objects. Use `agent`, instead.
- **environments** (*List [Environment]*) – Deprecated: List of Environment objects. Use `environment`, instead.

agents

close()

environments

episode

Deprecated property `episode` -> `global_episode`.

episode_lengths

global_step

reset (*history=None*)

Resets the Runner’s internal stats counters. If history is empty, use default values in `history.get()`.

Parameters **history** (*dict*) – A dictionary containing an already run experiment’s results. Keys should be: `episode_rewards` (list of rewards), `episode_timesteps` (lengths of episodes), `episode_times` (run-times)

run (*num_episodes=-1, max_episode_timesteps=-1, episode_finished=None, summary_report=None, summary_interval=0, num_timesteps=None, deterministic=False, episodes=None, max_timesteps=None*)

Executes this runner by starting all Agents in parallel (each one in one thread).

Parameters

- **episodes** (*int*) – Deprecated; see `num_episodes`.
- **max_timesteps** (*int*) – Deprecated; see `max_episode_timesteps`.

timestep

Deprecated property `timestep` -> `global_timestep`.

`tensorflow.execution.threaded_runner.WorkerAgentGenerator` (*agent_class*)

Worker Agent generator, receives an Agent class and creates a Worker Agent class that inherits from that Agent.

`tensorflow.execution.threaded_runner.clone_worker_agent` (*agent, factor, environment, network, agent_config*)

Clones a given Agent (`factor` times) and returns a list of the cloned Agents with the original Agent in the first slot.

Parameters

- **agent** (*Agent*) – The Agent object to clone.
- **factor** (*int*) – The length of the final list.
- **environment** (*Environment*) – The Environment to use for all cloned agents.
- **network** (*LayeredNetwork*) – The Network to use (or None) for an Agent’s Model.
- **agent_config** (*dict*) – A dict of Agent specifications passed into the Agent’s c’tor as kwargs.

Returns The list with `factor` cloned agents (including the original one).

Module contents

class `tensorforce.execution.BaseRunner` (*agent*, *environment*, *repeat_actions=1*, *history=None*)

Bases: `object`

Base class for all runner classes. Implements the `run` method.

__init__ (*agent*, *environment*, *repeat_actions=1*, *history=None*)

Parameters

- **agent** (*Agent*) – Agent object (or list of Agent objects) to use for the run.
- **environment** (*Environment*) – Environment object (or list of Environment objects) to use for the run.
- **repeat_actions** (*int*) – How many times the same given action will be repeated in subsequent calls to Environment’s `execute` method. Rewards collected in these calls are accumulated and reported as a sum in the following call to Agent’s `observe` method.
- **history** (*dict*) – A dictionary containing an already run experiment’s results. Keys should be: `episode_rewards` (list of rewards), `episode_timesteps` (lengths of episodes), `episode_times` (run-times)

close ()

Should perform clean up operations on Runner’s Agent(s) and Environment(s).

episode

Deprecated property `episode` -> `global_episode`.

reset (*history=None*)

Resets the Runner’s internal stats counters. If history is empty, use default values in `history.get()`.

Parameters **history** (*dict*) – A dictionary containing an already run experiment’s results. Keys should be: `episode_rewards` (list of rewards), `episode_timesteps` (lengths of episodes), `episode_times` (run-times)

run (*num_episodes*, *num_timesteps*, *max_episode_timesteps*, *deterministic*, *episode_finished*, *summary_report*, *summary_interval*)

Executes this runner by starting to act (via Agent(s)) in the given Environment(s). Stops execution according to certain conditions (e.g. max. number of episodes, etc..). Calls callback functions after each episode and/or after some summary criteria are met.

Parameters

- **num_episodes** (*int*) – Max. number of episodes to run globally in total (across all threads/workers).
- **num_timesteps** (*int*) – Max. number of time steps to run globally in total (across all threads/workers)
- **max_episode_timesteps** (*int*) – Max. number of timesteps per episode.
- **deterministic** (*bool*) – Whether to use exploration when selecting actions.
- **episode_finished** (*callable*) – A function to be called once an episodes has finished. Should take a BaseRunner object and some worker ID (e.g. thread-ID or task-ID). Can decide for itself every how many episodes it should report something and what to report.
- **summary_report** (*callable*) – Deprecated; Function that could produce a summary over the training progress so far.

- **summary_interval** (*int*) – Deprecated; The number of time steps to execute (globally) before `summary_report` is called.

timestep

Deprecated property `timestep` -> `global_timestep`.

`tensorflow.execution.SingleRunner`

alias of *Runner*

`tensorflow.execution.DistributedTFRunner`

alias of *Runner*

class `tensorflow.execution.Runner`(*agent*, *environment*, *repeat_actions=1*, *history=None*, *id_=0*)

Bases: `tensorflow.execution.base_runner.BaseRunner`

Simple runner for non-realtime single-process execution.

__init__ (*agent*, *environment*, *repeat_actions=1*, *history=None*, *id_=0*)

Initialize a single Runner object (one Agent/one Environment).

Parameters *id** – The ID of this Runner (for distributed TF runs).

:type *id**: int

close ()

episode

Deprecated property `episode` -> `global_episode`.

episode_timestep

reset (*history=None*)

Resets the Runner's internal stats counters. If history is empty, use default values in `history.get()`.

Parameters *history* (*dict*) – A dictionary containing an already run experiment's results.

Keys should be: `episode_rewards` (list of rewards), `episode_timesteps` (lengths of episodes), `episode_times` (run-times)

run (*num_timesteps=None*, *num_episodes=None*, *max_episode_timesteps=None*, *deterministic=False*, *episode_finished=None*, *summary_report=None*, *summary_interval=None*, *timesteps=None*, *episodes=None*)

Parameters

- **timesteps** (*int*) – Deprecated; see `num_timesteps`.
- **episodes** (*int*) – Deprecated; see `num_episodes`.

timestep

Deprecated property `timestep` -> `global_timestep`.

class `tensorflow.execution.ThreadedRunner`(*agent*, *environment*, *repeat_actions=1*, *save_path=None*, *save_episodes=None*, *save_frequency=None*, *save_frequency_unit=None*, *agents=None*, *environments=None*)

Bases: `tensorflow.execution.base_runner.BaseRunner`

Runner for non-realtime threaded execution of multiple agents.

__init__ (*agent*, *environment*, *repeat_actions=1*, *save_path=None*, *save_episodes=None*, *save_frequency=None*, *save_frequency_unit=None*, *agents=None*, *environments=None*)

Initialize a ThreadedRunner object.

Parameters

- **save_path** (*str*) – Path where to save the shared model.
- **save_episodes** (*int*) – Deprecated: Every how many (global) episodes do we save the shared model?
- **save_frequency** (*int*) – The frequency with which to save the model (could be sec, steps, or episodes).
- **save_frequency_unit** (*str*) – “s” (sec), “t” (timesteps), “e” (episodes)
- **agents** (*List [Agent]*) – Deprecated: List of Agent objects. Use `agent`, instead.
- **environments** (*List [Environment]*) – Deprecated: List of Environment objects. Use `environment`, instead.

agents

close ()

environments

episode

Deprecated property `episode` -> `global_episode`.

episode_lengths

global_step

reset (*history=None*)

Resets the Runner’s internal stats counters. If `history` is empty, use default values in `history.get()`.

Parameters **history** (*dict*) – A dictionary containing an already run experiment’s results. Keys should be: `episode_rewards` (list of rewards), `episode_timesteps` (lengths of episodes), `episode_times` (run-times)

run (*num_episodes=-1, max_episode_timesteps=-1, episode_finished=None, summary_report=None, summary_interval=0, num_timesteps=None, deterministic=False, episodes=None, max_timesteps=None*)

Executes this runner by starting all Agents in parallel (each one in one thread).

Parameters

- **episodes** (*int*) – Deprecated; see `num_episodes`.
- **max_timesteps** (*int*) – Deprecated; see `max_episode_timesteps`.

timestep

Deprecated property `timestep` -> `global_timestep`.

`tensorflow.execution.WorkerAgentGenerator` (*agent_class*)

Worker Agent generator, receives an Agent class and creates a Worker Agent class that inherits from that Agent.

tensorflow.models package

Submodules

tensorflow.models.constant_model module

class `tensorflow.models.constant_model.ConstantModel` (*states, actions, scope, device, saver, summarizer, execution, batching_capacity, action_values*)

Bases: `tensorflow.models.model.Model`

Utility class to return constant actions of a desired shape and with given bounds.

__init__ (*states, actions, scope, device, saver, summarizer, execution, batching_capacity, action_values*)

act (*states, internals, deterministic=False, independent=False, fetch_tensors=None*)

Does a forward pass through the model to retrieve action (outputs) given inputs for state (and internal state, if applicable (e.g. RNNs))

Parameters

- **states** (*dict*) – Dict of state values (each key represents one state space component).
- **internals** (*dict*) – Dict of internal state values (each key represents one internal state component).
- **deterministic** (*bool*) – If True, will not apply exploration after actions are calculated.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).

Returns

- Actual action-outputs (batched if state input is a batch).

Return type tuple

as_local_model ()

close ()

create_act_operations (*states, internals, deterministic, independent*)

create_observe_operations (*terminal, reward*)

create_operations (*states, internals, actions, terminal, reward, deterministic, independent*)

Creates output operations for acting, observing and interacting with the memory.

get_component (*component_name*)

Looks up a component by its name.

Parameters **component_name** – The name of the component to look up.

Returns The component for the provided name or None if there is no such component.

get_components ()

Returns a dictionary of component name to component of all the components within this model.

Returns (dict) The mapping of name to component.

get_feed_dict (*states=None, internals=None, actions=None, terminal=None, reward=None, deterministic=None, independent=None*)

get_savable_components ()

Returns the list of all of the components this model consists of that can be individually saved and restored. For instance the network or distribution.

Returns List of util.SavableComponent

get_summaries ()

Returns the TensorFlow summaries reported by the model

Returns List of summaries

get_variables (*include_submodules=False, include_nontrainable=False*)

Returns the TensorFlow variables used by the model.

Parameters

- **include_submodules** – Includes variables of submodules (e.g. baseline, target network) if true.
- **include_nontrainable** – Includes non-trainable variables if true.

Returns List of variables.

initialize (*custom_getter*)

Creates the TensorFlow placeholders and functions for this model. Moreover adds the internal state placeholders and initialization values to the model.

Parameters **custom_getter** – The `custom_getter_` object to use for `tf.make_template` when creating TensorFlow functions.

observe (*terminal, reward*)

Adds an observation (reward and is-terminal) to the model without updating its trainable variables.

Parameters

- **terminal** (*bool*) – Whether the episode has terminated.
- **reward** (*float*) – The observed reward value.

Returns The value of the model-internal episode counter.

reset ()

Resets the model to its initial state on episode start. This should also reset all preprocessor(s).

Returns Current episode, timestep counter and the shallow-copied list of internal state initialization Tensors.

Return type tuple

restore (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

restore_component (*component_name, save_path*)

Restores a component's parameters from a save location.

Parameters

- **component_name** – The component to restore.
- **save_path** – The save location.

save (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** – Optional checkpoint directory.
- **append_timestep** – Appends the current timestep to the checkpoint file if true.

Returns Checkpoint path where the model was saved.

save_component (*component_name*, *save_path*)

Saves a component of this model to the designated location.

Parameters

- **component_name** – The component to save.
- **save_path** – The location to save to.

Returns Checkpoint path where the component was saved.

setup ()

Sets up the TensorFlow model graph and initializes (and enters) the TensorFlow session.

tf_action_exploration (*action*, *exploration*, *action_spec*)

Applies optional exploration to the action (post-processor for action outputs).

Parameters

- **action** (*tf.Tensor*) – The original output action tensor (to be post-processed).
- **exploration** (*Exploration*) – The Exploration object to use.
- **action_spec** (*dict*) – Dict specifying the action space.

Returns The post-processed action output tensor.

tf_actions_and_internals (*states*, *internals*, *deterministic*)

tf_initialize ()

tf_observe_timestep (*states*, *internals*, *actions*, *terminal*, *reward*)

tf_preprocess (*states*, *actions*, *reward*)

tensorforce.models.distribution_model module

```
class tensorforce.models.distribution_model.DistributionModel (states, ac-
                                                                tions, scope,
                                                                device, saver,
                                                                summarizer,
                                                                execution, batch-
                                                                ing_capacity,
                                                                variable_noise,
                                                                states_preprocessing,
                                                                ac-
                                                                tions_exploration,
                                                                re-
                                                                ward_preprocessing,
                                                                update_mode,
                                                                memory, op-
                                                                timizer, dis-
                                                                count, network,
                                                                distributions, en-
                                                                tropy_regularization,
                                                                re-
                                                                quires_deterministic)
```

Bases: `tensorforce.models.memory_model.MemoryModel`

Base class for models using distributions parametrized by a neural network.

```
COMPONENT_DISTRIBUTION = 'distribution'
```

```
COMPONENT_NETWORK = 'network'
```

```
__init__(states, actions, scope, device, saver, summarizer, execution, batching_capacity,
         variable_noise, states_preprocessing, actions_exploration, reward_preprocessing, up-
         date_mode, memory, optimizer, discount, network, distributions, entropy_regularization, re-
         quires_deterministic)
```

```
act(states, internals, deterministic=False, independent=False, fetch_tensors=None)
```

Does a forward pass through the model to retrieve action (outputs) given inputs for state (and internal state, if applicable (e.g. RNNs))

Parameters

- **states** (*dict*) – Dict of state values (each key represents one state space component).
- **internals** (*dict*) – Dict of internal state values (each key represents one internal state component).
- **deterministic** (*bool*) – If True, will not apply exploration after actions are calculated.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).

Returns

- Actual action-outputs (batched if state input is a batch).

Return type

tuple

```
as_local_model()
```

```
close()
```

```
create_act_operations(states, internals, deterministic, independent)
```

```
create_distributions()
```

```
create_observe_operations(terminal, reward)
```

```
create_operations(states, internals, actions, terminal, reward, deterministic, independent)
```

```
get_component(component_name)
```

Looks up a component by its name.

Parameters **component_name** – The name of the component to look up.

Returns The component for the provided name or None if there is no such component.

```
get_components()
```

```
get_feed_dict(states=None, internals=None, actions=None, terminal=None, reward=None, deter-
              ministic=None, independent=None)
```

```
get_savable_components()
```

Returns the list of all of the components this model consists of that can be individually saved and restored.
For instance the network or distribution.

Returns List of util.SavableComponent

```
get_summaries()
```

```
get_variables(include_submodules=False, include_nontrainable=False)
```

import_experience (*states, internals, actions, terminal, reward*)

Stores experiences.

initialize (*custom_getter*)

observe (*terminal, reward*)

Adds an observation (reward and is-terminal) to the model without updating its trainable variables.

Parameters

- **terminal** (*bool*) – Whether the episode has terminated.
- **reward** (*float*) – The observed reward value.

Returns The value of the model-internal episode counter.

optimizer_arguments (*states, internals, actions, terminal, reward, next_states, next_internals*)

reset ()

Resets the model to its initial state on episode start. This should also reset all preprocessor(s).

Returns Current episode, timestep counter and the shallow-copied list of internal state initialization Tensors.

Return type tuple

restore (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

restore_component (*component_name, save_path*)

Restores a component's parameters from a save location.

Parameters

- **component_name** – The component to restore.
- **save_path** – The save location.

save (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** – Optional checkpoint directory.
- **append_timestep** – Appends the current timestep to the checkpoint file if true.

Returns Checkpoint path where the model was saved.

save_component (*component_name, save_path*)

Saves a component of this model to the designated location.

Parameters

- **component_name** – The component to save.
- **save_path** – The location to save to.

Returns Checkpoint path where the component was saved.

setup()

Sets up the TensorFlow model graph and initializes (and enters) the TensorFlow session.

tf_action_exploration(*action, exploration, action_spec*)

Applies optional exploration to the action (post-processor for action outputs).

Parameters

- **action** (*tf.Tensor*) – The original output action tensor (to be post-processed).
- **exploration** (*Exploration*) – The Exploration object to use.
- **action_spec** (*dict*) – Dict specifying the action space.

Returns The post-processed action output tensor.

tf_actions_and_internals(*states, internals, deterministic*)

tf_discounted_cumulative_reward(*terminal, reward, discount, final_reward=0.0*)

Creates the TensorFlow operations for calculating the discounted cumulative rewards for a given sequence of rewards.

Parameters

- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **discount** – Discount factor.
- **final_reward** – Last reward value in the sequence.

Returns Discounted cumulative reward tensor.

tf_import_experience(*states, internals, actions, terminal, reward*)

Imports experiences into the TensorFlow memory structure. Can be used to import off-policy data.

Parameters

- **states** – Dict of state values to import with keys as state names and values as values to set.
- **internals** – Internal values to set, can be fetched from agent via agent.current_internals if no values available.
- **actions** – Dict of action values to import with keys as action names and values as values to set.
- **terminal** – Terminal value(s)
- **reward** – Reward value(s)

tf_initialize()

tf_kl_divergence(*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

tf_loss(*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

Creates the TensorFlow operations for calculating the full loss of a batch.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.

- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor.

tf_loss_per_instance (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

Creates the TensorFlow operations for calculating the loss per batch instance.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss per instance tensor.

tf_observe_timestep (*states, internals, actions, terminal, reward*)

tf_optimization (*states, internals, actions, terminal, reward, next_states=None, next_internals=None*)

Creates the TensorFlow operations for performing an optimization update step based on the given input states and actions batch.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.

Returns The optimization operation.

tf_preprocess (*states, actions, reward*)

tf_reference (*states, internals, actions, terminal, reward, next_states, next_internals, update*)

Creates the TensorFlow operations for obtaining the reference tensor(s), in case of a comparative loss.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Reference tensor(s).

tf_regularization_losses (*states, internals, update*)

tensorforce.models.model module

The `Model` class coordinates the creation and execution of all TensorFlow operations within a model. It implements the `reset`, `act` and `update` functions, which form the interface the `Agent` class communicates with, and which should not need to be overwritten. Instead, the following TensorFlow functions need to be implemented:

- **tf_actions_and_internals**(*states, internals, deterministic*) returning the batch of actions and successor internal states.
- **tf_loss_per_instance**(*states, internals, actions, terminal, reward*) returning the loss per instance for a batch.

Further, the following TensorFlow functions should be extended accordingly:

- **initialize**(*custom_getter*) defining TensorFlow placeholders/functions and adding internal states.
- **get_variables**() returning the list of TensorFlow variables (to be optimized) of this model.
- **tf_regularization_losses**(*states, internals*) returning a dict of regularization losses.
- **get_optimizer_kwargs**(*states, internals, actions, terminal, reward*) returning a dict of potential arguments (argument-free functions) to the optimizer.

Finally, the following TensorFlow functions can be useful in some cases:

- **preprocess_states**(*states*) for state preprocessing, returning the processed batch of states.
- **tf_action_exploration**(*action, exploration, action_spec*) for action postprocessing (e.g. exploration), returning the processed batch of actions.
- **tf_preprocess_reward**(*states, internals, terminal, reward*) for reward preprocessing (e.g. reward normalization), returning the processed batch of rewards.
- **create_output_operations**(*states, internals, actions, terminal, reward, deterministic*) for creating output operations similar to the two above for `Model.act` and `Model.update`.
- **tf_optimization**(*states, internals, actions, terminal, reward*) for further optimization operations (e.g. the baseline update in a `PGModel` or the target network update in a `QModel`), returning a single grouped optimization operation.


```
class tensorflow.models.model.Model(states, actions, scope, device, saver, summarizer,  
                                     execution, batching_capacity, variable_noise,  
                                     states_preprocessing, actions_exploration, re-  
                                     ward_preprocessing)
```

Bases: object

Base class for all (TensorFlow-based) models.

```
__init__(states, actions, scope, device, saver, summarizer, execution, batching_capacity, vari-  
         able_noise, states_preprocessing, actions_exploration, reward_preprocessing)  
Model.
```

Parameters

- **states** (*spec*) – The state-space description dictionary.
- **actions** (*spec*) – The action-space description dictionary.
- **scope** (*str*) – The root scope str to use for tf variable scoping.
- **device** (*str*) – The name of the device to run the graph of this model on.
- **saver** (*spec*) – Dict specifying whether and how to save the model’s parameters.
- **summarizer** (*spec*) – Dict specifying which tensorboard summaries should be created and added to the graph.
- **execution** (*spec*) – Dict specifying whether and how to do distributed training on the model’s graph.
- **batching_capacity** (*int*) – Batching capacity.
- **variable_noise** (*float*) – The stddev value of a Normal distribution used for adding random noise to the model’s output (for each batch, noise can be toggled and - if active - will be resampled). Use None for not adding any noise.
- **states_preprocessing** (*spec / dict of specs*) – Dict specifying whether and how to preprocess state signals (e.g. normalization, greyscale, etc..).
- **actions_exploration** (*spec / dict of specs*) – Dict specifying whether and how to add exploration to the model’s “action outputs” (e.g. epsilon-greedy).
- **reward_preprocessing** (*spec*) – Dict specifying whether and how to preprocess rewards coming from the Environment (e.g. reward normalization).

```
act(states, internals, deterministic=False, independent=False, fetch_tensors=None)
```

Does a forward pass through the model to retrieve action (outputs) given inputs for state (and internal state, if applicable (e.g. RNNs))

Parameters

- **states** (*dict*) – Dict of state values (each key represents one state space component).
- **internals** (*dict*) – Dict of internal state values (each key represents one internal state component).
- **deterministic** (*bool*) – If True, will not apply exploration after actions are calculated.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).

Returns

- Actual action-outputs (batched if state input is a batch).

Return type tuple

as_local_model ()

close ()

create_act_operations (*states, internals, deterministic, independent*)

create_observe_operations (*terminal, reward*)

create_operations (*states, internals, actions, terminal, reward, deterministic, independent*)

Creates output operations for acting, observing and interacting with the memory.

get_component (*component_name*)

Looks up a component by its name.

Parameters **component_name** – The name of the component to look up.

Returns The component for the provided name or None if there is no such component.

get_components ()

Returns a dictionary of component name to component of all the components within this model.

Returns (dict) The mapping of name to component.

get_feed_dict (*states=None, internals=None, actions=None, terminal=None, reward=None, deterministic=None, independent=None*)

get_savable_components ()

Returns the list of all of the components this model consists of that can be individually saved and restored. For instance the network or distribution.

Returns List of util.SavableComponent

get_summaries ()

Returns the TensorFlow summaries reported by the model

Returns List of summaries

get_variables (*include_submodules=False, include_nontrainable=False*)

Returns the TensorFlow variables used by the model.

Parameters

- **include_submodules** – Includes variables of submodules (e.g. baseline, target network) if true.
- **include_nontrainable** – Includes non-trainable variables if true.

Returns List of variables.

initialize (*custom_getter*)

Creates the TensorFlow placeholders and functions for this model. Moreover adds the internal state placeholders and initialization values to the model.

Parameters **custom_getter** – The `custom_getter_` object to use for `tf.make_template` when creating TensorFlow functions.

observe (*terminal, reward*)

Adds an observation (reward and is-terminal) to the model without updating its trainable variables.

Parameters

- **terminal** (*bool*) – Whether the episode has terminated.
- **reward** (*float*) – The observed reward value.

Returns The value of the model-internal episode counter.

reset ()

Resets the model to its initial state on episode start. This should also reset all preprocessor(s).

Returns Current episode, timestep counter and the shallow-copied list of internal state initialization Tensors.

Return type tuple

restore (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

restore_component (*component_name, save_path*)

Restores a component's parameters from a save location.

Parameters

- **component_name** – The component to restore.
- **save_path** – The save location.

save (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** – Optional checkpoint directory.
- **append_timestep** – Appends the current timestep to the checkpoint file if true.

Returns Checkpoint path where the model was saved.

save_component (*component_name, save_path*)

Saves a component of this model to the designated location.

Parameters

- **component_name** – The component to save.
- **save_path** – The location to save to.

Returns Checkpoint path where the component was saved.

setup ()

Sets up the TensorFlow model graph and initializes (and enters) the TensorFlow session.

tf_action_exploration (*action, exploration, action_spec*)

Applies optional exploration to the action (post-processor for action outputs).

Parameters

- **action** (*tf.Tensor*) – The original output action tensor (to be post-processed).
- **exploration** (*Exploration*) – The Exploration object to use.
- **action_spec** (*dict*) – Dict specifying the action space.

Returns The post-processed action output tensor.

tf_actions_and_internals (*states, internals, deterministic*)

Creates and returns the TensorFlow operations for retrieving the actions and - if applicable - the posterior internal state Tensors in reaction to the given input states (and prior internal states).

Parameters

- **states** (*dict*) – Dict of state tensors (each key represents one state space component).
- **internals** – List of prior internal state tensors.
- **deterministic** – Boolean tensor indicating whether action should be chosen deterministically.

Returns

1. dict of output actions (with or without exploration applied (see *deterministic*))
2. list of posterior internal state Tensors (empty for non-internal state models)

Return type tuple

tf_initialize ()

tf_observe_timestep (*states, internals, actions, terminal, reward*)

Creates the TensorFlow operations for performing the observation of a full time step's information.

Parameters

- **states** (*dict*) – Dict of state tensors (each key represents one state space component).
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.

Returns The observation operation.

tf_preprocess (*states, actions, reward*)

tensorforce.models.pg_log_prob_model module

```
class tensorforce.models.pg_log_prob_model.PGLogProbModel (states, actions, scope,  
                                                         device, saver, sum-  
                                                         marizer, execution,  
                                                         batching_capacity,  
                                                         variable_noise,  
                                                         states_preprocessing,  
                                                         actions_exploration,  
                                                         reward_preprocessing,  
                                                         update_mode, memory,  
                                                         optimizer, discount,  
                                                         network, distributions,  
                                                         entropy_regularization,  
                                                         baseline_mode,  
                                                         baseline,         base-  
                                                         line_optimizer,  
                                                         gae_lambda)
```

Bases: *tensorforce.models.pg_model.PGModel*

Policy gradient model based on computing log likelihoods, e.g. VPG.

```
COMPONENT_BASELINE = 'baseline'
```

```
COMPONENT_DISTRIBUTION = 'distribution'
```

```
COMPONENT_NETWORK = 'network'
```

```
__init__(states, actions, scope, device, saver, summarizer, execution, batching_capacity,
         variable_noise, states_preprocessing, actions_exploration, reward_preprocessing, up-
         date_mode, memory, optimizer, discount, network, distributions, entropy_regularization,
         baseline_mode, baseline, baseline_optimizer, gae_lambda)
```

```
act(states, internals, deterministic=False, independent=False, fetch_tensors=None)
```

Does a forward pass through the model to retrieve action (outputs) given inputs for state (and internal state, if applicable (e.g. RNNs))

Parameters

- **states** (*dict*) – Dict of state values (each key represents one state space component).
- **internals** (*dict*) – Dict of internal state values (each key represents one internal state component).
- **deterministic** (*bool*) – If True, will not apply exploration after actions are calculated.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).

Returns

- Actual action-outputs (batched if state input is a batch).

Return type tuple

```
as_local_model()
```

```
baseline_optimizer_arguments(states, internals, reward)
```

Returns the baseline optimizer arguments including the time, the list of variables to optimize, and various functions which the optimizer might require to perform an update step.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.

Returns Baseline optimizer arguments as dict.

```
close()
```

```
create_act_operations(states, internals, deterministic, independent)
```

```
create_distributions()
```

```
create_observe_operations(terminal, reward)
```

```
create_operations(states, internals, actions, terminal, reward, deterministic, independent)
```

```
get_component(component_name)
```

Looks up a component by its name.

Parameters **component_name** – The name of the component to look up.

Returns The component for the provided name or None if there is no such component.

get_components ()

get_feed_dict (*states=None, internals=None, actions=None, terminal=None, reward=None, deterministic=None, independent=None*)

get_savable_components ()

Returns the list of all of the components this model consists of that can be individually saved and restored.
For instance the network or distribution.

Returns List of util.SavableComponent

get_summaries ()

get_variables (*include_submodules=False, include_nontrainable=False*)

import_experience (*states, internals, actions, terminal, reward*)

Stores experiences.

initialize (*custom_getter*)

observe (*terminal, reward*)

Adds an observation (reward and is-terminal) to the model without updating its trainable variables.

Parameters

- **terminal** (*bool*) – Whether the episode has terminated.
- **reward** (*float*) – The observed reward value.

Returns The value of the model-internal episode counter.

optimizer_arguments (*states, internals, actions, terminal, reward, next_states, next_internals*)

reset ()

Resets the model to its initial state on episode start. This should also reset all preprocessor(s).

Returns Current episode, timestep counter and the shallow-copied list of internal state initialization Tensors.

Return type tuple

restore (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

restore_component (*component_name, save_path*)

Restores a component's parameters from a save location.

Parameters

- **component_name** – The component to restore.
- **save_path** – The save location.

save (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** – Optional checkpoint directory.

- **append_timestep** – Appends the current timestep to the checkpoint file if true.

Returns Checkpoint path where the model was saved.

save_component (*component_name*, *save_path*)

Saves a component of this model to the designated location.

Parameters

- **component_name** – The component to save.
- **save_path** – The location to save to.

Returns Checkpoint path where the component was saved.

setup ()

Sets up the TensorFlow model graph and initializes (and enters) the TensorFlow session.

tf_action_exploration (*action*, *exploration*, *action_spec*)

Applies optional exploration to the action (post-processor for action outputs).

Parameters

- **action** (*tf.Tensor*) – The original output action tensor (to be post-processed).
- **exploration** (*Exploration*) – The Exploration object to use.
- **action_spec** (*dict*) – Dict specifying the action space.

Returns The post-processed action output tensor.

tf_actions_and_internals (*states*, *internals*, *deterministic*)

tf_baseline_loss (*states*, *internals*, *reward*, *update*, *reference=None*)

Creates the TensorFlow operations for calculating the baseline loss of a batch.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor.

tf_discounted_cumulative_reward (*terminal*, *reward*, *discount*, *final_reward=0.0*)

Creates the TensorFlow operations for calculating the discounted cumulative rewards for a given sequence of rewards.

Parameters

- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **discount** – Discount factor.
- **final_reward** – Last reward value in the sequence.

Returns Discounted cumulative reward tensor.

tf_import_experience (*states*, *internals*, *actions*, *terminal*, *reward*)

Imports experiences into the TensorFlow memory structure. Can be used to import off-policy data.

Parameters

- **states** – Dict of state values to import with keys as state names and values as values to set.
- **internals** – Internal values to set, can be fetched from agent via `agent.current_internals` if no values available.
- **actions** – Dict of action values to import with keys as action names and values as values to set.
- **terminal** – Terminal value(s)
- **reward** – Reward value(s)

tf_initialize()

tf_kl_divergence (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

tf_loss (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)
Creates the TensorFlow operations for calculating the full loss of a batch.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor.

tf_loss_per_instance (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

tf_observe_timestep (*states, internals, actions, terminal, reward*)

tf_optimization (*states, internals, actions, terminal, reward, next_states=None, next_internals=None*)

tf_preprocess (*states, actions, reward*)

tf_reference (*states, internals, actions, terminal, reward, next_states, next_internals, update*)
Creates the TensorFlow operations for obtaining the reference tensor(s), in case of a comparative loss.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.

- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Reference tensor(s).

tf_regularization_losses (*states, internals, update*)

tf_reward_estimation (*states, internals, terminal, reward, update*)

tensorforce.models.pg_model module

```
class tensorforce.models.pg_model.PGModel(states, actions, scope, device, saver, summarizer, execution, batching_capacity, variable_noise, states_preprocessing, actions_exploration, reward_preprocessing, update_mode, memory, optimizer, discount, network, distributions, entropy_regularization, baseline_mode, baseline, baseline_optimizer, gae_lambda)
```

Bases: *tensorforce.models.distribution_model.DistributionModel*

Base class for policy gradient models. It optionally defines a baseline and handles its optimization. It implements the `tf_loss_per_instance` function, but requires subclasses to implement `tf_pg_loss_per_instance`.

COMPONENT_BASELINE = 'baseline'

COMPONENT_DISTRIBUTION = 'distribution'

COMPONENT_NETWORK = 'network'

```
__init__(states, actions, scope, device, saver, summarizer, execution, batching_capacity, variable_noise, states_preprocessing, actions_exploration, reward_preprocessing, update_mode, memory, optimizer, discount, network, distributions, entropy_regularization, baseline_mode, baseline, baseline_optimizer, gae_lambda)
```

act (*states, internals, deterministic=False, independent=False, fetch_tensors=None*)

Does a forward pass through the model to retrieve action (outputs) given inputs for state (and internal state, if applicable (e.g. RNNs))

Parameters

- **states** (*dict*) – Dict of state values (each key represents one state space component).
- **internals** (*dict*) – Dict of internal state values (each key represents one internal state component).
- **deterministic** (*bool*) – If True, will not apply exploration after actions are calculated.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).

Returns

- Actual action-outputs (batched if state input is a batch).

Return type tuple

as_local_model ()

baseline_optimizer_arguments (*states, internals, reward*)

Returns the baseline optimizer arguments including the time, the list of variables to optimize, and various functions which the optimizer might require to perform an update step.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.

Returns Baseline optimizer arguments as dict.

close ()

create_act_operations (*states, internals, deterministic, independent*)

create_distributions ()

create_observe_operations (*terminal, reward*)

create_operations (*states, internals, actions, terminal, reward, deterministic, independent*)

get_component (*component_name*)

Looks up a component by its name.

Parameters **component_name** – The name of the component to look up.

Returns The component for the provided name or None if there is no such component.

get_components ()

get_feed_dict (*states=None, internals=None, actions=None, terminal=None, reward=None, deterministic=None, independent=None*)

get_savable_components ()

Returns the list of all of the components this model consists of that can be individually saved and restored. For instance the network or distribution.

Returns List of util.SavableComponent

get_summaries ()

get_variables (*include_submodules=False, include_nontrainable=False*)

import_experience (*states, internals, actions, terminal, reward*)

Stores experiences.

initialize (*custom_getter*)

observe (*terminal, reward*)

Adds an observation (reward and is-terminal) to the model without updating its trainable variables.

Parameters

- **terminal** (*bool*) – Whether the episode has terminated.
- **reward** (*float*) – The observed reward value.

Returns The value of the model-internal episode counter.

optimizer_arguments (*states, internals, actions, terminal, reward, next_states, next_internals*)

reset ()

Resets the model to its initial state on episode start. This should also reset all preprocessor(s).

Returns Current episode, timestep counter and the shallow-copied list of internal state initialization Tensors.

Return type tuple

restore (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

restore_component (*component_name, save_path*)

Restores a component's parameters from a save location.

Parameters

- **component_name** – The component to restore.
- **save_path** – The save location.

save (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** – Optional checkpoint directory.
- **append_timestep** – Appends the current timestep to the checkpoint file if true.

Returns Checkpoint path where the model was saved.

save_component (*component_name, save_path*)

Saves a component of this model to the designated location.

Parameters

- **component_name** – The component to save.
- **save_path** – The location to save to.

Returns Checkpoint path where the component was saved.

setup ()

Sets up the TensorFlow model graph and initializes (and enters) the TensorFlow session.

tf_action_exploration (*action, exploration, action_spec*)

Applies optional exploration to the action (post-processor for action outputs).

Parameters

- **action** (*tf.Tensor*) – The original output action tensor (to be post-processed).
- **exploration** (*Exploration*) – The Exploration object to use.
- **action_spec** (*dict*) – Dict specifying the action space.

Returns The post-processed action output tensor.

tf_actions_and_internals (*states, internals, deterministic*)

tf_baseline_loss (*states, internals, reward, update, reference=None*)

Creates the TensorFlow operations for calculating the baseline loss of a batch.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor.

tf_discounted_cumulative_reward (*terminal, reward, discount, final_reward=0.0*)

Creates the TensorFlow operations for calculating the discounted cumulative rewards for a given sequence of rewards.

Parameters

- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **discount** – Discount factor.
- **final_reward** – Last reward value in the sequence.

Returns Discounted cumulative reward tensor.

tf_import_experience (*states, internals, actions, terminal, reward*)

Imports experiences into the TensorFlow memory structure. Can be used to import off-policy data.

Parameters

- **states** – Dict of state values to import with keys as state names and values as values to set.
- **internals** – Internal values to set, can be fetched from agent via agent.current_internals if no values available.
- **actions** – Dict of action values to import with keys as action names and values as values to set.
- **terminal** – Terminal value(s)
- **reward** – Reward value(s)

tf_initialize ()

tf_kl_divergence (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

tf_loss (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

Creates the TensorFlow operations for calculating the full loss of a batch.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.

- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor.

tf_loss_per_instance (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

Creates the TensorFlow operations for calculating the loss per batch instance.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss per instance tensor.

tf_observe_timestep (*states, internals, actions, terminal, reward*)

tf_optimization (*states, internals, actions, terminal, reward, next_states=None, next_internals=None*)

tf_preprocess (*states, actions, reward*)

tf_reference (*states, internals, actions, terminal, reward, next_states, next_internals, update*)

Creates the TensorFlow operations for obtaining the reference tensor(s), in case of a comparative loss.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Reference tensor(s).

tf_regularization_losses (*states, internals, update*)

tf_reward_estimation (*states, internals, terminal, reward, update*)

tensorforce.models.pg_prob_ratio_model module

```
class tensorforce.models.pg_prob_ratio_model.PGProbRatioModel(states, actions, scope, device, saver, summarizer, execution, batching_capacity, variable_noise, states_preprocessing, actions_exploration, reward_preprocessing, update_mode, memory, optimizer, discount, network, distributions, entropy_regularization, baseline_mode, baseline, baseline_optimizer, gae_lambda, likelihood_ratio_clipping)
```

Bases: `tensorforce.models.pg_model.PGModel`

Policy gradient model based on computing likelihood ratios, e.g. TRPO and PPO.

COMPONENT_BASELINE = 'baseline'

COMPONENT_DISTRIBUTION = 'distribution'

COMPONENT_NETWORK = 'network'

```
__init__(states, actions, scope, device, saver, summarizer, execution, batching_capacity, variable_noise, states_preprocessing, actions_exploration, reward_preprocessing, update_mode, memory, optimizer, discount, network, distributions, entropy_regularization, baseline_mode, baseline, baseline_optimizer, gae_lambda, likelihood_ratio_clipping)
```

```
act(states, internals, deterministic=False, independent=False, fetch_tensors=None)
```

Does a forward pass through the model to retrieve action (outputs) given inputs for state (and internal state, if applicable (e.g. RNNs))

Parameters

- **states** (*dict*) – Dict of state values (each key represents one state space component).
- **internals** (*dict*) – Dict of internal state values (each key represents one internal state component).
- **deterministic** (*bool*) – If True, will not apply exploration after actions are calculated.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).

Returns

- Actual action-outputs (batched if state input is a batch).

Return type tuple

as_local_model ()

baseline_optimizer_arguments (*states, internals, reward*)

Returns the baseline optimizer arguments including the time, the list of variables to optimize, and various functions which the optimizer might require to perform an update step.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.

Returns Baseline optimizer arguments as dict.

close ()

create_act_operations (*states, internals, deterministic, independent*)

create_distributions ()

create_observe_operations (*terminal, reward*)

create_operations (*states, internals, actions, terminal, reward, deterministic, independent*)

get_component (*component_name*)

Looks up a component by its name.

Parameters **component_name** – The name of the component to look up.

Returns The component for the provided name or None if there is no such component.

get_components ()

get_feed_dict (*states=None, internals=None, actions=None, terminal=None, reward=None, deterministic=None, independent=None*)

get_savable_components ()

Returns the list of all of the components this model consists of that can be individually saved and restored. For instance the network or distribution.

Returns List of util.SavableComponent

get_summaries ()

get_variables (*include_submodules=False, include_nontrainable=False*)

import_experience (*states, internals, actions, terminal, reward*)

Stores experiences.

initialize (*custom_getter*)

observe (*terminal, reward*)

Adds an observation (reward and is-terminal) to the model without updating its trainable variables.

Parameters

- **terminal** (*bool*) – Whether the episode has terminated.
- **reward** (*float*) – The observed reward value.

Returns The value of the model-internal episode counter.

optimizer_arguments (*states, internals, actions, terminal, reward, next_states, next_internals*)

reset ()

Resets the model to its initial state on episode start. This should also reset all preprocessor(s).

Returns Current episode, timestep counter and the shallow-copied list of internal state initialization Tensors.

Return type tuple

restore (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

restore_component (*component_name, save_path*)

Restores a component's parameters from a save location.

Parameters

- **component_name** – The component to restore.
- **save_path** – The save location.

save (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** – Optional checkpoint directory.
- **append_timestep** – Appends the current timestep to the checkpoint file if true.

Returns Checkpoint path where the model was saved.

save_component (*component_name, save_path*)

Saves a component of this model to the designated location.

Parameters

- **component_name** – The component to save.
- **save_path** – The location to save to.

Returns Checkpoint path where the component was saved.

setup ()

Sets up the TensorFlow model graph and initializes (and enters) the TensorFlow session.

tf_action_exploration (*action, exploration, action_spec*)

Applies optional exploration to the action (post-processor for action outputs).

Parameters

- **action** (*tf.Tensor*) – The original output action tensor (to be post-processed).
- **exploration** (*Exploration*) – The Exploration object to use.
- **action_spec** (*dict*) – Dict specifying the action space.

Returns The post-processed action output tensor.

tf_actions_and_internals (*states, internals, deterministic*)

tf_baseline_loss (*states, internals, reward, update, reference=None*)

Creates the TensorFlow operations for calculating the baseline loss of a batch.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor.

tf_discounted_cumulative_reward (*terminal, reward, discount, final_reward=0.0*)

Creates the TensorFlow operations for calculating the discounted cumulative rewards for a given sequence of rewards.

Parameters

- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **discount** – Discount factor.
- **final_reward** – Last reward value in the sequence.

Returns Discounted cumulative reward tensor.

tf_import_experience (*states, internals, actions, terminal, reward*)

Imports experiences into the TensorFlow memory structure. Can be used to import off-policy data.

Parameters

- **states** – Dict of state values to import with keys as state names and values as values to set.
- **internals** – Internal values to set, can be fetched from agent via agent.current_internals if no values available.
- **actions** – Dict of action values to import with keys as action names and values as values to set.
- **terminal** – Terminal value(s)
- **reward** – Reward value(s)

tf_initialize ()

tf_kl_divergence (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

tf_loss (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

Creates the TensorFlow operations for calculating the full loss of a batch.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.

- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor.

```
tf_loss_per_instance (states, internals, actions, terminal, reward, next_states, next_internals, up-  
date, reference=None)  
tf_observe_timestep (states, internals, actions, terminal, reward)  
tf_optimization (states, internals, actions, terminal, reward, next_states=None,  
next_internals=None)  
tf_preprocess (states, actions, reward)  
tf_reference (states, internals, actions, terminal, reward, next_states, next_internals, update)  
tf_regularization_losses (states, internals, update)  
tf_reward_estimation (states, internals, terminal, reward, update)
```

tensorforce.models.q_demo_model module

```
class tensorforce.models.q_demo_model.QDemoModel (states, actions, scope, device,  
saver, summarizer, execu-  
tion, batching_capacity, vari-  
able_noise, states_preprocessing,  
actions_exploration, re-  
ward_preprocessing, up-  
date_mode, memory, optimizer,  
discount, network, distribu-  
tions, entropy_regularization,  
target_sync_frequency, tar-  
get_update_weight, dou-  
ble_q_model, huber_loss, ex-  
pert_margin, supervised_weight,  
demo_memory_capacity,  
demo_batch_size)
```

Bases: `tensorforce.models.q_model.QModel`

Model for deep Q-learning from demonstration. Principal structure similar to double deep Q-networks but uses additional loss terms for demo data.

```
COMPONENT_DISTRIBUTION = 'distribution'  
COMPONENT_NETWORK = 'network'  
COMPONENT_TARGET_DISTRIBUTION = 'target_distribution'  
COMPONENT_TARGET_NETWORK = 'target_network'
```

__init__ (*states, actions, scope, device, saver, summarizer, execution, batching_capacity, variable_noise, states_preprocessing, actions_exploration, reward_preprocessing, update_mode, memory, optimizer, discount, network, distributions, entropy_regularization, target_sync_frequency, target_update_weight, double_q_model, huber_loss, expert_margin, supervised_weight, demo_memory_capacity, demo_batch_size*)

act (*states, internals, deterministic=False, independent=False, fetch_tensors=None*)

Does a forward pass through the model to retrieve action (outputs) given inputs for state (and internal state, if applicable (e.g. RNNs))

Parameters

- **states** (*dict*) – Dict of state values (each key represents one state space component).
- **internals** (*dict*) – Dict of internal state values (each key represents one internal state component).
- **deterministic** (*bool*) – If True, will not apply exploration after actions are calculated.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).

Returns

- Actual action-outputs (batched if state input is a batch).

Return type

as_local_model ()

close ()

create_act_operations (*states, internals, deterministic, independent*)

create_distributions ()

create_observe_operations (*terminal, reward*)

create_operations (*states, internals, actions, terminal, reward, deterministic, independent*)

demo_update ()

Performs a demonstration update by calling the demo optimization operation. Note that the batch data does not have to be fetched from the demo memory as this is now part of the TensorFlow operation of the demo update.

get_component (*component_name*)

Looks up a component by its name.

Parameters **component_name** – The name of the component to look up.

Returns The component for the provided name or None if there is no such component.

get_components ()

get_feed_dict (*states=None, internals=None, actions=None, terminal=None, reward=None, deterministic=None, independent=None*)

get_savable_components ()

Returns the list of all of the components this model consists of that can be individually saved and restored. For instance the network or distribution.

Returns List of util.SavableComponent

get_summaries ()

get_variables (*include_submodules=False, include_nontrainable=False*)

Returns the TensorFlow variables used by the model.

Returns List of variables.

import_demo_experience (*states, internals, actions, terminal, reward*)

Stores demonstrations in the demo memory.

import_experience (*states, internals, actions, terminal, reward*)

Stores experiences.

initialize (*custom_getter*)

observe (*terminal, reward*)

Adds an observation (reward and is-terminal) to the model without updating its trainable variables.

Parameters

- **terminal** (*bool*) – Whether the episode has terminated.
- **reward** (*float*) – The observed reward value.

Returns The value of the model-internal episode counter.

optimizer_arguments (*states, internals, actions, terminal, reward, next_states, next_internals*)

reset ()

Resets the model to its initial state on episode start. This should also reset all preprocessor(s).

Returns Current episode, timestep counter and the shallow-copied list of internal state initialization Tensors.

Return type tuple

restore (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

restore_component (*component_name, save_path*)

Restores a component's parameters from a save location.

Parameters

- **component_name** – The component to restore.
- **save_path** – The save location.

save (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** – Optional checkpoint directory.
- **append_timestep** – Appends the current timestep to the checkpoint file if true.

Returns Checkpoint path where the model was saved.

save_component (*component_name*, *save_path*)

Saves a component of this model to the designated location.

Parameters

- **component_name** – The component to save.
- **save_path** – The location to save to.

Returns Checkpoint path where the component was saved.

setup ()

Sets up the TensorFlow model graph and initializes (and enters) the TensorFlow session.

target_optimizer_arguments ()

Returns the target optimizer arguments including the time, the list of variables to optimize, and various functions which the optimizer might require to perform an update step.

Returns Target optimizer arguments as dict.

tf_action_exploration (*action*, *exploration*, *action_spec*)

Applies optional exploration to the action (post-processor for action outputs).

Parameters

- **action** (*tf.Tensor*) – The original output action tensor (to be post-processed).
- **exploration** (*Exploration*) – The Exploration object to use.
- **action_spec** (*dict*) – Dict specifying the action space.

Returns The post-processed action output tensor.

tf_actions_and_internals (*states*, *internals*, *deterministic*)

tf_combined_loss (*states*, *internals*, *actions*, *terminal*, *reward*, *next_states*, *next_internals*, *update*, *reference=None*)

Combines Q-loss and demo loss.

tf_demo_loss (*states*, *actions*, *terminal*, *reward*, *internals*, *update*, *reference=None*)

Extends the q-model loss via the dqfd large-margin loss.

tf_demo_optimization (*states*, *internals*, *actions*, *terminal*, *reward*, *next_states*, *next_internals*)

tf_discounted_cumulative_reward (*terminal*, *reward*, *discount*, *final_reward=0.0*)

Creates the TensorFlow operations for calculating the discounted cumulative rewards for a given sequence of rewards.

Parameters

- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **discount** – Discount factor.
- **final_reward** – Last reward value in the sequence.

Returns Discounted cumulative reward tensor.

tf_import_demo_experience (*states*, *internals*, *actions*, *terminal*, *reward*)

Imports a single experience to memory.

tf_import_experience (*states*, *internals*, *actions*, *terminal*, *reward*)

Imports experiences into the TensorFlow memory structure. Can be used to import off-policy data.

Parameters

- **states** – Dict of state values to import with keys as state names and values as values to set.
- **internals** – Internal values to set, can be fetched from agent via `agent.current_internals` if no values available.
- **actions** – Dict of action values to import with keys as action names and values as values to set.
- **terminal** – Terminal value(s)
- **reward** – Reward value(s)

tf_initialize()

tf_kl_divergence (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

tf_loss (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

Creates the TensorFlow operations for calculating the full loss of a batch.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor.

tf_loss_per_instance (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

tf_observe_timestep (*states, internals, actions, terminal, reward*)

tf_optimization (*states, internals, actions, terminal, reward, next_states=None, next_internals=None*)

tf_preprocess (*states, actions, reward*)

tf_q_delta (*q_value, next_q_value, terminal, reward*)

Creates the deltas (or advantage) of the Q values.

Returns A list of deltas per action

tf_q_value (*embedding, distr_params, action, name*)

tf_reference (*states, internals, actions, terminal, reward, next_states, next_internals, update*)

Creates the TensorFlow operations for obtaining the reference tensor(s), in case of a comparative loss.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.

- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Reference tensor(s).

tf_regularization_losses (*states, internals, update*)

tensorforce.models.q_model module

```
class tensorforce.models.q_model.QModel(states, actions, scope, device, saver, summarizer,
                                         execution, batching_capacity, variable_noise,
                                         states_preprocessing, actions_exploration, re-
                                         ward_preprocessing, update_mode, memory,
                                         optimizer, discount, network, distributions, en-
                                         tropy_regularization, target_sync_frequency,
                                         target_update_weight, double_q_model, hu-
                                         ber_loss)
```

Bases: *tensorforce.models.distribution_model.DistributionModel*

Q-value model.

COMPONENT_DISTRIBUTION = 'distribution'

COMPONENT_NETWORK = 'network'

COMPONENT_TARGET_DISTRIBUTION = 'target_distribution'

COMPONENT_TARGET_NETWORK = 'target_network'

```
__init__(states, actions, scope, device, saver, summarizer, execution, batching_capacity,
          variable_noise, states_preprocessing, actions_exploration, reward_preprocessing, up-
          date_mode, memory, optimizer, discount, network, distributions, entropy_regularization, tar-
          get_sync_frequency, target_update_weight, double_q_model, huber_loss)
```

```
act (states, internals, deterministic=False, independent=False, fetch_tensors=None)
```

Does a forward pass through the model to retrieve action (outputs) given inputs for state (and internal state, if applicable (e.g. RNNs))

Parameters

- **states** (*dict*) – Dict of state values (each key represents one state space component).
- **internals** (*dict*) – Dict of internal state values (each key represents one internal state component).
- **deterministic** (*bool*) – If True, will not apply exploration after actions are calculated.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).

Returns

- Actual action-outputs (batched if state input is a batch).

Return type tuple

as_local_model ()

close ()

create_act_operations (*states, internals, deterministic, independent*)

create_distributions ()

create_observe_operations (*terminal, reward*)

create_operations (*states, internals, actions, terminal, reward, deterministic, independent*)

get_component (*component_name*)

Looks up a component by its name.

Parameters **component_name** – The name of the component to look up.

Returns The component for the provided name or None if there is no such component.

get_components ()

get_feed_dict (*states=None, internals=None, actions=None, terminal=None, reward=None, deterministic=None, independent=None*)

get_savable_components ()

Returns the list of all of the components this model consists of that can be individually saved and restored. For instance the network or distribution.

Returns List of util.SavableComponent

get_summaries ()

get_variables (*include_submodules=False, include_nontrainable=False*)

import_experience (*states, internals, actions, terminal, reward*)

Stores experiences.

initialize (*custom_getter*)

observe (*terminal, reward*)

Adds an observation (reward and is-terminal) to the model without updating its trainable variables.

Parameters

- **terminal** (*bool*) – Whether the episode has terminated.
- **reward** (*float*) – The observed reward value.

Returns The value of the model-internal episode counter.

optimizer_arguments (*states, internals, actions, terminal, reward, next_states, next_internals*)

reset ()

Resets the model to its initial state on episode start. This should also reset all preprocessor(s).

Returns Current episode, timestep counter and the shallow-copied list of internal state initialization Tensors.

Return type tuple

restore (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

restore_component (*component_name*, *save_path*)

Restores a component's parameters from a save location.

Parameters

- **component_name** – The component to restore.
- **save_path** – The save location.

save (*directory=None*, *append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** – Optional checkpoint directory.
- **append_timestep** – Appends the current timestep to the checkpoint file if true.

Returns Checkpoint path where the model was saved.

save_component (*component_name*, *save_path*)

Saves a component of this model to the designated location.

Parameters

- **component_name** – The component to save.
- **save_path** – The location to save to.

Returns Checkpoint path where the component was saved.

setup ()

Sets up the TensorFlow model graph and initializes (and enters) the TensorFlow session.

target_optimizer_arguments ()

Returns the target optimizer arguments including the time, the list of variables to optimize, and various functions which the optimizer might require to perform an update step.

Returns Target optimizer arguments as dict.

tf_action_exploration (*action*, *exploration*, *action_spec*)

Applies optional exploration to the action (post-processor for action outputs).

Parameters

- **action** (*tf.Tensor*) – The original output action tensor (to be post-processed).
- **exploration** (*Exploration*) – The Exploration object to use.
- **action_spec** (*dict*) – Dict specifying the action space.

Returns The post-processed action output tensor.

tf_actions_and_internals (*states*, *internals*, *deterministic*)

tf_discounted_cumulative_reward (*terminal*, *reward*, *discount*, *final_reward=0.0*)

Creates the TensorFlow operations for calculating the discounted cumulative rewards for a given sequence of rewards.

Parameters

- **terminal** – Terminal boolean tensor.

- **reward** – Reward tensor.
- **discount** – Discount factor.
- **final_reward** – Last reward value in the sequence.

Returns Discounted cumulative reward tensor.

tf_import_experience (*states, internals, actions, terminal, reward*)

Imports experiences into the TensorFlow memory structure. Can be used to import off-policy data.

Parameters

- **states** – Dict of state values to import with keys as state names and values as values to set.
- **internals** – Internal values to set, can be fetched from agent via agent.current_internals if no values available.
- **actions** – Dict of action values to import with keys as action names and values as values to set.
- **terminal** – Terminal value(s)
- **reward** – Reward value(s)

tf_initialize ()

tf_kl_divergence (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

tf_loss (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

Creates the TensorFlow operations for calculating the full loss of a batch.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor.

tf_loss_per_instance (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

tf_observe_timestep (*states, internals, actions, terminal, reward*)

tf_optimization (*states, internals, actions, terminal, reward, next_states=None, next_internals=None*)

tf_preprocess (*states, actions, reward*)

tf_q_delta (*q_value, next_q_value, terminal, reward*)

Creates the deltas (or advantage) of the Q values.

Returns A list of deltas per action

tf_q_value (*embedding, distr_params, action, name*)

tf_reference (*states, internals, actions, terminal, reward, next_states, next_internals, update*)

Creates the TensorFlow operations for obtaining the reference tensor(s), in case of a comparative loss.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Reference tensor(s).

tf_regularization_losses (*states, internals, update*)

tensorforce.models.q_naf_model module

```
class tensorforce.models.q_naf_model.QNAFModel (states, actions, scope, de-
vice, saver, summarizer, exe-
cution, batching_capacity, vari-
able_noise, states_preprocessing,
actions_exploration, re-
ward_preprocessing, update_mode,
memory, optimizer, discount, network,
distributions, entropy_regularization,
target_sync_frequency, tar-
get_update_weight, double_q_model,
huber_loss)
```

Bases: `tensorforce.models.q_model.QModel`

COMPONENT_DISTRIBUTION = 'distribution'

COMPONENT_NETWORK = 'network'

COMPONENT_TARGET_DISTRIBUTION = 'target_distribution'

COMPONENT_TARGET_NETWORK = 'target_network'

__init__ (*states, actions, scope, device, saver, summarizer, execution, batching_capacity, variable_noise, states_preprocessing, actions_exploration, reward_preprocessing, update_mode, memory, optimizer, discount, network, distributions, entropy_regularization, target_sync_frequency, target_update_weight, double_q_model, huber_loss*)

act (*states, internals, deterministic=False, independent=False, fetch_tensors=None*)

Does a forward pass through the model to retrieve action (outputs) given inputs for state (and internal state, if applicable (e.g. RNNs))

Parameters

- **states** (*dict*) – Dict of state values (each key represents one state space component).

- **internals** (*dict*) – Dict of internal state values (each key represents one internal state component).
- **deterministic** (*bool*) – If True, will not apply exploration after actions are calculated.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).

Returns

- Actual action-outputs (batched if state input is a batch).

Return type tuple

as_local_model ()

close ()

create_act_operations (*states, internals, deterministic, independent*)

create_distributions ()

create_observe_operations (*terminal, reward*)

create_operations (*states, internals, actions, terminal, reward, deterministic, independent*)

get_component (*component_name*)

Looks up a component by its name.

Parameters **component_name** – The name of the component to look up.

Returns The component for the provided name or None if there is no such component.

get_components ()

get_feed_dict (*states=None, internals=None, actions=None, terminal=None, reward=None, deterministic=None, independent=None*)

get_savable_components ()

Returns the list of all of the components this model consists of that can be individually saved and restored.
For instance the network or distribution.

Returns List of util.SavableComponent

get_summaries ()

get_variables (*include_submodules=False, include_nontrainable=False*)

import_experience (*states, internals, actions, terminal, reward*)

Stores experiences.

initialize (*custom_getter*)

observe (*terminal, reward*)

Adds an observation (reward and is-terminal) to the model without updating its trainable variables.

Parameters

- **terminal** (*bool*) – Whether the episode has terminated.
- **reward** (*float*) – The observed reward value.

Returns The value of the model-internal episode counter.

optimizer_arguments (*states, internals, actions, terminal, reward, next_states, next_internals*)

reset ()

Resets the model to its initial state on episode start. This should also reset all preprocessor(s).

Returns Current episode, timestep counter and the shallow-copied list of internal state initialization Tensors.

Return type tuple

restore (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

restore_component (*component_name, save_path*)

Restores a component's parameters from a save location.

Parameters

- **component_name** – The component to restore.
- **save_path** – The save location.

save (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** – Optional checkpoint directory.
- **append_timestep** – Appends the current timestep to the checkpoint file if true.

Returns Checkpoint path where the model was saved.

save_component (*component_name, save_path*)

Saves a component of this model to the designated location.

Parameters

- **component_name** – The component to save.
- **save_path** – The location to save to.

Returns Checkpoint path where the component was saved.

setup ()

Sets up the TensorFlow model graph and initializes (and enters) the TensorFlow session.

target_optimizer_arguments ()

Returns the target optimizer arguments including the time, the list of variables to optimize, and various functions which the optimizer might require to perform an update step.

Returns Target optimizer arguments as dict.

tf_action_exploration (*action, exploration, action_spec*)

Applies optional exploration to the action (post-processor for action outputs).

Parameters

- **action** (*tf.Tensor*) – The original output action tensor (to be post-processed).

- **exploration** ([Exploration](#)) – The Exploration object to use.
- **action_spec** (*dict*) – Dict specifying the action space.

Returns The post-processed action output tensor.

tf_actions_and_internals (*states, internals, deterministic*)

tf_discounted_cumulative_reward (*terminal, reward, discount, final_reward=0.0*)

Creates the TensorFlow operations for calculating the discounted cumulative rewards for a given sequence of rewards.

Parameters

- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **discount** – Discount factor.
- **final_reward** – Last reward value in the sequence.

Returns Discounted cumulative reward tensor.

tf_import_experience (*states, internals, actions, terminal, reward*)

Imports experiences into the TensorFlow memory structure. Can be used to import off-policy data.

Parameters

- **states** – Dict of state values to import with keys as state names and values as values to set.
- **internals** – Internal values to set, can be fetched from agent via agent.current_internals if no values available.
- **actions** – Dict of action values to import with keys as action names and values as values to set.
- **terminal** – Terminal value(s)
- **reward** – Reward value(s)

tf_initialize ()

tf_kl_divergence (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

tf_loss (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

Creates the TensorFlow operations for calculating the full loss of a batch.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.

- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor.

tf_loss_per_instance (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

tf_observe_timestep (*states, internals, actions, terminal, reward*)

tf_optimization (*states, internals, actions, terminal, reward, next_states=None, next_internals=None*)

tf_preprocess (*states, actions, reward*)

tf_q_delta (*q_value, next_q_value, terminal, reward*)

Creates the deltas (or advantage) of the Q values.

Returns A list of deltas per action

tf_q_value (*embedding, distr_params, action, name*)

tf_reference (*states, internals, actions, terminal, reward, next_states, next_internals, update*)

Creates the TensorFlow operations for obtaining the reference tensor(s), in case of a comparative loss.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Reference tensor(s).

tf_regularization_losses (*states, internals, update*)

tensorforce.models.q_nstep_model module

```
class tensorforce.models.q_nstep_model.QNstepModel(states, actions, scope, device,
                                                    saver, summarizer, execu-
                                                    tion, batching_capacity, vari-
                                                    able_noise, states_preprocessing,
                                                    actions_exploration, re-
                                                    ward_preprocessing, up-
                                                    date_mode, memory, optimizer,
                                                    discount, network, distribu-
                                                    tions, entropy_regularization,
                                                    target_sync_frequency, tar-
                                                    get_update_weight, dou-
                                                    ble_q_model, huber_loss)
```

Bases: `tensorforce.models.q_model.QModel`

Deep Q network using n-step rewards as described in Asynchronous Methods for Reinforcement Learning.

COMPONENT_DISTRIBUTION = 'distribution'

```
COMPONENT_NETWORK = 'network'
COMPONENT_TARGET_DISTRIBUTION = 'target_distribution'
COMPONENT_TARGET_NETWORK = 'target_network'

__init__(states, actions, scope, device, saver, summarizer, execution, batching_capacity,
          variable_noise, states_preprocessing, actions_exploration, reward_preprocessing, up-
          date_mode, memory, optimizer, discount, network, distributions, entropy_regularization, tar-
          get_sync_frequency, target_update_weight, double_q_model, huber_loss)

act(states, internals, deterministic=False, independent=False, fetch_tensors=None)
    Does a forward pass through the model to retrieve action (outputs) given inputs for state (and internal state,
    if applicable (e.g. RNNs))
```

Parameters

- **states** (*dict*) – Dict of state values (each key represents one state space component).
- **internals** (*dict*) – Dict of internal state values (each key represents one internal state component).
- **deterministic** (*bool*) – If True, will not apply exploration after actions are calculated.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).

Returns

- Actual action-outputs (batched if state input is a batch).

Return type

tuple

```
as_local_model()
close()
create_act_operations(states, internals, deterministic, independent)
create_distributions()
create_observe_operations(terminal, reward)
create_operations(states, internals, actions, terminal, reward, deterministic, independent)
get_component(component_name)
    Looks up a component by its name.

    Parameters component_name – The name of the component to look up.

    Returns The component for the provided name or None if there is no such component.

get_components()
get_feed_dict(states=None, internals=None, actions=None, terminal=None, reward=None, deter-
              ministic=None, independent=None)
get_savable_components()
    Returns the list of all of the components this model consists of that can be individually saved and restored.
    For instance the network or distribution.

    Returns List of util.SavableComponent

get_summaries()
get_variables(include_submodules=False, include_nontrainable=False)
```


import_experience (*states, internals, actions, terminal, reward*)

Stores experiences.

initialize (*custom_getter*)

observe (*terminal, reward*)

Adds an observation (reward and is-terminal) to the model without updating its trainable variables.

Parameters

- **terminal** (*bool*) – Whether the episode has terminated.
- **reward** (*float*) – The observed reward value.

Returns The value of the model-internal episode counter.

optimizer_arguments (*states, internals, actions, terminal, reward, next_states, next_internals*)

reset ()

Resets the model to its initial state on episode start. This should also reset all preprocessor(s).

Returns Current episode, timestep counter and the shallow-copied list of internal state initialization Tensors.

Return type tuple

restore (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

restore_component (*component_name, save_path*)

Restores a component's parameters from a save location.

Parameters

- **component_name** – The component to restore.
- **save_path** – The save location.

save (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** – Optional checkpoint directory.
- **append_timestep** – Appends the current timestep to the checkpoint file if true.

Returns Checkpoint path where the model was saved.

save_component (*component_name, save_path*)

Saves a component of this model to the designated location.

Parameters

- **component_name** – The component to save.
- **save_path** – The location to save to.

Returns Checkpoint path where the component was saved.

setup()

Sets up the TensorFlow model graph and initializes (and enters) the TensorFlow session.

target_optimizer_arguments()

Returns the target optimizer arguments including the time, the list of variables to optimize, and various functions which the optimizer might require to perform an update step.

Returns Target optimizer arguments as dict.

tf_action_exploration(action, exploration, action_spec)

Applies optional exploration to the action (post-processor for action outputs).

Parameters

- **action** (*tf.Tensor*) – The original output action tensor (to be post-processed).
- **exploration** (*Exploration*) – The Exploration object to use.
- **action_spec** (*dict*) – Dict specifying the action space.

Returns The post-processed action output tensor.

tf_actions_and_internals(states, internals, deterministic)

tf_discounted_cumulative_reward(terminal, reward, discount, final_reward=0.0)

Creates the TensorFlow operations for calculating the discounted cumulative rewards for a given sequence of rewards.

Parameters

- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **discount** – Discount factor.
- **final_reward** – Last reward value in the sequence.

Returns Discounted cumulative reward tensor.

tf_import_experience(states, internals, actions, terminal, reward)

Imports experiences into the TensorFlow memory structure. Can be used to import off-policy data.

Parameters

- **states** – Dict of state values to import with keys as state names and values as values to set.
- **internals** – Internal values to set, can be fetched from agent via agent.current_internals if no values available.
- **actions** – Dict of action values to import with keys as action names and values as values to set.
- **terminal** – Terminal value(s)
- **reward** – Reward value(s)

tf_initialize()

tf_kl_divergence(states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None)

tf_loss(states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None)

Creates the TensorFlow operations for calculating the full loss of a batch.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor.

tf_loss_per_instance (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

tf_observe_timestep (*states, internals, actions, terminal, reward*)

tf_optimization (*states, internals, actions, terminal, reward, next_states=None, next_internals=None*)

tf_preprocess (*states, actions, reward*)

tf_q_delta (*q_value, next_q_value, terminal, reward*)

tf_q_value (*embedding, distr_params, action, name*)

tf_reference (*states, internals, actions, terminal, reward, next_states, next_internals, update*)

Creates the TensorFlow operations for obtaining the reference tensor(s), in case of a comparative loss.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Reference tensor(s).

tf_regularization_losses (*states, internals, update*)

tensorforce.models.random_model module

class tensorforce.models.random_model.**RandomModel** (*states, actions, scope, device, saver, summarizer, execution, batching_capacity*)

Bases: *tensorforce.models.model.Model*

Utility class to return random actions of a desired shape and with given bounds.

__init__ (*states, actions, scope, device, saver, summarizer, execution, batching_capacity*)

act (*states, internals, deterministic=False, independent=False, fetch_tensors=None*)

Does a forward pass through the model to retrieve action (outputs) given inputs for state (and internal state, if applicable (e.g. RNNs))

Parameters

- **states** (*dict*) – Dict of state values (each key represents one state space component).
- **internals** (*dict*) – Dict of internal state values (each key represents one internal state component).
- **deterministic** (*bool*) – If True, will not apply exploration after actions are calculated.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).

Returns

- Actual action-outputs (batched if state input is a batch).

Return type

tuple

as_local_model ()

close ()

create_act_operations (*states, internals, deterministic, independent*)

create_observe_operations (*terminal, reward*)

create_operations (*states, internals, actions, terminal, reward, deterministic, independent*)

Creates output operations for acting, observing and interacting with the memory.

get_component (*component_name*)

Looks up a component by its name.

Parameters **component_name** – The name of the component to look up.

Returns The component for the provided name or None if there is no such component.

get_components ()

Returns a dictionary of component name to component of all the components within this model.

Returns (dict) The mapping of name to component.

get_feed_dict (*states=None, internals=None, actions=None, terminal=None, reward=None, deterministic=None, independent=None*)

get_savable_components ()

Returns the list of all of the components this model consists of that can be individually saved and restored. For instance the network or distribution.

Returns List of util.SavableComponent

get_summaries ()

Returns the TensorFlow summaries reported by the model

Returns List of summaries

get_variables (*include_submodules=False, include_nontrainable=False*)

Returns the TensorFlow variables used by the model.

Parameters

- **include_submodules** – Includes variables of submodules (e.g. baseline, target network) if true.
- **include_nontrainable** – Includes non-trainable variables if true.

Returns List of variables.

initialize (*custom_getter*)

Creates the TensorFlow placeholders and functions for this model. Moreover adds the internal state placeholders and initialization values to the model.

Parameters **custom_getter** – The `custom_getter_` object to use for `tf.make_template` when creating TensorFlow functions.

observe (*terminal, reward*)

Adds an observation (reward and is-terminal) to the model without updating its trainable variables.

Parameters

- **terminal** (*bool*) – Whether the episode has terminated.
- **reward** (*float*) – The observed reward value.

Returns The value of the model-internal episode counter.

reset ()

Resets the model to its initial state on episode start. This should also reset all preprocessor(s).

Returns Current episode, timestep counter and the shallow-copied list of internal state initialization Tensors.

Return type tuple

restore (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

restore_component (*component_name, save_path*)

Restores a component's parameters from a save location.

Parameters

- **component_name** – The component to restore.
- **save_path** – The save location.

save (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** – Optional checkpoint directory.
- **append_timestep** – Appends the current timestep to the checkpoint file if true.

Returns Checkpoint path where the model was saved.

save_component (*component_name, save_path*)

Saves a component of this model to the designated location.

Parameters

- **component_name** – The component to save.
- **save_path** – The location to save to.

Returns Checkpoint path where the component was saved.

setup()

Sets up the TensorFlow model graph and initializes (and enters) the TensorFlow session.

tf_action_exploration(*action, exploration, action_spec*)

Applies optional exploration to the action (post-processor for action outputs).

Parameters

- **action** (*tf.Tensor*) – The original output action tensor (to be post-processed).
- **exploration** (*Exploration*) – The Exploration object to use.
- **action_spec** (*dict*) – Dict specifying the action space.

Returns The post-processed action output tensor.

tf_actions_and_internals(*states, internals, deterministic*)**tf_initialize()****tf_observe_timestep**(*states, internals, actions, terminal, reward*)**tf_preprocess**(*states, actions, reward*)**Module contents**

```
class tensorflow.models.Model(states, actions, scope, device, saver, summarizer, execution,  
                               batching_capacity, variable_noise, states_preprocessing, ac-  
                               tions_exploration, reward_preprocessing)
```

Bases: object

Base class for all (TensorFlow-based) models.

```
__init__(states, actions, scope, device, saver, summarizer, execution, batching_capacity, vari-  
         able_noise, states_preprocessing, actions_exploration, reward_preprocessing)  
Model.
```

Parameters

- **states** (*spec*) – The state-space description dictionary.
- **actions** (*spec*) – The action-space description dictionary.
- **scope** (*str*) – The root scope str to use for tf variable scoping.
- **device** (*str*) – The name of the device to run the graph of this model on.
- **saver** (*spec*) – Dict specifying whether and how to save the model's parameters.
- **summarizer** (*spec*) – Dict specifying which tensorboard summaries should be created and added to the graph.
- **execution** (*spec*) – Dict specifying whether and how to do distributed training on the model's graph.
- **batching_capacity** (*int*) – Batching capacity.

- **variable_noise** (*float*) – The stddev value of a Normal distribution used for adding random noise to the model’s output (for each batch, noise can be toggled and - if active - will be resampled). Use None for not adding any noise.
- **states_preprocessing** (*spec / dict of specs*) – Dict specifying whether and how to preprocess state signals (e.g. normalization, greyscale, etc..).
- **actions_exploration** (*spec / dict of specs*) – Dict specifying whether and how to add exploration to the model’s “action outputs” (e.g. epsilon-greedy).
- **reward_preprocessing** (*spec*) – Dict specifying whether and how to preprocess rewards coming from the Environment (e.g. reward normalization).

act (*states, internals, deterministic=False, independent=False, fetch_tensors=None*)

Does a forward pass through the model to retrieve action (outputs) given inputs for state (and internal state, if applicable (e.g. RNNs))

Parameters

- **states** (*dict*) – Dict of state values (each key represents one state space component).
- **internals** (*dict*) – Dict of internal state values (each key represents one internal state component).
- **deterministic** (*bool*) – If True, will not apply exploration after actions are calculated.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).

Returns

- Actual action-outputs (batched if state input is a batch).

Return type tuple

as_local_model ()

close ()

create_act_operations (*states, internals, deterministic, independent*)

create_observe_operations (*terminal, reward*)

create_operations (*states, internals, actions, terminal, reward, deterministic, independent*)

Creates output operations for acting, observing and interacting with the memory.

get_component (*component_name*)

Looks up a component by its name.

Parameters **component_name** – The name of the component to look up.

Returns The component for the provided name or None if there is no such component.

get_components ()

Returns a dictionary of component name to component of all the components within this model.

Returns (dict) The mapping of name to component.

get_feed_dict (*states=None, internals=None, actions=None, terminal=None, reward=None, deterministic=None, independent=None*)

get_savable_components ()

Returns the list of all of the components this model consists of that can be individually saved and restored. For instance the network or distribution.

Returns List of `util.SavableComponent`

get_summaries()

Returns the TensorFlow summaries reported by the model

Returns List of summaries

get_variables (*include_submodules=False, include_nontrainable=False*)

Returns the TensorFlow variables used by the model.

Parameters

- **include_submodules** – Includes variables of submodules (e.g. baseline, target network) if true.
- **include_nontrainable** – Includes non-trainable variables if true.

Returns List of variables.

initialize (*custom_getter*)

Creates the TensorFlow placeholders and functions for this model. Moreover adds the internal state placeholders and initialization values to the model.

Parameters **custom_getter** – The `custom_getter_` object to use for `tf.make_template` when creating TensorFlow functions.

observe (*terminal, reward*)

Adds an observation (reward and is-terminal) to the model without updating its trainable variables.

Parameters

- **terminal** (*bool*) – Whether the episode has terminated.
- **reward** (*float*) – The observed reward value.

Returns The value of the model-internal episode counter.

reset ()

Resets the model to its initial state on episode start. This should also reset all preprocessor(s).

Returns Current episode, timestep counter and the shallow-copied list of internal state initialization Tensors.

Return type tuple

restore (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

restore_component (*component_name, save_path*)

Restores a component's parameters from a save location.

Parameters

- **component_name** – The component to restore.
- **save_path** – The save location.

save (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used.

Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** – Optional checkpoint directory.
- **append_timestep** – Appends the current timestep to the checkpoint file if true.

Returns Checkpoint path where the model was saved.

save_component (*component_name*, *save_path*)

Saves a component of this model to the designated location.

Parameters

- **component_name** – The component to save.
- **save_path** – The location to save to.

Returns Checkpoint path where the component was saved.

setup ()

Sets up the TensorFlow model graph and initializes (and enters) the TensorFlow session.

tf_action_exploration (*action*, *exploration*, *action_spec*)

Applies optional exploration to the action (post-processor for action outputs).

Parameters

- **action** (*tf.Tensor*) – The original output action tensor (to be post-processed).
- **exploration** (*Exploration*) – The Exploration object to use.
- **action_spec** (*dict*) – Dict specifying the action space.

Returns The post-processed action output tensor.

tf_actions_and_internals (*states*, *internals*, *deterministic*)

Creates and returns the TensorFlow operations for retrieving the actions and - if applicable - the posterior internal state Tensors in reaction to the given input states (and prior internal states).

Parameters

- **states** (*dict*) – Dict of state tensors (each key represents one state space component).
- **internals** – List of prior internal state tensors.
- **deterministic** – Boolean tensor indicating whether action should be chosen deterministically.

Returns

1. dict of output actions (with or without exploration applied (see *deterministic*))
2. list of posterior internal state Tensors (empty for non-internal state models)

Return type tuple

tf_initialize ()

tf_observe_timestep (*states*, *internals*, *actions*, *terminal*, *reward*)

Creates the TensorFlow operations for performing the observation of a full time step's information.

Parameters

- **states** (*dict*) – Dict of state tensors (each key represents one state space component).
- **internals** – List of prior internal state tensors.

- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.

Returns The observation operation.

tf_preprocess (*states, actions, reward*)

```
class tensorflow.models.MemoryModel (states, actions, scope, device, saver, summarizer,  
                                     execution, batching_capacity, variable_noise,  
                                     states_preprocessing, actions_exploration, re-  
                                     ward_preprocessing, update_mode, memory, optimizer,  
                                     discount)
```

Bases: `tensorflow.models.model.Model`

A memory model is a generical model to accumulate and sample data.

```
__init__ (states, actions, scope, device, saver, summarizer, execution, batching_capacity,  
          variable_noise, states_preprocessing, actions_exploration, reward_preprocessing, up-  
          date_mode, memory, optimizer, discount)
```

Memory model.

Parameters

- **states** (*spec*) – The state-space description dictionary.
- **actions** (*spec*) – The action-space description dictionary.
- **scope** (*str*) – The root scope str to use for tf variable scoping.
- **device** (*str*) – The name of the device to run the graph of this model on.
- **saver** (*spec*) – Dict specifying whether and how to save the model’s parameters.
- **summarizer** (*spec*) – Dict specifying which tensorboard summaries should be created and added to the graph.
- **execution** (*spec*) – Dict specifying whether and how to do distributed training on the model’s graph.
- **batching_capacity** (*int*) – Batching capacity.
- **variable_noise** (*float*) – The stddev value of a Normal distribution used for adding random noise to the model’s output (for each batch, noise can be toggled and - if active - will be resampled). Use None for not adding any noise.
- **states_preprocessing** (*spec / dict of specs*) – Dict specifying whether and how to preprocess state signals (e.g. normalization, greyscale, etc..).
- **actions_exploration** (*spec / dict of specs*) – Dict specifying whether and how to add exploration to the model’s “action outputs” (e.g. epsilon-greedy).
- **reward_preprocessing** (*spec*) – Dict specifying whether and how to preprocess rewards coming from the Environment (e.g. reward normalization).
- **update_mode** (*spec*) – Update mode.
- **memory** (*spec*) – Memory.
- **optimizer** (*spec*) – Dict specifying the tf optimizer to use for tuning the model’s trainable parameters.
- **discount** (*float*) – The RL reward discount factor (gamma).

act (*states, internals, deterministic=False, independent=False, fetch_tensors=None*)

Does a forward pass through the model to retrieve action (outputs) given inputs for state (and internal state, if applicable (e.g. RNNs))

Parameters

- **states** (*dict*) – Dict of state values (each key represents one state space component).
- **internals** (*dict*) – Dict of internal state values (each key represents one internal state component).
- **deterministic** (*bool*) – If True, will not apply exploration after actions are calculated.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).

Returns

- Actual action-outputs (batched if state input is a batch).

Return type tuple

as_local_model ()

close ()

create_act_operations (*states, internals, deterministic, independent*)

create_observe_operations (*terminal, reward*)

create_operations (*states, internals, actions, terminal, reward, deterministic, independent*)

get_component (*component_name*)

Looks up a component by its name.

Parameters **component_name** – The name of the component to look up.

Returns The component for the provided name or None if there is no such component.

get_components ()

Returns a dictionary of component name to component of all the components within this model.

Returns (dict) The mapping of name to component.

get_feed_dict (*states=None, internals=None, actions=None, terminal=None, reward=None, deterministic=None, independent=None*)

get_savable_components ()

Returns the list of all of the components this model consists of that can be individually saved and restored. For instance the network or distribution.

Returns List of util.SavableComponent

get_summaries ()

get_variables (*include_submodules=False, include_nontrainable=False*)

import_experience (*states, internals, actions, terminal, reward*)

Stores experiences.

initialize (*custom_getter*)

observe (*terminal, reward*)

Adds an observation (reward and is-terminal) to the model without updating its trainable variables.

Parameters

- **terminal** (*bool*) – Whether the episode has terminated.
- **reward** (*float*) – The observed reward value.

Returns The value of the model-internal episode counter.

optimizer_arguments (*states, internals, actions, terminal, reward, next_states, next_internals*)

Returns the optimizer arguments including the time, the list of variables to optimize, and various functions which the optimizer might require to perform an update step.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.

Returns Optimizer arguments as dict.

reset ()

Resets the model to its initial state on episode start. This should also reset all preprocessor(s).

Returns Current episode, timestep counter and the shallow-copied list of internal state initialization Tensors.

Return type tuple

restore (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

restore_component (*component_name, save_path*)

Restores a component's parameters from a save location.

Parameters

- **component_name** – The component to restore.
- **save_path** – The save location.

save (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** – Optional checkpoint directory.
- **append_timestep** – Appends the current timestep to the checkpoint file if true.

Returns Checkpoint path where the model was saved.

save_component (*component_name*, *save_path*)

Saves a component of this model to the designated location.

Parameters

- **component_name** – The component to save.
- **save_path** – The location to save to.

Returns Checkpoint path where the component was saved.

setup ()

Sets up the TensorFlow model graph and initializes (and enters) the TensorFlow session.

tf_action_exploration (*action*, *exploration*, *action_spec*)

Applies optional exploration to the action (post-processor for action outputs).

Parameters

- **action** (*tf.Tensor*) – The original output action tensor (to be post-processed).
- **exploration** (*Exploration*) – The Exploration object to use.
- **action_spec** (*dict*) – Dict specifying the action space.

Returns The post-processed action output tensor.

tf_actions_and_internals (*states*, *internals*, *deterministic*)

Creates and returns the TensorFlow operations for retrieving the actions and - if applicable - the posterior internal state Tensors in reaction to the given input states (and prior internal states).

Parameters

- **states** (*dict*) – Dict of state tensors (each key represents one state space component).
- **internals** – List of prior internal state tensors.
- **deterministic** – Boolean tensor indicating whether action should be chosen deterministically.

Returns

1. dict of output actions (with or without exploration applied (see *deterministic*))
2. list of posterior internal state Tensors (empty for non-internal state models)

Return type tuple

tf_discounted_cumulative_reward (*terminal*, *reward*, *discount*, *final_reward=0.0*)

Creates the TensorFlow operations for calculating the discounted cumulative rewards for a given sequence of rewards.

Parameters

- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **discount** – Discount factor.
- **final_reward** – Last reward value in the sequence.

Returns Discounted cumulative reward tensor.

tf_import_experience (*states*, *internals*, *actions*, *terminal*, *reward*)

Imports experiences into the TensorFlow memory structure. Can be used to import off-policy data.

Parameters

- **states** – Dict of state values to import with keys as state names and values as values to set.
- **internals** – Internal values to set, can be fetched from agent via `agent.current_internals` if no values available.
- **actions** – Dict of action values to import with keys as action names and values as values to set.
- **terminal** – Terminal value(s)
- **reward** – Reward value(s)

tf_initialize()

tf_loss (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

Creates the TensorFlow operations for calculating the full loss of a batch.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor.

tf_loss_per_instance (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

Creates the TensorFlow operations for calculating the loss per batch instance.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss per instance tensor.

tf_observe_timestep (*states, internals, actions, terminal, reward*)

tf_optimization (*states, internals, actions, terminal, reward, next_states=None, next_internals=None*)

Creates the TensorFlow operations for performing an optimization update step based on the given input states and actions batch.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.

Returns The optimization operation.

tf_preprocess (*states, actions, reward*)

tf_reference (*states, internals, actions, terminal, reward, next_states, next_internals, update*)

Creates the TensorFlow operations for obtaining the reference tensor(s), in case of a comparative loss.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Reference tensor(s).

tf_regularization_losses (*states, internals, update*)

Creates the TensorFlow operations for calculating the regularization losses for the given input states.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Dict of regularization loss tensors.

class tensorflow.models.**DistributionModel** (*states, actions, scope, device, saver, summarizer, execution, batching_capacity, variable_noise, states_preprocessing, actions_exploration, reward_preprocessing, update_mode, memory, optimizer, discount, network, distributions, entropy_regularization, requires_deterministic*)

Bases: `tensorforce.models.memory_model.MemoryModel`

Base class for models using distributions parametrized by a neural network.

COMPONENT_DISTRIBUTION = `'distribution'`

COMPONENT_NETWORK = `'network'`

__init__(*states, actions, scope, device, saver, summarizer, execution, batching_capacity, variable_noise, states_preprocessing, actions_exploration, reward_preprocessing, update_mode, memory, optimizer, discount, network, distributions, entropy_regularization, requires_deterministic*)

act(*states, internals, deterministic=False, independent=False, fetch_tensors=None*)

Does a forward pass through the model to retrieve action (outputs) given inputs for state (and internal state, if applicable (e.g. RNNs))

Parameters

- **states** (*dict*) – Dict of state values (each key represents one state space component).
- **internals** (*dict*) – Dict of internal state values (each key represents one internal state component).
- **deterministic** (*bool*) – If True, will not apply exploration after actions are calculated.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).

Returns

- Actual action-outputs (batched if state input is a batch).

Return type `tuple`

as_local_model()

close()

create_act_operations(*states, internals, deterministic, independent*)

create_distributions()

create_observe_operations(*terminal, reward*)

create_operations(*states, internals, actions, terminal, reward, deterministic, independent*)

get_component(*component_name*)

Looks up a component by its name.

Parameters **component_name** – The name of the component to look up.

Returns The component for the provided name or None if there is no such component.

get_components()

get_feed_dict(*states=None, internals=None, actions=None, terminal=None, reward=None, deterministic=None, independent=None*)

get_savable_components()

Returns the list of all of the components this model consists of that can be individually saved and restored. For instance the network or distribution.

Returns List of `util.SavableComponent`

get_summaries()

get_variables (*include_submodules=False, include_nontrainable=False*)

import_experience (*states, internals, actions, terminal, reward*)

Stores experiences.

initialize (*custom_getter*)

observe (*terminal, reward*)

Adds an observation (reward and is-terminal) to the model without updating its trainable variables.

Parameters

- **terminal** (*bool*) – Whether the episode has terminated.
- **reward** (*float*) – The observed reward value.

Returns The value of the model-internal episode counter.

optimizer_arguments (*states, internals, actions, terminal, reward, next_states, next_internals*)

reset ()

Resets the model to its initial state on episode start. This should also reset all preprocessor(s).

Returns Current episode, timestep counter and the shallow-copied list of internal state initialization Tensors.

Return type tuple

restore (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

restore_component (*component_name, save_path*)

Restores a component's parameters from a save location.

Parameters

- **component_name** – The component to restore.
- **save_path** – The save location.

save (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** – Optional checkpoint directory.
- **append_timestep** – Appends the current timestep to the checkpoint file if true.

Returns Checkpoint path where the model was saved.

save_component (*component_name, save_path*)

Saves a component of this model to the designated location.

Parameters

- **component_name** – The component to save.
- **save_path** – The location to save to.

Returns Checkpoint path where the component was saved.

setup()

Sets up the TensorFlow model graph and initializes (and enters) the TensorFlow session.

tf_action_exploration(*action, exploration, action_spec*)

Applies optional exploration to the action (post-processor for action outputs).

Parameters

- **action** (*tf.Tensor*) – The original output action tensor (to be post-processed).
- **exploration** (*Exploration*) – The Exploration object to use.
- **action_spec** (*dict*) – Dict specifying the action space.

Returns The post-processed action output tensor.

tf_actions_and_internals(*states, internals, deterministic*)

tf_discounted_cumulative_reward(*terminal, reward, discount, final_reward=0.0*)

Creates the TensorFlow operations for calculating the discounted cumulative rewards for a given sequence of rewards.

Parameters

- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **discount** – Discount factor.
- **final_reward** – Last reward value in the sequence.

Returns Discounted cumulative reward tensor.

tf_import_experience(*states, internals, actions, terminal, reward*)

Imports experiences into the TensorFlow memory structure. Can be used to import off-policy data.

Parameters

- **states** – Dict of state values to import with keys as state names and values as values to set.
- **internals** – Internal values to set, can be fetched from agent via agent.current_internals if no values available.
- **actions** – Dict of action values to import with keys as action names and values as values to set.
- **terminal** – Terminal value(s)
- **reward** – Reward value(s)

tf_initialize()

tf_kl_divergence(*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

tf_loss(*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

Creates the TensorFlow operations for calculating the full loss of a batch.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.

- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor.

tf_loss_per_instance (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

Creates the TensorFlow operations for calculating the loss per batch instance.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss per instance tensor.

tf_observe_timestep (*states, internals, actions, terminal, reward*)

tf_optimization (*states, internals, actions, terminal, reward, next_states=None, next_internals=None*)

Creates the TensorFlow operations for performing an optimization update step based on the given input states and actions batch.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.

Returns The optimization operation.

tf_preprocess (*states, actions, reward*)

tf_reference (*states, internals, actions, terminal, reward, next_states, next_internals, update*)
Creates the TensorFlow operations for obtaining the reference tensor(s), in case of a comparative loss.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Reference tensor(s).

tf_regularization_losses (*states, internals, update*)

class `tensorforce.models.PGModel` (*states, actions, scope, device, saver, summarizer, execution, batching_capacity, variable_noise, states_preprocessing, actions_exploration, reward_preprocessing, update_mode, memory, optimizer, discount, network, distributions, entropy_regularization, baseline_mode, baseline, baseline_optimizer, gae_lambda*)

Bases: `tensorforce.models.distribution_model.DistributionModel`

Base class for policy gradient models. It optionally defines a baseline and handles its optimization. It implements the `tf_loss_per_instance` function, but requires subclasses to implement `tf_pg_loss_per_instance`.

COMPONENT_BASELINE = 'baseline'

COMPONENT_DISTRIBUTION = 'distribution'

COMPONENT_NETWORK = 'network'

__init__ (*states, actions, scope, device, saver, summarizer, execution, batching_capacity, variable_noise, states_preprocessing, actions_exploration, reward_preprocessing, update_mode, memory, optimizer, discount, network, distributions, entropy_regularization, baseline_mode, baseline, baseline_optimizer, gae_lambda*)

act (*states, internals, deterministic=False, independent=False, fetch_tensors=None*)

Does a forward pass through the model to retrieve action (outputs) given inputs for state (and internal state, if applicable (e.g. RNNs))

Parameters

- **states** (*dict*) – Dict of state values (each key represents one state space component).
- **internals** (*dict*) – Dict of internal state values (each key represents one internal state component).
- **deterministic** (*bool*) – If True, will not apply exploration after actions are calculated.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).

Returns

- Actual action-outputs (batched if state input is a batch).

Return type tuple

as_local_model ()

baseline_optimizer_arguments (*states, internals, reward*)

Returns the baseline optimizer arguments including the time, the list of variables to optimize, and various functions which the optimizer might require to perform an update step.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.

Returns Baseline optimizer arguments as dict.

close ()

create_act_operations (*states, internals, deterministic, independent*)

create_distributions ()

create_observe_operations (*terminal, reward*)

create_operations (*states, internals, actions, terminal, reward, deterministic, independent*)

get_component (*component_name*)

Looks up a component by its name.

Parameters **component_name** – The name of the component to look up.

Returns The component for the provided name or None if there is no such component.

get_components ()

get_feed_dict (*states=None, internals=None, actions=None, terminal=None, reward=None, deterministic=None, independent=None*)

get_savable_components ()

Returns the list of all of the components this model consists of that can be individually saved and restored. For instance the network or distribution.

Returns List of util.SavableComponent

get_summaries ()

get_variables (*include_submodules=False, include_nontrainable=False*)

import_experience (*states, internals, actions, terminal, reward*)

Stores experiences.

initialize (*custom_getter*)

observe (*terminal, reward*)

Adds an observation (reward and is-terminal) to the model without updating its trainable variables.

Parameters

- **terminal** (*bool*) – Whether the episode has terminated.
- **reward** (*float*) – The observed reward value.

Returns The value of the model-internal episode counter.

optimizer_arguments (*states, internals, actions, terminal, reward, next_states, next_internals*)

reset ()

Resets the model to its initial state on episode start. This should also reset all preprocessor(s).

Returns Current episode, timestep counter and the shallow-copied list of internal state initialization Tensors.

Return type tuple

restore (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

restore_component (*component_name, save_path*)

Restores a component's parameters from a save location.

Parameters

- **component_name** – The component to restore.
- **save_path** – The save location.

save (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** – Optional checkpoint directory.
- **append_timestep** – Appends the current timestep to the checkpoint file if true.

Returns Checkpoint path where the model was saved.

save_component (*component_name, save_path*)

Saves a component of this model to the designated location.

Parameters

- **component_name** – The component to save.
- **save_path** – The location to save to.

Returns Checkpoint path where the component was saved.

setup ()

Sets up the TensorFlow model graph and initializes (and enters) the TensorFlow session.

tf_action_exploration (*action, exploration, action_spec*)

Applies optional exploration to the action (post-processor for action outputs).

Parameters

- **action** (*tf.Tensor*) – The original output action tensor (to be post-processed).
- **exploration** (*Exploration*) – The Exploration object to use.
- **action_spec** (*dict*) – Dict specifying the action space.

Returns The post-processed action output tensor.

tf_actions_and_internals (*states, internals, deterministic*)

tf_baseline_loss (*states, internals, reward, update, reference=None*)

Creates the TensorFlow operations for calculating the baseline loss of a batch.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor.

tf_discounted_cumulative_reward (*terminal, reward, discount, final_reward=0.0*)

Creates the TensorFlow operations for calculating the discounted cumulative rewards for a given sequence of rewards.

Parameters

- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **discount** – Discount factor.
- **final_reward** – Last reward value in the sequence.

Returns Discounted cumulative reward tensor.

tf_import_experience (*states, internals, actions, terminal, reward*)

Imports experiences into the TensorFlow memory structure. Can be used to import off-policy data.

Parameters

- **states** – Dict of state values to import with keys as state names and values as values to set.
- **internals** – Internal values to set, can be fetched from agent via agent.current_internals if no values available.
- **actions** – Dict of action values to import with keys as action names and values as values to set.
- **terminal** – Terminal value(s)
- **reward** – Reward value(s)

tf_initialize ()

tf_kl_divergence (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

tf_loss (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

Creates the TensorFlow operations for calculating the full loss of a batch.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.

- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor.

tf_loss_per_instance (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

Creates the TensorFlow operations for calculating the loss per batch instance.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss per instance tensor.

tf_observe_timestep (*states, internals, actions, terminal, reward*)

tf_optimization (*states, internals, actions, terminal, reward, next_states=None, next_internals=None*)

tf_preprocess (*states, actions, reward*)

tf_reference (*states, internals, actions, terminal, reward, next_states, next_internals, update*)

Creates the TensorFlow operations for obtaining the reference tensor(s), in case of a comparative loss.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Reference tensor(s).

tf_regularization_losses (*states, internals, update*)

tf_reward_estimation (*states, internals, terminal, reward, update*)

```
class tensorflow.models.PGProbRatioModel(states, actions, scope, device, saver, summarizer, execution, batching_capacity, variable_noise, states_preprocessing, actions_exploration, reward_preprocessing, update_mode, memory, optimizer, discount, network, distributions, entropy_regularization, baseline_mode, baseline, baseline_optimizer, gae_lambda, likelihood_ratio_clipping)
```

Bases: *tensorflow.models.pg_model.PGModel*

Policy gradient model based on computing likelihood ratios, e.g. TRPO and PPO.

COMPONENT_BASELINE = 'baseline'

COMPONENT_DISTRIBUTION = 'distribution'

COMPONENT_NETWORK = 'network'

```
__init__(states, actions, scope, device, saver, summarizer, execution, batching_capacity, variable_noise, states_preprocessing, actions_exploration, reward_preprocessing, update_mode, memory, optimizer, discount, network, distributions, entropy_regularization, baseline_mode, baseline, baseline_optimizer, gae_lambda, likelihood_ratio_clipping)
```

act (*states, internals, deterministic=False, independent=False, fetch_tensors=None*)

Does a forward pass through the model to retrieve action (outputs) given inputs for state (and internal state, if applicable (e.g. RNNs))

Parameters

- **states** (*dict*) – Dict of state values (each key represents one state space component).
- **internals** (*dict*) – Dict of internal state values (each key represents one internal state component).
- **deterministic** (*bool*) – If True, will not apply exploration after actions are calculated.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).

Returns

- Actual action-outputs (batched if state input is a batch).

Return type

tuple

```
as_local_model()
```

baseline_optimizer_arguments (*states, internals, reward*)

Returns the baseline optimizer arguments including the time, the list of variables to optimize, and various functions which the optimizer might require to perform an update step.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.

Returns Baseline optimizer arguments as dict.

```
close()
```

create_act_operations (*states, internals, deterministic, independent*)

create_distributions ()

create_observe_operations (*terminal, reward*)

create_operations (*states, internals, actions, terminal, reward, deterministic, independent*)

get_component (*component_name*)

Looks up a component by its name.

Parameters **component_name** – The name of the component to look up.

Returns The component for the provided name or None if there is no such component.

get_components ()

get_feed_dict (*states=None, internals=None, actions=None, terminal=None, reward=None, deterministic=None, independent=None*)

get_savable_components ()

Returns the list of all of the components this model consists of that can be individually saved and restored. For instance the network or distribution.

Returns List of util.SavableComponent

get_summaries ()

get_variables (*include_submodules=False, include_nontrainable=False*)

import_experience (*states, internals, actions, terminal, reward*)

Stores experiences.

initialize (*custom_getter*)

observe (*terminal, reward*)

Adds an observation (reward and is-terminal) to the model without updating its trainable variables.

Parameters

- **terminal** (*bool*) – Whether the episode has terminated.
- **reward** (*float*) – The observed reward value.

Returns The value of the model-internal episode counter.

optimizer_arguments (*states, internals, actions, terminal, reward, next_states, next_internals*)

reset ()

Resets the model to its initial state on episode start. This should also reset all preprocessor(s).

Returns Current episode, timestep counter and the shallow-copied list of internal state initialization Tensors.

Return type tuple

restore (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

restore_component (*component_name, save_path*)

Restores a component's parameters from a save location.

Parameters

- **component_name** – The component to restore.
- **save_path** – The save location.

save (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** – Optional checkpoint directory.
- **append_timestep** – Appends the current timestep to the checkpoint file if true.

Returns Checkpoint path where the model was saved.

save_component (*component_name, save_path*)

Saves a component of this model to the designated location.

Parameters

- **component_name** – The component to save.
- **save_path** – The location to save to.

Returns Checkpoint path where the component was saved.

setup ()

Sets up the TensorFlow model graph and initializes (and enters) the TensorFlow session.

tf_action_exploration (*action, exploration, action_spec*)

Applies optional exploration to the action (post-processor for action outputs).

Parameters

- **action** (*tf.Tensor*) – The original output action tensor (to be post-processed).
- **exploration** (*Exploration*) – The Exploration object to use.
- **action_spec** (*dict*) – Dict specifying the action space.

Returns The post-processed action output tensor.

tf_actions_and_internals (*states, internals, deterministic*)

tf_baseline_loss (*states, internals, reward, update, reference=None*)

Creates the TensorFlow operations for calculating the baseline loss of a batch.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor.

tf_discounted_cumulative_reward (*terminal, reward, discount, final_reward=0.0*)

Creates the TensorFlow operations for calculating the discounted cumulative rewards for a given sequence of rewards.

Parameters

- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **discount** – Discount factor.
- **final_reward** – Last reward value in the sequence.

Returns Discounted cumulative reward tensor.

tf_import_experience (*states, internals, actions, terminal, reward*)

Imports experiences into the TensorFlow memory structure. Can be used to import off-policy data.

Parameters

- **states** – Dict of state values to import with keys as state names and values as values to set.
- **internals** – Internal values to set, can be fetched from agent via agent.current_internals if no values available.
- **actions** – Dict of action values to import with keys as action names and values as values to set.
- **terminal** – Terminal value(s)
- **reward** – Reward value(s)

tf_initialize ()

tf_kl_divergence (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

tf_loss (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

Creates the TensorFlow operations for calculating the full loss of a batch.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor.

tf_loss_per_instance (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

tf_observe_timestep (*states, internals, actions, terminal, reward*)

tf_optimization (*states, internals, actions, terminal, reward, next_states=None, next_internals=None*)

tf_preprocess (*states, actions, reward*)

```

tf_reference (states, internals, actions, terminal, reward, next_states, next_internals, update)

tf_regularization_losses (states, internals, update)

tf_reward_estimation (states, internals, terminal, reward, update)

class tensorflow.models.DPGTargetModel (states, actions, scope, device, saver, summarizer,
                                         execution, batching_capacity, variable_noise,
                                         states_preprocessing, actions_exploration,
                                         reward_preprocessing, update_mode, mem-
                                         ory, optimizer, discount, network, distribu-
                                         tions, entropy_regularization, critic_network,
                                         critic_optimizer, target_sync_frequency, tar-
                                         get_update_weight)

Bases: tensorflow.models.distribution_model.DistributionModel

Policy gradient model log likelihood model with target network (e.g. DDPG)

COMPONENT_CRITIC = 'critic'

COMPONENT_DISTRIBUTION = 'distribution'

COMPONENT_NETWORK = 'network'

COMPONENT_TARGET_DISTRIBUTION = 'target_distribution'

COMPONENT_TARGET_NETWORK = 'target_network'

__init__ (states, actions, scope, device, saver, summarizer, execution, batching_capacity,
          variable_noise, states_preprocessing, actions_exploration, reward_preprocessing, up-
          date_mode, memory, optimizer, discount, network, distributions, entropy_regularization,
          critic_network, critic_optimizer, target_sync_frequency, target_update_weight)

act (states, internals, deterministic=False, independent=False, fetch_tensors=None)
    Does a forward pass through the model to retrieve action (outputs) given inputs for state (and internal state,
    if applicable (e.g. RNNs))

```

Parameters

- **states** (*dict*) – Dict of state values (each key represents one state space component).
- **internals** (*dict*) – Dict of internal state values (each key represents one internal state component).
- **deterministic** (*bool*) – If True, will not apply exploration after actions are calculated.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).

Returns

- Actual action-outputs (batched if state input is a batch).

Return type

```

as_local_model ()

close ()

create_act_operations (states, internals, deterministic, independent)

create_distributions ()

create_observe_operations (terminal, reward)

```

create_operations (*states, internals, actions, terminal, reward, deterministic, independent*)

get_component (*component_name*)

Looks up a component by its name.

Parameters **component_name** – The name of the component to look up.

Returns The component for the provided name or None if there is no such component.

get_components ()

get_feed_dict (*states=None, internals=None, actions=None, terminal=None, reward=None, deterministic=None, independent=None*)

get_savable_components ()

Returns the list of all of the components this model consists of that can be individually saved and restored. For instance the network or distribution.

Returns List of util.SavableComponent

get_summaries ()

get_variables (*include_submodules=False, include_nontrainable=False*)

import_experience (*states, internals, actions, terminal, reward*)

Stores experiences.

initialize (*custom_getter*)

observe (*terminal, reward*)

Adds an observation (reward and is-terminal) to the model without updating its trainable variables.

Parameters

- **terminal** (*bool*) – Whether the episode has terminated.
- **reward** (*float*) – The observed reward value.

Returns The value of the model-internal episode counter.

optimizer_arguments (*states, internals, actions, terminal, reward, next_states, next_internals*)

reset ()

Resets the model to its initial state on episode start. This should also reset all preprocessor(s).

Returns Current episode, timestep counter and the shallow-copied list of internal state initialization Tensors.

Return type tuple

restore (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

restore_component (*component_name, save_path*)

Restores a component's parameters from a save location.

Parameters

- **component_name** – The component to restore.
- **save_path** – The save location.

save (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** – Optional checkpoint directory.
- **append_timestep** – Appends the current timestep to the checkpoint file if true.

Returns Checkpoint path where the model was saved.

save_component (*component_name, save_path*)

Saves a component of this model to the designated location.

Parameters

- **component_name** – The component to save.
- **save_path** – The location to save to.

Returns Checkpoint path where the component was saved.

setup ()

Sets up the TensorFlow model graph and initializes (and enters) the TensorFlow session.

tf_action_exploration (*action, exploration, action_spec*)

Applies optional exploration to the action (post-processor for action outputs).

Parameters

- **action** (*tf.Tensor*) – The original output action tensor (to be post-processed).
- **exploration** (*Exploration*) – The Exploration object to use.
- **action_spec** (*dict*) – Dict specifying the action space.

Returns The post-processed action output tensor.

tf_actions_and_internals (*states, internals, deterministic*)

tf_discounted_cumulative_reward (*terminal, reward, discount, final_reward=0.0*)

Creates the TensorFlow operations for calculating the discounted cumulative rewards for a given sequence of rewards.

Parameters

- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **discount** – Discount factor.
- **final_reward** – Last reward value in the sequence.

Returns Discounted cumulative reward tensor.

tf_import_experience (*states, internals, actions, terminal, reward*)

Imports experiences into the TensorFlow memory structure. Can be used to import off-policy data.

Parameters

- **states** – Dict of state values to import with keys as state names and values as values to set.
- **internals** – Internal values to set, can be fetched from agent via agent.current_internals if no values available.

- **actions** – Dict of action values to import with keys as action names and values as values to set.
- **terminal** – Terminal value(s)
- **reward** – Reward value(s)

tf_initialize()

tf_kl_divergence (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

tf_loss (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

Creates the TensorFlow operations for calculating the full loss of a batch.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor.

tf_loss_per_instance (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

tf_observe_timestep (*states, internals, actions, terminal, reward*)

tf_optimization (*states, internals, actions, terminal, reward, next_states=None, next_internals=None*)

tf_predict_target_q (*states, internals, terminal, actions, reward, update*)

tf_preprocess (*states, actions, reward*)

tf_reference (*states, internals, actions, terminal, reward, next_states, next_internals, update*)

Creates the TensorFlow operations for obtaining the reference tensor(s), in case of a comparative loss.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Reference tensor(s).

tf_regularization_losses (*states, internals, update*)

tf_target_actions_and_internals (*states, internals, deterministic=True*)

```
class tensorflow.models.PGLogProbModel(states, actions, scope, device, saver, summarizer,
                                     execution, batching_capacity, variable_noise,
                                     states_preprocessing, actions_exploration, re-
                                     ward_preprocessing, update_mode, memory,
                                     optimizer, discount, network, distributions, en-
                                     tropy_regularization, baseline_mode, baseline,
                                     baseline_optimizer, gae_lambda)
```

Bases: `tensorflow.models.pg_model.PGModel`

Policy gradient model based on computing log likelihoods, e.g. VPG.

COMPONENT_BASELINE = 'baseline'

COMPONENT_DISTRIBUTION = 'distribution'

COMPONENT_NETWORK = 'network'

```
__init__(states, actions, scope, device, saver, summarizer, execution, batching_capacity,
         variable_noise, states_preprocessing, actions_exploration, reward_preprocessing, up-
         date_mode, memory, optimizer, discount, network, distributions, entropy_regularization,
         baseline_mode, baseline, baseline_optimizer, gae_lambda)
```

act (*states, internals, deterministic=False, independent=False, fetch_tensors=None*)

Does a forward pass through the model to retrieve action (outputs) given inputs for state (and internal state, if applicable (e.g. RNNs))

Parameters

- **states** (*dict*) – Dict of state values (each key represents one state space component).
- **internals** (*dict*) – Dict of internal state values (each key represents one internal state component).
- **deterministic** (*bool*) – If True, will not apply exploration after actions are calculated.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).

Returns

- Actual action-outputs (batched if state input is a batch).

Return type

tuple

as_local_model ()

baseline_optimizer_arguments (*states, internals, reward*)

Returns the baseline optimizer arguments including the time, the list of variables to optimize, and various functions which the optimizer might require to perform an update step.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.

Returns Baseline optimizer arguments as dict.

close ()

create_act_operations (*states, internals, deterministic, independent*)

create_distributions ()

create_observe_operations (*terminal, reward*)

create_operations (*states, internals, actions, terminal, reward, deterministic, independent*)

get_component (*component_name*)

Looks up a component by its name.

Parameters **component_name** – The name of the component to look up.

Returns The component for the provided name or None if there is no such component.

get_components ()

get_feed_dict (*states=None, internals=None, actions=None, terminal=None, reward=None, deterministic=None, independent=None*)

get_savable_components ()

Returns the list of all of the components this model consists of that can be individually saved and restored. For instance the network or distribution.

Returns List of util.SavableComponent

get_summaries ()

get_variables (*include_submodules=False, include_nontrainable=False*)

import_experience (*states, internals, actions, terminal, reward*)

Stores experiences.

initialize (*custom_getter*)

observe (*terminal, reward*)

Adds an observation (reward and is-terminal) to the model without updating its trainable variables.

Parameters

- **terminal** (*bool*) – Whether the episode has terminated.
- **reward** (*float*) – The observed reward value.

Returns The value of the model-internal episode counter.

optimizer_arguments (*states, internals, actions, terminal, reward, next_states, next_internals*)

reset ()

Resets the model to its initial state on episode start. This should also reset all preprocessor(s).

Returns Current episode, timestep counter and the shallow-copied list of internal state initialization Tensors.

Return type tuple

restore (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

restore_component (*component_name*, *save_path*)

Restores a component's parameters from a save location.

Parameters

- **component_name** – The component to restore.
- **save_path** – The save location.

save (*directory=None*, *append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** – Optional checkpoint directory.
- **append_timestep** – Appends the current timestep to the checkpoint file if true.

Returns Checkpoint path where the model was saved.

save_component (*component_name*, *save_path*)

Saves a component of this model to the designated location.

Parameters

- **component_name** – The component to save.
- **save_path** – The location to save to.

Returns Checkpoint path where the component was saved.

setup ()

Sets up the TensorFlow model graph and initializes (and enters) the TensorFlow session.

tf_action_exploration (*action*, *exploration*, *action_spec*)

Applies optional exploration to the action (post-processor for action outputs).

Parameters

- **action** (*tf.Tensor*) – The original output action tensor (to be post-processed).
- **exploration** (*Exploration*) – The Exploration object to use.
- **action_spec** (*dict*) – Dict specifying the action space.

Returns The post-processed action output tensor.

tf_actions_and_internals (*states*, *internals*, *deterministic*)

tf_baseline_loss (*states*, *internals*, *reward*, *update*, *reference=None*)

Creates the TensorFlow operations for calculating the baseline loss of a batch.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **reward** – Reward tensor.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor.

tf_discounted_cumulative_reward (*terminal, reward, discount, final_reward=0.0*)

Creates the TensorFlow operations for calculating the discounted cumulative rewards for a given sequence of rewards.

Parameters

- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **discount** – Discount factor.
- **final_reward** – Last reward value in the sequence.

Returns Discounted cumulative reward tensor.

tf_import_experience (*states, internals, actions, terminal, reward*)

Imports experiences into the TensorFlow memory structure. Can be used to import off-policy data.

Parameters

- **states** – Dict of state values to import with keys as state names and values as values to set.
- **internals** – Internal values to set, can be fetched from agent via agent.current_internals if no values available.
- **actions** – Dict of action values to import with keys as action names and values as values to set.
- **terminal** – Terminal value(s)
- **reward** – Reward value(s)

tf_initialize ()

tf_kl_divergence (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

tf_loss (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

Creates the TensorFlow operations for calculating the full loss of a batch.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor.

tf_loss_per_instance (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

tf_observe_timestep (*states, internals, actions, terminal, reward*)

tf_optimization (*states, internals, actions, terminal, reward, next_states=None, next_internals=None*)

tf_preprocess (*states, actions, reward*)

tf_reference (*states, internals, actions, terminal, reward, next_states, next_internals, update*)

Creates the TensorFlow operations for obtaining the reference tensor(s), in case of a comparative loss.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Reference tensor(s).

tf_regularization_losses (*states, internals, update*)

tf_reward_estimation (*states, internals, terminal, reward, update*)

class `tensorforce.models.QModel` (*states, actions, scope, device, saver, summarizer, execution, batching_capacity, variable_noise, states_preprocessing, actions_exploration, reward_preprocessing, update_mode, memory, optimizer, discount, network, distributions, entropy_regularization, target_sync_frequency, target_update_weight, double_q_model, huber_loss*)

Bases: `tensorforce.models.distribution_model.DistributionModel`

Q-value model.

COMPONENT_DISTRIBUTION = 'distribution'

COMPONENT_NETWORK = 'network'

COMPONENT_TARGET_DISTRIBUTION = 'target_distribution'

COMPONENT_TARGET_NETWORK = 'target_network'

__init__ (*states, actions, scope, device, saver, summarizer, execution, batching_capacity, variable_noise, states_preprocessing, actions_exploration, reward_preprocessing, update_mode, memory, optimizer, discount, network, distributions, entropy_regularization, target_sync_frequency, target_update_weight, double_q_model, huber_loss*)

act (*states, internals, deterministic=False, independent=False, fetch_tensors=None*)

Does a forward pass through the model to retrieve action (outputs) given inputs for state (and internal state, if applicable (e.g. RNNs))

Parameters

- **states** (*dict*) – Dict of state values (each key represents one state space component).
- **internals** (*dict*) – Dict of internal state values (each key represents one internal state component).

- **deterministic** (*bool*) – If True, will not apply exploration after actions are calculated.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).

Returns

- Actual action-outputs (batched if state input is a batch).

Return type tuple

as_local_model ()

close ()

create_act_operations (*states, internals, deterministic, independent*)

create_distributions ()

create_observe_operations (*terminal, reward*)

create_operations (*states, internals, actions, terminal, reward, deterministic, independent*)

get_component (*component_name*)

Looks up a component by its name.

Parameters **component_name** – The name of the component to look up.

Returns The component for the provided name or None if there is no such component.

get_components ()

get_feed_dict (*states=None, internals=None, actions=None, terminal=None, reward=None, deterministic=None, independent=None*)

get_savable_components ()

Returns the list of all of the components this model consists of that can be individually saved and restored.
For instance the network or distribution.

Returns List of util.SavableComponent

get_summaries ()

get_variables (*include_submodules=False, include_nontrainable=False*)

import_experience (*states, internals, actions, terminal, reward*)

Stores experiences.

initialize (*custom_getter*)

observe (*terminal, reward*)

Adds an observation (reward and is-terminal) to the model without updating its trainable variables.

Parameters

- **terminal** (*bool*) – Whether the episode has terminated.
- **reward** (*float*) – The observed reward value.

Returns The value of the model-internal episode counter.

optimizer_arguments (*states, internals, actions, terminal, reward, next_states, next_internals*)

reset ()

Resets the model to its initial state on episode start. This should also reset all preprocessor(s).

Returns Current episode, timestep counter and the shallow-copied list of internal state initialization Tensors.

Return type tuple

restore (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

restore_component (*component_name, save_path*)

Restores a component's parameters from a save location.

Parameters

- **component_name** – The component to restore.
- **save_path** – The save location.

save (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** – Optional checkpoint directory.
- **append_timestep** – Appends the current timestep to the checkpoint file if true.

Returns Checkpoint path where the model was saved.

save_component (*component_name, save_path*)

Saves a component of this model to the designated location.

Parameters

- **component_name** – The component to save.
- **save_path** – The location to save to.

Returns Checkpoint path where the component was saved.

setup ()

Sets up the TensorFlow model graph and initializes (and enters) the TensorFlow session.

target_optimizer_arguments ()

Returns the target optimizer arguments including the time, the list of variables to optimize, and various functions which the optimizer might require to perform an update step.

Returns Target optimizer arguments as dict.

tf_action_exploration (*action, exploration, action_spec*)

Applies optional exploration to the action (post-processor for action outputs).

Parameters

- **action** (*tf.Tensor*) – The original output action tensor (to be post-processed).
- **exploration** (*Exploration*) – The Exploration object to use.
- **action_spec** (*dict*) – Dict specifying the action space.

Returns The post-processed action output tensor.

tf_actions_and_internals (*states, internals, deterministic*)

tf_discounted_cumulative_reward (*terminal, reward, discount, final_reward=0.0*)

Creates the TensorFlow operations for calculating the discounted cumulative rewards for a given sequence of rewards.

Parameters

- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **discount** – Discount factor.
- **final_reward** – Last reward value in the sequence.

Returns Discounted cumulative reward tensor.

tf_import_experience (*states, internals, actions, terminal, reward*)

Imports experiences into the TensorFlow memory structure. Can be used to import off-policy data.

Parameters

- **states** – Dict of state values to import with keys as state names and values as values to set.
- **internals** – Internal values to set, can be fetched from agent via agent.current_internals if no values available.
- **actions** – Dict of action values to import with keys as action names and values as values to set.
- **terminal** – Terminal value(s)
- **reward** – Reward value(s)

tf_initialize ()

tf_kl_divergence (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

tf_loss (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

Creates the TensorFlow operations for calculating the full loss of a batch.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor.

tf_loss_per_instance (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

tf_observe_timestep (*states, internals, actions, terminal, reward*)

tf_optimization (*states, internals, actions, terminal, reward, next_states=None, next_internals=None*)

tf_preprocess (*states, actions, reward*)

tf_q_delta (*q_value, next_q_value, terminal, reward*)

Creates the deltas (or advantage) of the Q values.

Returns A list of deltas per action

tf_q_value (*embedding, distr_params, action, name*)

tf_reference (*states, internals, actions, terminal, reward, next_states, next_internals, update*)

Creates the TensorFlow operations for obtaining the reference tensor(s), in case of a comparative loss.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Reference tensor(s).

tf_regularization_losses (*states, internals, update*)

```
class tensorflow.models.QNstepModel(states, actions, scope, device, saver, summarizer,
                                execution, batching_capacity, variable_noise,
                                states_preprocessing, actions_exploration, re-
                                ward_preprocessing, update_mode, memory,
                                optimizer, discount, network, distributions, en-
                                tropy_regularization, target_sync_frequency, tar-
                                get_update_weight, double_q_model, huber_loss)
```

Bases: [*tensorflow.models.q_model.QModel*](#)

Deep Q network using n-step rewards as described in Asynchronous Methods for Reinforcement Learning.

COMPONENT_DISTRIBUTION = 'distribution'

COMPONENT_NETWORK = 'network'

COMPONENT_TARGET_DISTRIBUTION = 'target_distribution'

COMPONENT_TARGET_NETWORK = 'target_network'

```
__init__(states, actions, scope, device, saver, summarizer, execution, batching_capacity,
        variable_noise, states_preprocessing, actions_exploration, reward_preprocessing, up-
        date_mode, memory, optimizer, discount, network, distributions, entropy_regularization, tar-
        get_sync_frequency, target_update_weight, double_q_model, huber_loss)
```

act (*states, internals, deterministic=False, independent=False, fetch_tensors=None*)

Does a forward pass through the model to retrieve action (outputs) given inputs for state (and internal state, if applicable (e.g. RNNs))

Parameters

- **states** (*dict*) – Dict of state values (each key represents one state space component).
- **internals** (*dict*) – Dict of internal state values (each key represents one internal state component).
- **deterministic** (*bool*) – If True, will not apply exploration after actions are calculated.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).

Returns

- Actual action-outputs (batched if state input is a batch).

Return type tuple

as_local_model ()

close ()

create_act_operations (*states, internals, deterministic, independent*)

create_distributions ()

create_observe_operations (*terminal, reward*)

create_operations (*states, internals, actions, terminal, reward, deterministic, independent*)

get_component (*component_name*)

Looks up a component by its name.

Parameters **component_name** – The name of the component to look up.

Returns The component for the provided name or None if there is no such component.

get_components ()

get_feed_dict (*states=None, internals=None, actions=None, terminal=None, reward=None, deterministic=None, independent=None*)

get_savable_components ()

Returns the list of all of the components this model consists of that can be individually saved and restored.
For instance the network or distribution.

Returns List of util.SavableComponent

get_summaries ()

get_variables (*include_submodules=False, include_nontrainable=False*)

import_experience (*states, internals, actions, terminal, reward*)

Stores experiences.

initialize (*custom_getter*)

observe (*terminal, reward*)

Adds an observation (reward and is-terminal) to the model without updating its trainable variables.

Parameters

- **terminal** (*bool*) – Whether the episode has terminated.
- **reward** (*float*) – The observed reward value.

Returns The value of the model-internal episode counter.

optimizer_arguments (*states, internals, actions, terminal, reward, next_states, next_internals*)

reset ()

Resets the model to its initial state on episode start. This should also reset all preprocessor(s).

Returns Current episode, timestep counter and the shallow-copied list of internal state initialization Tensors.

Return type tuple

restore (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

restore_component (*component_name, save_path*)

Restores a component's parameters from a save location.

Parameters

- **component_name** – The component to restore.
- **save_path** – The save location.

save (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** – Optional checkpoint directory.
- **append_timestep** – Appends the current timestep to the checkpoint file if true.

Returns Checkpoint path where the model was saved.

save_component (*component_name, save_path*)

Saves a component of this model to the designated location.

Parameters

- **component_name** – The component to save.
- **save_path** – The location to save to.

Returns Checkpoint path where the component was saved.

setup ()

Sets up the TensorFlow model graph and initializes (and enters) the TensorFlow session.

target_optimizer_arguments ()

Returns the target optimizer arguments including the time, the list of variables to optimize, and various functions which the optimizer might require to perform an update step.

Returns Target optimizer arguments as dict.

tf_action_exploration (*action, exploration, action_spec*)

Applies optional exploration to the action (post-processor for action outputs).

Parameters

- **action** (*tf.Tensor*) – The original output action tensor (to be post-processed).
- **exploration** (*Exploration*) – The Exploration object to use.
- **action_spec** (*dict*) – Dict specifying the action space.

Returns The post-processed action output tensor.

tf_actions_and_internals (*states, internals, deterministic*)

tf_discounted_cumulative_reward (*terminal, reward, discount, final_reward=0.0*)

Creates the TensorFlow operations for calculating the discounted cumulative rewards for a given sequence of rewards.

Parameters

- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **discount** – Discount factor.
- **final_reward** – Last reward value in the sequence.

Returns Discounted cumulative reward tensor.

tf_import_experience (*states, internals, actions, terminal, reward*)

Imports experiences into the TensorFlow memory structure. Can be used to import off-policy data.

Parameters

- **states** – Dict of state values to import with keys as state names and values as values to set.
- **internals** – Internal values to set, can be fetched from agent via agent.current_internals if no values available.
- **actions** – Dict of action values to import with keys as action names and values as values to set.
- **terminal** – Terminal value(s)
- **reward** – Reward value(s)

tf_initialize ()

tf_kl_divergence (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

tf_loss (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

Creates the TensorFlow operations for calculating the full loss of a batch.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.

- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor.

tf_loss_per_instance (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

tf_observe_timestep (*states, internals, actions, terminal, reward*)

tf_optimization (*states, internals, actions, terminal, reward, next_states=None, next_internals=None*)

tf_preprocess (*states, actions, reward*)

tf_q_delta (*q_value, next_q_value, terminal, reward*)

tf_q_value (*embedding, distr_params, action, name*)

tf_reference (*states, internals, actions, terminal, reward, next_states, next_internals, update*)

Creates the TensorFlow operations for obtaining the reference tensor(s), in case of a comparative loss.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Reference tensor(s).

tf_regularization_losses (*states, internals, update*)

```
class tensorflow.models.QNAFModel(states, actions, scope, device, saver, summarizer, execution, batching_capacity, variable_noise, states_preprocessing, actions_exploration, reward_preprocessing, update_mode, memory, optimizer, discount, network, distributions, entropy_regularization, target_sync_frequency, target_update_weight, double_q_model, huber_loss)
```

Bases: `tensorflow.models.q_model.QModel`

COMPONENT_DISTRIBUTION = 'distribution'

COMPONENT_NETWORK = 'network'

COMPONENT_TARGET_DISTRIBUTION = 'target_distribution'

COMPONENT_TARGET_NETWORK = 'target_network'

__init__ (*states, actions, scope, device, saver, summarizer, execution, batching_capacity, variable_noise, states_preprocessing, actions_exploration, reward_preprocessing, update_mode, memory, optimizer, discount, network, distributions, entropy_regularization, target_sync_frequency, target_update_weight, double_q_model, huber_loss*)

act (*states, internals, deterministic=False, independent=False, fetch_tensors=None*)

Does a forward pass through the model to retrieve action (outputs) given inputs for state (and internal state, if applicable (e.g. RNNs))

Parameters

- **states** (*dict*) – Dict of state values (each key represents one state space component).
- **internals** (*dict*) – Dict of internal state values (each key represents one internal state component).
- **deterministic** (*bool*) – If True, will not apply exploration after actions are calculated.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).

Returns

- Actual action-outputs (batched if state input is a batch).

Return type tuple

as_local_model ()

close ()

create_act_operations (*states, internals, deterministic, independent*)

create_distributions ()

create_observe_operations (*terminal, reward*)

create_operations (*states, internals, actions, terminal, reward, deterministic, independent*)

get_component (*component_name*)

Looks up a component by its name.

Parameters **component_name** – The name of the component to look up.

Returns The component for the provided name or None if there is no such component.

get_components ()

get_feed_dict (*states=None, internals=None, actions=None, terminal=None, reward=None, deterministic=None, independent=None*)

get_savable_components ()

Returns the list of all of the components this model consists of that can be individually saved and restored.
For instance the network or distribution.

Returns List of util.SavableComponent

get_summaries ()

get_variables (*include_submodules=False, include_nontrainable=False*)

import_experience (*states, internals, actions, terminal, reward*)

Stores experiences.

initialize (*custom_getter*)

observe (*terminal, reward*)

Adds an observation (reward and is-terminal) to the model without updating its trainable variables.

Parameters

- **terminal** (*bool*) – Whether the episode has terminated.
- **reward** (*float*) – The observed reward value.

Returns The value of the model-internal episode counter.

optimizer_arguments (*states, internals, actions, terminal, reward, next_states, next_internals*)

reset ()

Resets the model to its initial state on episode start. This should also reset all preprocessor(s).

Returns Current episode, timestep counter and the shallow-copied list of internal state initialization Tensors.

Return type tuple

restore (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

restore_component (*component_name, save_path*)

Restores a component's parameters from a save location.

Parameters

- **component_name** – The component to restore.
- **save_path** – The save location.

save (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** – Optional checkpoint directory.
- **append_timestep** – Appends the current timestep to the checkpoint file if true.

Returns Checkpoint path where the model was saved.

save_component (*component_name, save_path*)

Saves a component of this model to the designated location.

Parameters

- **component_name** – The component to save.
- **save_path** – The location to save to.

Returns Checkpoint path where the component was saved.

setup ()

Sets up the TensorFlow model graph and initializes (and enters) the TensorFlow session.

target_optimizer_arguments ()

Returns the target optimizer arguments including the time, the list of variables to optimize, and various functions which the optimizer might require to perform an update step.

Returns Target optimizer arguments as dict.

tf_action_exploration (*action, exploration, action_spec*)

Applies optional exploration to the action (post-processor for action outputs).

Parameters

- **action** (*tf.Tensor*) – The original output action tensor (to be post-processed).
- **exploration** (*Exploration*) – The Exploration object to use.
- **action_spec** (*dict*) – Dict specifying the action space.

Returns The post-processed action output tensor.

tf_actions_and_internals (*states, internals, deterministic*)

tf_discounted_cumulative_reward (*terminal, reward, discount, final_reward=0.0*)

Creates the TensorFlow operations for calculating the discounted cumulative rewards for a given sequence of rewards.

Parameters

- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **discount** – Discount factor.
- **final_reward** – Last reward value in the sequence.

Returns Discounted cumulative reward tensor.

tf_import_experience (*states, internals, actions, terminal, reward*)

Imports experiences into the TensorFlow memory structure. Can be used to import off-policy data.

Parameters

- **states** – Dict of state values to import with keys as state names and values as values to set.
- **internals** – Internal values to set, can be fetched from agent via agent.current_internals if no values available.
- **actions** – Dict of action values to import with keys as action names and values as values to set.
- **terminal** – Terminal value(s)
- **reward** – Reward value(s)

tf_initialize ()

tf_kl_divergence (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

tf_loss (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

Creates the TensorFlow operations for calculating the full loss of a batch.

Parameters

- **states** – Dict of state tensors.

- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor.

tf_loss_per_instance (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

tf_observe_timestep (*states, internals, actions, terminal, reward*)

tf_optimization (*states, internals, actions, terminal, reward, next_states=None, next_internals=None*)

tf_preprocess (*states, actions, reward*)

tf_q_delta (*q_value, next_q_value, terminal, reward*)
Creates the deltas (or advantage) of the Q values.

Returns A list of deltas per action

tf_q_value (*embedding, distr_params, action, name*)

tf_reference (*states, internals, actions, terminal, reward, next_states, next_internals, update*)
Creates the TensorFlow operations for obtaining the reference tensor(s), in case of a comparative loss.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Reference tensor(s).

tf_regularization_losses (*states, internals, update*)

```
class tensorflow.models.QDemoModel(states, actions, scope, device, saver, summarizer, execution, batching_capacity, variable_noise, states_preprocessing, actions_exploration, reward_preprocessing, update_mode, memory, optimizer, discount, network, distributions, entropy_regularization, target_sync_frequency, target_update_weight, double_q_model, huber_loss, expert_margin, supervised_weight, demo_memory_capacity, demo_batch_size)
```

Bases: `tensorflow.models.q_model.QModel`

Model for deep Q-learning from demonstration. Principal structure similar to double deep Q-networks but uses additional loss terms for demo data.

COMPONENT_DISTRIBUTION = 'distribution'

COMPONENT_NETWORK = 'network'

COMPONENT_TARGET_DISTRIBUTION = 'target_distribution'

COMPONENT_TARGET_NETWORK = 'target_network'

__init__(*states, actions, scope, device, saver, summarizer, execution, batching_capacity, variable_noise, states_preprocessing, actions_exploration, reward_preprocessing, update_mode, memory, optimizer, discount, network, distributions, entropy_regularization, target_sync_frequency, target_update_weight, double_q_model, huber_loss, expert_margin, supervised_weight, demo_memory_capacity, demo_batch_size*)

act(*states, internals, deterministic=False, independent=False, fetch_tensors=None*)

Does a forward pass through the model to retrieve action (outputs) given inputs for state (and internal state, if applicable (e.g. RNNs))

Parameters

- **states** (*dict*) – Dict of state values (each key represents one state space component).
- **internals** (*dict*) – Dict of internal state values (each key represents one internal state component).
- **deterministic** (*bool*) – If True, will not apply exploration after actions are calculated.
- **independent** (*bool*) – If true, action is not followed by observe (and hence not included in updates).

Returns

- Actual action-outputs (batched if state input is a batch).

Return type tuple

as_local_model()

close()

create_act_operations(*states, internals, deterministic, independent*)

create_distributions()

create_observe_operations(*terminal, reward*)

create_operations(*states, internals, actions, terminal, reward, deterministic, independent*)

demo_update()

Performs a demonstration update by calling the demo optimization operation. Note that the batch data does not have to be fetched from the demo memory as this is now part of the TensorFlow operation of the demo update.

get_component(*component_name*)

Looks up a component by its name.

Parameters **component_name** – The name of the component to look up.

Returns The component for the provided name or None if there is no such component.

get_components ()

get_feed_dict (*states=None, internals=None, actions=None, terminal=None, reward=None, deterministic=None, independent=None*)

get_savable_components ()

Returns the list of all of the components this model consists of that can be individually saved and restored. For instance the network or distribution.

Returns List of util.SavableComponent

get_summaries ()

get_variables (*include_submodules=False, include_nontrainable=False*)

Returns the TensorFlow variables used by the model.

Returns List of variables.

import_demo_experience (*states, internals, actions, terminal, reward*)

Stores demonstrations in the demo memory.

import_experience (*states, internals, actions, terminal, reward*)

Stores experiences.

initialize (*custom_getter*)

observe (*terminal, reward*)

Adds an observation (reward and is-terminal) to the model without updating its trainable variables.

Parameters

- **terminal** (*bool*) – Whether the episode has terminated.
- **reward** (*float*) – The observed reward value.

Returns The value of the model-internal episode counter.

optimizer_arguments (*states, internals, actions, terminal, reward, next_states, next_internals*)

reset ()

Resets the model to its initial state on episode start. This should also reset all preprocessor(s).

Returns Current episode, timestep counter and the shallow-copied list of internal state initialization Tensors.

Return type tuple

restore (*directory=None, file=None*)

Restore TensorFlow model. If no checkpoint file is given, the latest checkpoint is restored. If no checkpoint directory is given, the model's default saver directory is used (unless file specifies the entire path).

Parameters

- **directory** – Optional checkpoint directory.
- **file** – Optional checkpoint file, or path if directory not given.

restore_component (*component_name, save_path*)

Restores a component's parameters from a save location.

Parameters

- **component_name** – The component to restore.
- **save_path** – The save location.

save (*directory=None, append_timestep=True*)

Save TensorFlow model. If no checkpoint directory is given, the model's default saver directory is used. Optionally appends current timestep to prevent overwriting previous checkpoint files. Turn off to be able to load model from the same given path argument as given here.

Parameters

- **directory** – Optional checkpoint directory.
- **append_timestep** – Appends the current timestep to the checkpoint file if true.

Returns Checkpoint path where the model was saved.

save_component (*component_name, save_path*)

Saves a component of this model to the designated location.

Parameters

- **component_name** – The component to save.
- **save_path** – The location to save to.

Returns Checkpoint path where the component was saved.

setup ()

Sets up the TensorFlow model graph and initializes (and enters) the TensorFlow session.

target_optimizer_arguments ()

Returns the target optimizer arguments including the time, the list of variables to optimize, and various functions which the optimizer might require to perform an update step.

Returns Target optimizer arguments as dict.

tf_action_exploration (*action, exploration, action_spec*)

Applies optional exploration to the action (post-processor for action outputs).

Parameters

- **action** (*tf.Tensor*) – The original output action tensor (to be post-processed).
- **exploration** (*Exploration*) – The Exploration object to use.
- **action_spec** (*dict*) – Dict specifying the action space.

Returns The post-processed action output tensor.

tf_actions_and_internals (*states, internals, deterministic*)

tf_combined_loss (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

Combines Q-loss and demo loss.

tf_demo_loss (*states, actions, terminal, reward, internals, update, reference=None*)

Extends the q-model loss via the dqfd large-margin loss.

tf_demo_optimization (*states, internals, actions, terminal, reward, next_states, next_internals*)

tf_discounted_cumulative_reward (*terminal, reward, discount, final_reward=0.0*)

Creates the TensorFlow operations for calculating the discounted cumulative rewards for a given sequence of rewards.

Parameters

- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **discount** – Discount factor.

- **final_reward** – Last reward value in the sequence.

Returns Discounted cumulative reward tensor.

tf_import_demo_experience (*states, internals, actions, terminal, reward*)

Imports a single experience to memory.

tf_import_experience (*states, internals, actions, terminal, reward*)

Imports experiences into the TensorFlow memory structure. Can be used to import off-policy data.

Parameters

- **states** – Dict of state values to import with keys as state names and values as values to set.
- **internals** – Internal values to set, can be fetched from agent via agent.current_internals if no values available.
- **actions** – Dict of action values to import with keys as action names and values as values to set.
- **terminal** – Terminal value(s)
- **reward** – Reward value(s)

tf_initialize ()

tf_kl_divergence (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

tf_loss (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

Creates the TensorFlow operations for calculating the full loss of a batch.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.
- **reference** – Optional reference tensor(s), in case of a comparative loss.

Returns Loss tensor.

tf_loss_per_instance (*states, internals, actions, terminal, reward, next_states, next_internals, update, reference=None*)

tf_observe_timestep (*states, internals, actions, terminal, reward*)

tf_optimization (*states, internals, actions, terminal, reward, next_states=None, next_internals=None*)

tf_preprocess (*states, actions, reward*)

tf_q_delta (*q_value, next_q_value, terminal, reward*)

Creates the deltas (or advantage) of the Q values.

Returns A list of deltas per action

tf_q_value (*embedding, distr_params, action, name*)

tf_reference (*states, internals, actions, terminal, reward, next_states, next_internals, update*)

Creates the TensorFlow operations for obtaining the reference tensor(s), in case of a comparative loss.

Parameters

- **states** – Dict of state tensors.
- **internals** – List of prior internal state tensors.
- **actions** – Dict of action tensors.
- **terminal** – Terminal boolean tensor.
- **reward** – Reward tensor.
- **next_states** – Dict of successor state tensors.
- **next_internals** – List of posterior internal state tensors.
- **update** – Boolean tensor indicating whether this call happens during an update.

Returns Reference tensor(s).

tf_regularization_losses (*states, internals, update*)

tensorforce.tests package

Submodules

tensorforce.tests.base_agent_test module

class tensorforce.tests.base_agent_test.**BaseAgentTest**

Bases: *tensorforce.tests.base_test.BaseTest*

Base class for tests of fundamental Agent functionality, i.e. various action types and shapes and internal states.

__init__

x.init(...) initializes x; see help(type(x)) for signature

agent = None

base_test_pass (*name, environment, network, **kwargs*)

Basic test loop, requires an Agent to achieve a certain performance on an environment.

Parameters

- **name** (*str*) – The name of the test.
- **environment** (*Environment*) – The Environment object to use for the test.
- **network** (*LayerBasedNetwork*) – The Network to use for the agent’s model.
- **kwargs** (*any*) – Agent arguments.

base_test_run (*name, environment, network, **kwargs*)

Run test, tests whether algorithm can run and update without compilation errors, not whether it passes.

Parameters

- **name** (*str*) – The name of the test.
- **environment** (*Environment*) – The Environment object to use for the test.

- **network** (`LayerBasedNetwork`) – The Network to use for the agent’s model.
- **kwargs** (*any*) – Agent arguments.

```

config = None
exclude_bool = False
exclude_bounded = False
exclude_float = False
exclude_int = False
exclude_lstm = False
exclude_multi = False
multi_config = None
pass_threshold = 0.8
pre_run (agent, environment)
    Called before Runner.run.
requires_network = True
test_bool ()
    Tests the case of one boolean action.
test_bounded_float ()
    Tests the case of one bounded float action, i.e. with min and max value.
test_float ()
    Tests the case of one float action.
test_int ()
    Tests the case of one integer action.
test_lstm ()
    Tests the case of using internal states via an LSTM layer (for one integer action).
test_multi ()
    Tests the case of multiple actions of different type and shape.

```

tensorforce.tests.base_test module

```

class tensorforce.tests.base_test.BaseTest
    Bases: object

    Base class for tests of Agent functionality.

    __init__
        x.init(...) initializes x; see help(type(x)) for signature

    agent = None

    base_test_pass (name, environment, network, **kwargs)
        Basic test loop, requires an Agent to achieve a certain performance on an environment.

        Parameters
        • name (str) – The name of the test.
        • environment (Environment) – The Environment object to use for the test.

```

- **network** (`LayerBasedNetwork`) – The Network to use for the agent’s model.
- **kwargs** (*any*) – Agent arguments.

base_test_run (*name, environment, network, **kwargs*)

Run test, tests whether algorithm can run and update without compilation errors, not whether it passes.

Parameters

- **name** (*str*) – The name of the test.
- **environment** (`Environment`) – The Environment object to use for the test.
- **network** (`LayerBasedNetwork`) – The Network to use for the agent’s model.
- **kwargs** (*any*) – Agent arguments.

pass_threshold = 0.8

pre_run (*agent, environment*)

Called before `Runner.run`.

requires_network = True

tensorforce.tests.test_constant_agent module

class `tensorforce.tests.test_constant_agent.TestConstantAgent` (*methodName='runTest'*)

Bases: `tensorforce.tests.base_agent_test.BaseAgentTest`, `unittest.case.TestCase`

__init__ (*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a `ValueError` if the instance does not have a method with the specified name.

addCleanup (*function, *args, **kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after `tearDown` on test failure or success.

Cleanup items are called even if `setUp` fails (unlike `tearDown`).

addTypeEqualityFunc (*typeobj, function*)

Add a type specific `assertEqual` style function to compare a type.

This method is for use by `TestCase` subclasses that need to register their own type equality functions to provide nicer error messages.

Parameters

- **typeobj** – The data type to call this function on when both values are of the same type in `assertEqual()`.
- **function** – The callable taking two arguments and an optional `msg=` argument that raises `self.failureException` with a useful error message when the two arguments are not equal.

agent

alias of `ConstantAgent`

assertAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertAlmostEquals (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertDictContainsSubset (*expected, actual, msg=None*)

Checks whether actual is a superset of expected.

assertDictEqual (*d1, d2, msg=None*)

assertEqual (*first, second, msg=None*)

Fail if the two objects are unequal as determined by the '==' operator.

assertEquals (*first, second, msg=None*)

Fail if the two objects are unequal as determined by the '==' operator.

assertFalse (*expr, msg=None*)

Check that the expression is false.

assertGreater (*a, b, msg=None*)

Just like self.assertTrue(a > b), but with a nicer default message.

assertGreaterEqual (*a, b, msg=None*)

Just like self.assertTrue(a >= b), but with a nicer default message.

assertIn (*member, container, msg=None*)

Just like self.assertTrue(a in b), but with a nicer default message.

assertIs (*expr1, expr2, msg=None*)

Just like self.assertTrue(a is b), but with a nicer default message.

assertIsInstance (*obj, cls, msg=None*)

Same as self.assertTrue(isinstance(obj, cls)), with a nicer default message.

assertIsNone (*obj, msg=None*)

Same as self.assertTrue(obj is None), with a nicer default message.

assertIsNot (*expr1, expr2, msg=None*)

Just like self.assertTrue(a is not b), but with a nicer default message.

assertIsNotNone (*obj, msg=None*)

Included for symmetry with assertIsNone.

assertItemsEqual (*expected_seq, actual_seq, msg=None*)

An unordered sequence specific comparison. It asserts that actual_seq and expected_seq have the same element counts. Equivalent to:

Asserts that each element has the same count in both sequences. .. rubric:: Example

- [0, 1, 1] and [1, 0, 1] compare equal.
- [0, 0, 1] and [0, 1] compare unequal.

assertLess (*a, b, msg=None*)

Just like self.assertTrue(a < b), but with a nicer default message.

assertLessEqual (*a, b, msg=None*)

Just like self.assertTrue(a <= b), but with a nicer default message.

assertListEqual (*list1, list2, msg=None*)

A list-specific equality assertion.

Parameters

- **list1** – The first list to compare.
- **list2** – The second list to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assertMultiLineEqual (*first, second, msg=None*)

Assert that two multi-line strings are equal.

assertNotAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotAlmostEquals (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotEqual (*first, second, msg=None*)

Fail if the two objects are equal as determined by the '!=' operator.

assertNotEquals (*first, second, msg=None*)

Fail if the two objects are equal as determined by the '!=' operator.

assertNotIn (*member, container, msg=None*)

Just like self.assertTrue(a not in b), but with a nicer default message.

assertNotIsInstance (*obj, cls, msg=None*)

Included for symmetry with assertIsInstance.

assertNotRegexpMatches (*text, unexpected_regexp, msg=None*)

Fail the test if the text matches the regular expression.

assertRaises (*excClass, callableObj=None, *args, **kwargs*)

Fail unless an exception of class excClass is raised by callableObj when invoked with arguments args and keyword arguments kwargs. If a different type of exception is raised, it will not be caught, and the test case will be deemed to have suffered an error, exactly as for an unexpected exception.

If called with callableObj omitted or None, will return a context object used like this:

The context manager keeps a reference to the exception as the 'exception' attribute. This allows you to inspect the exception after the assertion:

assertRaisesRegexp (*expected_exception, expected_regexp, callable_obj=None, *args, **kwargs*)

Asserts that the message in a raised exception matches a regexp.

Parameters

- **expected_exception** – Exception class expected to be raised.
- **expected_regexp** – Regexp (re pattern object or string) expected to be found in error message.
- **callable_obj** – Function to be called.
- **args** – Extra args.
- **kwargs** – Extra kwargs.

assertRegexpMatches (*text, expected_regexp, msg=None*)

Fail the test unless the text matches the regular expression.

assertSequenceEqual (*seq1, seq2, msg=None, seq_type=None*)

An equality assertion for ordered sequences (like lists and tuples).

For the purposes of this function, a valid ordered sequence type is one which can be indexed, has a length, and has an equality operator.

Parameters

- **seq1** – The first sequence to compare.
- **seq2** – The second sequence to compare.
- **seq_type** – The expected datatype of the sequences, or None if no datatype should be enforced.
- **msg** – Optional message to use on failure instead of a list of differences.

assertSetEqual (*set1, set2, msg=None*)

A set-specific equality assertion.

Parameters

- **set1** – The first set to compare.
- **set2** – The second set to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assertSetEqual uses ducktyping to support different types of sets, and is optimized for sets specifically (parameters must support a difference method).

assertTrue (*expr, msg=None*)

Check that the expression is true.

assertTupleEqual (*tuple1, tuple2, msg=None*)

A tuple-specific equality assertion.

Parameters

- **tuple1** – The first tuple to compare.
- **tuple2** – The second tuple to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assert_ (*expr, msg=None*)

Check that the expression is true.

base_test_pass (*name, environment, network, **kwargs*)

Basic test loop, requires an Agent to achieve a certain performance on an environment.

Parameters

- **name** (*str*) – The name of the test.
- **environment** (*Environment*) – The Environment object to use for the test.
- **network** (*LayerBasedNetwork*) – The Network to use for the agent’s model.
- **kwargs** (*any*) – Agent arguments.

base_test_run (*name, environment, network, **kwargs*)

Run test, tests whether algorithm can run and update without compilation errors, not whether it passes.

Parameters

- **name** (*str*) – The name of the test.
- **environment** (*Environment*) – The Environment object to use for the test.
- **network** (*LayerBasedNetwork*) – The Network to use for the agent’s model.
- **kwargs** (*any*) – Agent arguments.

config = {'action_values': {'action': 1.0}}

countTestCases ()

debug ()

Run the test without collecting errors in a TestResult

defaultTestResult ()

doCleanups ()

Execute all cleanup functions. Normally called for you after tearDown.

exclude_bool = True

exclude_bounded = False

exclude_float = False

exclude_int = True

exclude_lstm = True

exclude_multi = True

fail (*msg=None*)

Fail immediately, with the given message.

failIf (**args, **kwargs*)

failIfAlmostEqual (**args, **kwargs*)

failIfEqual (**args, **kwargs*)

failUnless (**args, **kwargs*)

failUnlessAlmostEqual (**args, **kwargs*)

failUnlessEqual (**args, **kwargs*)

failUnlessRaises (**args, **kwargs*)

failureException

alias of AssertionError

id ()

longMessage = False

maxDiff = 640

```

multi_config = None
pass_threshold = 0.8
pre_run (agent, environment)
    Called before Runner.run.
requires_network = False
run (result=None)
setUp ()
    Hook method for setting up the test fixture before exercising it.
setUpClass ()
    Hook method for setting up class fixture before running tests in the class.
shortDescription ()
    Returns a one-line description of the test, or None if no description has been provided.

    The default implementation of this method returns the first line of the specified test method's docstring.
skipTest (reason)
    Skip this test.
tearDown ()
    Hook method for deconstructing the test fixture after testing it.
tearDownClass ()
    Hook method for deconstructing the class fixture after running all tests in the class.
test_bool ()
    Tests the case of one boolean action.
test_bounded_float ()
    Tests the case of one bounded float action, i.e. with min and max value.
test_float ()
    Tests the case of one float action.
test_int ()
    Tests the case of one integer action.
test_lstm ()
    Tests the case of using internal states via an LSTM layer (for one integer action).
test_multi ()
    Tests the case of multiple actions of different type and shape.

```

tensorforce.tests.test_ddqn_agent module

tensorforce.tests.test_dqfd_agent module

```

class tensorforce.tests.test_dqfd_agent.TestDQFDAgent (methodName='runTest')
    Bases: tensorforce.tests.base_agent_test.BaseAgentTest, unittest.case.TestCase
    __init__ (methodName='runTest')
        Create an instance of the class that will use the named test method when executed. Raises a ValueError if
        the instance does not have a method with the specified name.

```

addCleanup (*function, *args, **kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after `tearDown` on test failure or success.

Cleanup items are called even if `setUp` fails (unlike `tearDown`).

addTypeEqualityFunc (*typeobj, function*)

Add a type specific `assertEqual` style function to compare a type.

This method is for use by `TestCase` subclasses that need to register their own type equality functions to provide nicer error messages.

Parameters

- **typeobj** – The data type to call this function on when both values are of the same type in `assertEqual()`.
- **function** – The callable taking two arguments and an optional `msg=` argument that raises `self.failureException` with a useful error message when the two arguments are not equal.

agent

alias of `DQFDAgent`

assertAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertAlmostEquals (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertDictContainsSubset (*expected, actual, msg=None*)

Checks whether `actual` is a superset of `expected`.

assertDictEqual (*d1, d2, msg=None*)

assertEqual (*first, second, msg=None*)

Fail if the two objects are unequal as determined by the `'=='` operator.

assertEquals (*first, second, msg=None*)

Fail if the two objects are unequal as determined by the `'=='` operator.

assertFalse (*expr, msg=None*)

Check that the expression is false.

assertGreater (*a, b, msg=None*)

Just like `self.assertTrue(a > b)`, but with a nicer default message.

assertGreaterEqual (*a, b, msg=None*)

Just like `self.assertTrue(a >= b)`, but with a nicer default message.

assertIn (*member, container, msg=None*)

Just like self.assertTrue(a in b), but with a nicer default message.

assertIs (*expr1, expr2, msg=None*)

Just like self.assertTrue(a is b), but with a nicer default message.

assertIsInstance (*obj, cls, msg=None*)

Same as self.assertTrue(isinstance(obj, cls)), with a nicer default message.

assertIsNone (*obj, msg=None*)

Same as self.assertTrue(obj is None), with a nicer default message.

assertIsNot (*expr1, expr2, msg=None*)

Just like self.assertTrue(a is not b), but with a nicer default message.

assertIsNotNone (*obj, msg=None*)

Included for symmetry with assertIsNone.

assertItemsEqual (*expected_seq, actual_seq, msg=None*)

An unordered sequence specific comparison. It asserts that actual_seq and expected_seq have the same element counts. Equivalent to:

Asserts that each element has the same count in both sequences. .. rubric:: Example

- [0, 1, 1] and [1, 0, 1] compare equal.
- [0, 0, 1] and [0, 1] compare unequal.

assertLess (*a, b, msg=None*)

Just like self.assertTrue(a < b), but with a nicer default message.

assertLessEqual (*a, b, msg=None*)

Just like self.assertTrue(a <= b), but with a nicer default message.

assertListEqual (*list1, list2, msg=None*)

A list-specific equality assertion.

Parameters

- **list1** – The first list to compare.
- **list2** – The second list to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assertMultiLineEqual (*first, second, msg=None*)

Assert that two multi-line strings are equal.

assertNotAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotAlmostEquals (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotEqual (*first, second, msg=None*)

Fail if the two objects are equal as determined by the '!=' operator.

assertNotEquals (*first, second, msg=None*)

Fail if the two objects are equal as determined by the '!=' operator.

assertNotIn (*member, container, msg=None*)

Just like self.assertTrue(a not in b), but with a nicer default message.

assertNotIsInstance (*obj, cls, msg=None*)

Included for symmetry with assertIsInstance.

assertNotRegexpMatches (*text, unexpected_regexp, msg=None*)

Fail the test if the text matches the regular expression.

assertRaises (*excClass, callableObj=None, *args, **kwargs*)

Fail unless an exception of class excClass is raised by callableObj when invoked with arguments args and keyword arguments kwargs. If a different type of exception is raised, it will not be caught, and the test case will be deemed to have suffered an error, exactly as for an unexpected exception.

If called with callableObj omitted or None, will return a context object used like this:

The context manager keeps a reference to the exception as the 'exception' attribute. This allows you to inspect the exception after the assertion:

assertRaisesRegexp (*expected_exception, expected_regexp, callable_obj=None, *args, **kwargs*)

Asserts that the message in a raised exception matches a regexp.

Parameters

- **expected_exception** – Exception class expected to be raised.
- **expected_regexp** – Regexp (re pattern object or string) expected to be found in error message.
- **callable_obj** – Function to be called.
- **args** – Extra args.
- **kwargs** – Extra kwargs.

assertRegexpMatches (*text, expected_regexp, msg=None*)

Fail the test unless the text matches the regular expression.

assertSequenceEqual (*seq1, seq2, msg=None, seq_type=None*)

An equality assertion for ordered sequences (like lists and tuples).

For the purposes of this function, a valid ordered sequence type is one which can be indexed, has a length, and has an equality operator.

Parameters

- **seq1** – The first sequence to compare.
- **seq2** – The second sequence to compare.
- **seq_type** – The expected datatype of the sequences, or None if no datatype should be enforced.
- **msg** – Optional message to use on failure instead of a list of differences.

assertSetEqual (*set1, set2, msg=None*)

A set-specific equality assertion.

Parameters

- **set1** – The first set to compare.
- **set2** – The second set to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

`assertSetEqual` uses ducktyping to support different types of sets, and is optimized for sets specifically (parameters must support a difference method).

assertTrue (*expr*, *msg=None*)

Check that the expression is true.

assertTupleEqual (*tuple1*, *tuple2*, *msg=None*)

A tuple-specific equality assertion.

Parameters

- **tuple1** – The first tuple to compare.
- **tuple2** – The second tuple to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assert_ (*expr*, *msg=None*)

Check that the expression is true.

base_test_pass (*name*, *environment*, *network*, ***kwargs*)

Basic test loop, requires an Agent to achieve a certain performance on an environment.

Parameters

- **name** (*str*) – The name of the test.
- **environment** (*Environment*) – The Environment object to use for the test.
- **network** (*LayerBasedNetwork*) – The Network to use for the agent’s model.
- **kwargs** (*any*) – Agent arguments.

base_test_run (*name*, *environment*, *network*, ***kwargs*)

Run test, tests whether algorithm can run and update without compilation errors, not whether it passes.

Parameters

- **name** (*str*) – The name of the test.
- **environment** (*Environment*) – The Environment object to use for the test.
- **network** (*LayerBasedNetwork*) – The Network to use for the agent’s model.
- **kwargs** (*any*) – Agent arguments.

```
config = {'update_mode': {'frequency': 4, 'batch_size': 8, 'unit': 'timesteps'}, 'countTestCases ()
```

```
debug ()
```

```
debug ()
```

Run the test without collecting errors in a `TestResult`

```
defaultTestResult ()
```

```
doCleanups ()
```

Execute all cleanup functions. Normally called for you after `tearDown`.

```
exclude_bool = False
```

```
exclude_bounded = True
```

exclude_float = True

exclude_int = False

exclude_lstm = False

exclude_multi = False

fail (*msg=None*)
Fail immediately, with the given message.

failIf (**args, **kwargs*)

failIfAlmostEqual (**args, **kwargs*)

failIfEqual (**args, **kwargs*)

failUnless (**args, **kwargs*)

failUnlessAlmostEqual (**args, **kwargs*)

failUnlessEqual (**args, **kwargs*)

failUnlessRaises (**args, **kwargs*)

failureException
alias of AssertionError

id ()

longMessage = False

maxDiff = 640

multi_config = None

pass_threshold = 0.8

pre_run (*agent, environment*)

requires_network = True

run (*result=None*)

setUp ()
Hook method for setting up the test fixture before exercising it.

setUpClass ()
Hook method for setting up class fixture before running tests in the class.

shortDescription ()
Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

skipTest (*reason*)
Skip this test.

tearDown ()
Hook method for deconstructing the test fixture after testing it.

tearDownClass ()
Hook method for deconstructing the class fixture after running all tests in the class.

test_bool ()
Tests the case of one boolean action.

test_bounded_float()

Tests the case of one bounded float action, i.e. with min and max value.

test_float()

Tests the case of one float action.

test_int()

Tests the case of one integer action.

test_lstm()

Tests the case of using internal states via an LSTM layer (for one integer action).

test_multi()

Tests the case of multiple actions of different type and shape.

tensorforce.tests.test_dqn_agent module

class tensorforce.tests.test_dqn_agent.**TestDQNAgent** (*methodName='runTest'*)

Bases: *tensorforce.tests.base_agent_test.BaseAgentTest*, *unittest.case.TestCase*

__init__ (*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

addCleanup (*function, *args, **kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after `tearDown` on test failure or success.

Cleanup items are called even if `setUp` fails (unlike `tearDown`).

addTypeEqualityFunc (*typeobj, function*)

Add a type specific `assertEqual` style function to compare a type.

This method is for use by `TestCase` subclasses that need to register their own type equality functions to provide nicer error messages.

Parameters

- **typeobj** – The data type to call this function on when both values are of the same type in `assertEqual()`.
- **function** – The callable taking two arguments and an optional `msg=` argument that raises `self.failureException` with a useful error message when the two arguments are not equal.

agent

alias of `DQNAgent`

assertAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertAlmostEquals (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal

places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertDictContainsSubset (*expected, actual, msg=None*)

Checks whether actual is a superset of expected.

assertDictEqual (*d1, d2, msg=None*)

assertEqual (*first, second, msg=None*)

Fail if the two objects are unequal as determined by the '==' operator.

assertEquals (*first, second, msg=None*)

Fail if the two objects are unequal as determined by the '==' operator.

assertFalse (*expr, msg=None*)

Check that the expression is false.

assertGreater (*a, b, msg=None*)

Just like self.assertTrue(a > b), but with a nicer default message.

assertGreaterEqual (*a, b, msg=None*)

Just like self.assertTrue(a >= b), but with a nicer default message.

assertIn (*member, container, msg=None*)

Just like self.assertTrue(a in b), but with a nicer default message.

assertIs (*expr1, expr2, msg=None*)

Just like self.assertTrue(a is b), but with a nicer default message.

assertIsInstance (*obj, cls, msg=None*)

Same as self.assertTrue(isinstance(obj, cls)), with a nicer default message.

assertIsNone (*obj, msg=None*)

Same as self.assertTrue(obj is None), with a nicer default message.

assertIsNot (*expr1, expr2, msg=None*)

Just like self.assertTrue(a is not b), but with a nicer default message.

assertIsNotNone (*obj, msg=None*)

Included for symmetry with assertIsNone.

assertItemsEqual (*expected_seq, actual_seq, msg=None*)

An unordered sequence specific comparison. It asserts that actual_seq and expected_seq have the same element counts. Equivalent to:

Asserts that each element has the same count in both sequences. .. rubric:: Example

- [0, 1, 1] and [1, 0, 1] compare equal.
- [0, 0, 1] and [0, 1] compare unequal.

assertLess (*a, b, msg=None*)

Just like self.assertTrue(a < b), but with a nicer default message.

assertLessEqual (*a, b, msg=None*)

Just like self.assertTrue(a <= b), but with a nicer default message.

assertListEqual (*list1, list2, msg=None*)

A list-specific equality assertion.

Parameters

- **list1** – The first list to compare.
- **list2** – The second list to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assertMultiLineEqual (*first, second, msg=None*)

Assert that two multi-line strings are equal.

assertNotAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotAlmostEquals (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotEqual (*first, second, msg=None*)

Fail if the two objects are equal as determined by the '!=' operator.

assertNotEquals (*first, second, msg=None*)

Fail if the two objects are equal as determined by the '!=' operator.

assertNotIn (*member, container, msg=None*)

Just like self.assertTrue(a not in b), but with a nicer default message.

assertNotIsInstance (*obj, cls, msg=None*)

Included for symmetry with assertIsInstance.

assertNotRegexpMatches (*text, unexpected_regexp, msg=None*)

Fail the test if the text matches the regular expression.

assertRaises (*excClass, callableObj=None, *args, **kwargs*)

Fail unless an exception of class excClass is raised by callableObj when invoked with arguments args and keyword arguments kwargs. If a different type of exception is raised, it will not be caught, and the test case will be deemed to have suffered an error, exactly as for an unexpected exception.

If called with callableObj omitted or None, will return a context object used like this:

The context manager keeps a reference to the exception as the 'exception' attribute. This allows you to inspect the exception after the assertion:

assertRaisesRegexp (*expected_exception, expected_regexp, callable_obj=None, *args, **kwargs*)

Asserts that the message in a raised exception matches a regexp.

Parameters

- **expected_exception** – Exception class expected to be raised.
- **expected_regexp** – Regexp (re pattern object or string) expected to be found in error message.

- **callable_obj** – Function to be called.
- **args** – Extra args.
- **kwargs** – Extra kwargs.

assertRegexpMatches (*text, expected_regexp, msg=None*)

Fail the test unless the text matches the regular expression.

assertSequenceEqual (*seq1, seq2, msg=None, seq_type=None*)

An equality assertion for ordered sequences (like lists and tuples).

For the purposes of this function, a valid ordered sequence type is one which can be indexed, has a length, and has an equality operator.

Parameters

- **seq1** – The first sequence to compare.
- **seq2** – The second sequence to compare.
- **seq_type** – The expected datatype of the sequences, or None if no datatype should be enforced.
- **msg** – Optional message to use on failure instead of a list of differences.

assertSetEqual (*set1, set2, msg=None*)

A set-specific equality assertion.

Parameters

- **set1** – The first set to compare.
- **set2** – The second set to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

`assertSetEqual` uses ducktyping to support different types of sets, and is optimized for sets specifically (parameters must support a difference method).

assertTrue (*expr, msg=None*)

Check that the expression is true.

assertTupleEqual (*tuple1, tuple2, msg=None*)

A tuple-specific equality assertion.

Parameters

- **tuple1** – The first tuple to compare.
- **tuple2** – The second tuple to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assert_ (*expr, msg=None*)

Check that the expression is true.

base_test_pass (*name, environment, network, **kwargs*)

Basic test loop, requires an Agent to achieve a certain performance on an environment.

Parameters

- **name** (*str*) – The name of the test.
- **environment** (*Environment*) – The Environment object to use for the test.
- **network** (*LayerBasedNetwork*) – The Network to use for the agent’s model.

- **kwargs** (*any*) – Agent arguments.

base_test_run (*name, environment, network, **kwargs*)

Run test, tests whether algorithm can run and update without compilation errors, not whether it passes.

Parameters

- **name** (*str*) – The name of the test.
- **environment** (*Environment*) – The Environment object to use for the test.
- **network** (*LayerBasedNetwork*) – The Network to use for the agent’s model.
- **kwargs** (*any*) – Agent arguments.

config = {'update_mode': {'frequency': 8, 'batch_size': 8, 'unit': 'timesteps'}, 'countTestCases'()

debug ()

Run the test without collecting errors in a TestResult

defaultTestResult ()

doCleanups ()

Execute all cleanup functions. Normally called for you after tearDown.

exclude_bool = False

exclude_bounded = True

exclude_float = True

exclude_int = False

exclude_lstm = False

exclude_multi = False

fail (*msg=None*)

Fail immediately, with the given message.

failIf (**args, **kwargs*)

failIfAlmostEqual (**args, **kwargs*)

failIfEqual (**args, **kwargs*)

failUnless (**args, **kwargs*)

failUnlessAlmostEqual (**args, **kwargs*)

failUnlessEqual (**args, **kwargs*)

failUnlessRaises (**args, **kwargs*)

failureException

alias of AssertionError

id ()

longMessage = False

maxDiff = 640

multi_config = None

pass_threshold = 0.8

pre_run (*agent, environment*)
Called before `Runner.run`.

requires_network = **True**

run (*result=None*)

setUp ()
Hook method for setting up the test fixture before exercising it.

setUpClass ()
Hook method for setting up class fixture before running tests in the class.

shortDescription ()
Returns a one-line description of the test, or `None` if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

skipTest (*reason*)
Skip this test.

tearDown ()
Hook method for deconstructing the test fixture after testing it.

tearDownClass ()
Hook method for deconstructing the class fixture after running all tests in the class.

test_bool ()
Tests the case of one boolean action.

test_bounded_float ()
Tests the case of one bounded float action, i.e. with min and max value.

test_float ()
Tests the case of one float action.

test_int ()
Tests the case of one integer action.

test_lstm ()
Tests the case of using internal states via an LSTM layer (for one integer action).

test_multi ()
Tests the case of multiple actions of different type and shape.

tensorforce.tests.test_dqn_memories module

tensorforce.tests.test_dqn_nstep_agent module

class `tensorforce.tests.test_dqn_nstep_agent.TestDQNNstepAgent` (*methodName='runTest'*)
Bases: `tensorforce.tests.base_agent_test.BaseAgentTest`, `unittest.case.TestCase`

__init__ (*methodName='runTest'*)
Create an instance of the class that will use the named test method when executed. Raises a `ValueError` if the instance does not have a method with the specified name.

addCleanup (*function, *args, **kwargs*)
Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after `tearDown` on test failure or success.

Cleanup items are called even if `setUp` fails (unlike `tearDown`).

addTypeEqualityFunc (*typeobj, function*)

Add a type specific assertEquals style function to compare a type.

This method is for use by TestCase subclasses that need to register their own type equality functions to provide nicer error messages.

Parameters

- **typeobj** – The data type to call this function on when both values are of the same type in assertEquals().
- **function** – The callable taking two arguments and an optional msg= argument that raises self.failureException with a useful error message when the two arguments are not equal.

agent

alias of DQNNstepAgent

assertAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertAlmostEquals (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertDictContainsSubset (*expected, actual, msg=None*)

Checks whether actual is a superset of expected.

assertDictEqual (*d1, d2, msg=None*)**assertEqual** (*first, second, msg=None*)

Fail if the two objects are unequal as determined by the '==' operator.

assertEquals (*first, second, msg=None*)

Fail if the two objects are unequal as determined by the '==' operator.

assertFalse (*expr, msg=None*)

Check that the expression is false.

assertGreater (*a, b, msg=None*)

Just like self.assertTrue(a > b), but with a nicer default message.

assertGreaterEqual (*a, b, msg=None*)

Just like self.assertTrue(a >= b), but with a nicer default message.

assertIn (*member, container, msg=None*)

Just like self.assertTrue(a in b), but with a nicer default message.

assertIs (*expr1, expr2, msg=None*)

Just like self.assertTrue(a is b), but with a nicer default message.

assertIsInstance (*obj, cls, msg=None*)

Same as `self.assertTrue(isinstance(obj, cls))`, with a nicer default message.

assertIsNone (*obj, msg=None*)

Same as `self.assertTrue(obj is None)`, with a nicer default message.

assertIsNot (*expr1, expr2, msg=None*)

Just like `self.assertTrue(a is not b)`, but with a nicer default message.

assertIsNotNone (*obj, msg=None*)

Included for symmetry with `assertIsNone`.

assertItemsEqual (*expected_seq, actual_seq, msg=None*)

An unordered sequence specific comparison. It asserts that `actual_seq` and `expected_seq` have the same element counts. Equivalent to:

Asserts that each element has the same count in both sequences. .. rubric:: Example

- `[0, 1, 1]` and `[1, 0, 1]` compare equal.
- `[0, 0, 1]` and `[0, 1]` compare unequal.

assertLess (*a, b, msg=None*)

Just like `self.assertTrue(a < b)`, but with a nicer default message.

assertLessEqual (*a, b, msg=None*)

Just like `self.assertTrue(a <= b)`, but with a nicer default message.

assertListEqual (*list1, list2, msg=None*)

A list-specific equality assertion.

Parameters

- **list1** – The first list to compare.
- **list2** – The second list to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assertMultiLineEqual (*first, second, msg=None*)

Assert that two multi-line strings are equal.

assertNotAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotAlmostEquals (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotEqual (*first, second, msg=None*)

Fail if the two objects are equal as determined by the `'!='` operator.

assertNotEquals (*first, second, msg=None*)

Fail if the two objects are equal as determined by the ‘!=’ operator.

assertNotIn (*member, container, msg=None*)

Just like self.assertTrue(a not in b), but with a nicer default message.

assertNotIsInstance (*obj, cls, msg=None*)

Included for symmetry with assertIsInstance.

assertNotRegexpMatches (*text, unexpected_regexp, msg=None*)

Fail the test if the text matches the regular expression.

assertRaises (*excClass, callableObj=None, *args, **kwargs*)

Fail unless an exception of class excClass is raised by callableObj when invoked with arguments args and keyword arguments kwargs. If a different type of exception is raised, it will not be caught, and the test case will be deemed to have suffered an error, exactly as for an unexpected exception.

If called with callableObj omitted or None, will return a context object used like this:

The context manager keeps a reference to the exception as the ‘exception’ attribute. This allows you to inspect the exception after the assertion:

assertRaisesRegexp (*expected_exception, expected_regexp, callable_obj=None, *args, **kwargs*)

Asserts that the message in a raised exception matches a regexp.

Parameters

- **expected_exception** – Exception class expected to be raised.
- **expected_regexp** – Regexp (re pattern object or string) expected to be found in error message.
- **callable_obj** – Function to be called.
- **args** – Extra args.
- **kwargs** – Extra kwargs.

assertRegexpMatches (*text, expected_regexp, msg=None*)

Fail the test unless the text matches the regular expression.

assertSequenceEqual (*seq1, seq2, msg=None, seq_type=None*)

An equality assertion for ordered sequences (like lists and tuples).

For the purposes of this function, a valid ordered sequence type is one which can be indexed, has a length, and has an equality operator.

Parameters

- **seq1** – The first sequence to compare.
- **seq2** – The second sequence to compare.
- **seq_type** – The expected datatype of the sequences, or None if no datatype should be enforced.
- **msg** – Optional message to use on failure instead of a list of differences.

assertSetEqual (*set1, set2, msg=None*)

A set-specific equality assertion.

Parameters

- **set1** – The first set to compare.
- **set2** – The second set to compare.

- **msg** – Optional message to use on failure instead of a list of differences.

`assertSetEqual` uses ducktyping to support different types of sets, and is optimized for sets specifically (parameters must support a difference method).

assertTrue (*expr*, *msg=None*)

Check that the expression is true.

assertTupleEqual (*tuple1*, *tuple2*, *msg=None*)

A tuple-specific equality assertion.

Parameters

- **tuple1** – The first tuple to compare.
- **tuple2** – The second tuple to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assert_ (*expr*, *msg=None*)

Check that the expression is true.

base_test_pass (*name*, *environment*, *network*, ***kwargs*)

Basic test loop, requires an Agent to achieve a certain performance on an environment.

Parameters

- **name** (*str*) – The name of the test.
- **environment** (`Environment`) – The Environment object to use for the test.
- **network** (`LayerBasedNetwork`) – The Network to use for the agent’s model.
- **kwargs** (*any*) – Agent arguments.

base_test_run (*name*, *environment*, *network*, ***kwargs*)

Run test, tests whether algorithm can run and update without compilation errors, not whether it passes.

Parameters

- **name** (*str*) – The name of the test.
- **environment** (`Environment`) – The Environment object to use for the test.
- **network** (`LayerBasedNetwork`) – The Network to use for the agent’s model.
- **kwargs** (*any*) – Agent arguments.

```
config = {'update_mode': {'frequency': 4, 'batch_size': 4, 'unit': 'episodes'}, 'o
```

```
countTestCases ()
```

```
debug ()
```

Run the test without collecting errors in a `TestResult`

```
defaultTestResult ()
```

```
doCleanups ()
```

Execute all cleanup functions. Normally called for you after `tearDown`.

```
exclude_bool = False
```

```
exclude_bounded = True
```

```
exclude_float = True
```

```
exclude_int = False
```

```
exclude_lstm = False
```

```

exclude_multi = True

fail (msg=None)
    Fail immediately, with the given message.

failIf (*args, **kwargs)

failIfAlmostEqual (*args, **kwargs)

failIfEqual (*args, **kwargs)

failUnless (*args, **kwargs)

failUnlessAlmostEqual (*args, **kwargs)

failUnlessEqual (*args, **kwargs)

failUnlessRaises (*args, **kwargs)

failureException
    alias of AssertionError

id()

longMessage = False

maxDiff = 640

multi_config = None

pass_threshold = 0.8

pre_run (agent, environment)
    Called before Runner.run.

requires_network = True

run (result=None)

setUp()
    Hook method for setting up the test fixture before exercising it.

setUpClass()
    Hook method for setting up class fixture before running tests in the class.

shortDescription()
    Returns a one-line description of the test, or None if no description has been provided.

    The default implementation of this method returns the first line of the specified test method's docstring.

skipTest (reason)
    Skip this test.

tearDown()
    Hook method for deconstructing the test fixture after testing it.

tearDownClass()
    Hook method for deconstructing the class fixture after running all tests in the class.

test_bool()
    Tests the case of one boolean action.

test_bounded_float()
    Tests the case of one bounded float action, i.e. with min and max value.

test_float()
    Tests the case of one float action.

```

test_int()

Tests the case of one integer action.

test_lstm()

Tests the case of using internal states via an LSTM layer (for one integer action).

test_multi()

Tests the case of multiple actions of different type and shape.

tensorforce.tests.test_naf_agent module

class tensorforce.tests.test_naf_agent.**TestNAFAgent** (*methodName='runTest'*)

Bases: *tensorforce.tests.base_agent_test.BaseAgentTest*, *unittest.case.TestCase*

__init__ (*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a `ValueError` if the instance does not have a method with the specified name.

addCleanup (*function, *args, **kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after `tearDown` on test failure or success.

Cleanup items are called even if `setUp` fails (unlike `tearDown`).

addTypeEqualityFunc (*typeobj, function*)

Add a type specific `assertEqual` style function to compare a type.

This method is for use by `TestCase` subclasses that need to register their own type equality functions to provide nicer error messages.

Parameters

- **typeobj** – The data type to call this function on when both values are of the same type in `assertEqual()`.
- **function** – The callable taking two arguments and an optional `msg=` argument that raises `self.failureException` with a useful error message when the two arguments are not equal.

agent

alias of `NAFAgent`

assertAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertAlmostEquals (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertDictContainsSubset (*expected, actual, msg=None*)

Checks whether actual is a superset of expected.

assertDictEqual (*d1, d2, msg=None*)

assertEqual (*first, second, msg=None*)

Fail if the two objects are unequal as determined by the '==' operator.

assertEquals (*first, second, msg=None*)

Fail if the two objects are unequal as determined by the '==' operator.

assertFalse (*expr, msg=None*)

Check that the expression is false.

assertGreater (*a, b, msg=None*)

Just like self.assertTrue(a > b), but with a nicer default message.

assertGreaterEqual (*a, b, msg=None*)

Just like self.assertTrue(a >= b), but with a nicer default message.

assertIn (*member, container, msg=None*)

Just like self.assertTrue(a in b), but with a nicer default message.

assertIs (*expr1, expr2, msg=None*)

Just like self.assertTrue(a is b), but with a nicer default message.

assertIsInstance (*obj, cls, msg=None*)

Same as self.assertTrue(isinstance(obj, cls)), with a nicer default message.

assertIsNone (*obj, msg=None*)

Same as self.assertTrue(obj is None), with a nicer default message.

assertIsNot (*expr1, expr2, msg=None*)

Just like self.assertTrue(a is not b), but with a nicer default message.

assertIsNotNone (*obj, msg=None*)

Included for symmetry with assertIsNone.

assertItemsEqual (*expected_seq, actual_seq, msg=None*)

An unordered sequence specific comparison. It asserts that actual_seq and expected_seq have the same element counts. Equivalent to:

Asserts that each element has the same count in both sequences. .. rubric:: Example

- [0, 1, 1] and [1, 0, 1] compare equal.
- [0, 0, 1] and [0, 1] compare unequal.

assertLess (*a, b, msg=None*)

Just like self.assertTrue(a < b), but with a nicer default message.

assertLessEqual (*a, b, msg=None*)

Just like self.assertTrue(a <= b), but with a nicer default message.

assertListEqual (*list1, list2, msg=None*)

A list-specific equality assertion.

Parameters

- **list1** – The first list to compare.
- **list2** – The second list to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assertMultiLineEqual (*first, second, msg=None*)

Assert that two multi-line strings are equal.

assertNotAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotAlmostEquals (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotEqual (*first, second, msg=None*)

Fail if the two objects are equal as determined by the '!=' operator.

assertNotEquals (*first, second, msg=None*)

Fail if the two objects are equal as determined by the '!=' operator.

assertNotIn (*member, container, msg=None*)

Just like self.assertTrue(a not in b), but with a nicer default message.

assertNotIsInstance (*obj, cls, msg=None*)

Included for symmetry with assertIsInstance.

assertNotRegexpMatches (*text, unexpected_regexp, msg=None*)

Fail the test if the text matches the regular expression.

assertRaises (*excClass, callableObj=None, *args, **kwargs*)

Fail unless an exception of class excClass is raised by callableObj when invoked with arguments args and keyword arguments kwargs. If a different type of exception is raised, it will not be caught, and the test case will be deemed to have suffered an error, exactly as for an unexpected exception.

If called with callableObj omitted or None, will return a context object used like this:

The context manager keeps a reference to the exception as the 'exception' attribute. This allows you to inspect the exception after the assertion:

assertRaisesRegexp (*expected_exception, expected_regexp, callable_obj=None, *args, **kwargs*)

Asserts that the message in a raised exception matches a regexp.

Parameters

- **expected_exception** – Exception class expected to be raised.
- **expected_regexp** – Regexp (re pattern object or string) expected to be found in error message.
- **callable_obj** – Function to be called.
- **args** – Extra args.
- **kwargs** – Extra kwargs.

assertRegexpMatches (*text, expected_regexp, msg=None*)

Fail the test unless the text matches the regular expression.

assertSequenceEqual (*seq1, seq2, msg=None, seq_type=None*)

An equality assertion for ordered sequences (like lists and tuples).

For the purposes of this function, a valid ordered sequence type is one which can be indexed, has a length, and has an equality operator.

Parameters

- **seq1** – The first sequence to compare.
- **seq2** – The second sequence to compare.
- **seq_type** – The expected datatype of the sequences, or None if no datatype should be enforced.
- **msg** – Optional message to use on failure instead of a list of differences.

assertSetEqual (*set1, set2, msg=None*)

A set-specific equality assertion.

Parameters

- **set1** – The first set to compare.
- **set2** – The second set to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

`assertSetEqual` uses ducktyping to support different types of sets, and is optimized for sets specifically (parameters must support a difference method).

assertTrue (*expr, msg=None*)

Check that the expression is true.

assertTupleEqual (*tuple1, tuple2, msg=None*)

A tuple-specific equality assertion.

Parameters

- **tuple1** – The first tuple to compare.
- **tuple2** – The second tuple to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assert_ (*expr, msg=None*)

Check that the expression is true.

base_test_pass (*name, environment, network, **kwargs*)

Basic test loop, requires an Agent to achieve a certain performance on an environment.

Parameters

- **name** (*str*) – The name of the test.
- **environment** (`Environment`) – The Environment object to use for the test.
- **network** (`LayerBasedNetwork`) – The Network to use for the agent’s model.
- **kwargs** (*any*) – Agent arguments.

base_test_run (*name, environment, network, **kwargs*)

Run test, tests whether algorithm can run and update without compilation errors, not whether it passes.

Parameters

- **name** (*str*) – The name of the test.
- **environment** (*Environment*) – The Environment object to use for the test.
- **network** (*LayerBasedNetwork*) – The Network to use for the agent’s model.
- **kwargs** (*any*) – Agent arguments.

```
config = {'update_mode': {'frequency': 4, 'batch_size': 8, 'unit': 'timesteps'}, '
countTestCases ()
debug ()
    Run the test without collecting errors in a TestResult
defaultTestResult ()
doCleanups ()
    Execute all cleanup functions. Normally called for you after tearDown.
exclude_bool = True
exclude_bounded = True
exclude_float = False
exclude_int = True
exclude_lstm = True
exclude_multi = True
fail (msg=None)
    Fail immediately, with the given message.
failIf (*args, **kwargs)
failIfAlmostEqual (*args, **kwargs)
failIfEqual (*args, **kwargs)
failUnless (*args, **kwargs)
failUnlessAlmostEqual (*args, **kwargs)
failUnlessEqual (*args, **kwargs)
failUnlessRaises (*args, **kwargs)
failureException
    alias of AssertionError
id ()
longMessage = False
maxDiff = 640
multi_config = None
pass_threshold = 0.8
pre_run (agent, environment)
    Called before Runner.run.
requires_network = True
run (result=None)
```

setUp()
Hook method for setting up the test fixture before exercising it.

setUpClass()
Hook method for setting up class fixture before running tests in the class.

shortDescription()
Returns a one-line description of the test, or None if no description has been provided.
The default implementation of this method returns the first line of the specified test method's docstring.

skipTest(reason)
Skip this test.

tearDown()
Hook method for deconstructing the test fixture after testing it.

tearDownClass()
Hook method for deconstructing the class fixture after running all tests in the class.

test_bool()
Tests the case of one boolean action.

test_bounded_float()
Tests the case of one bounded float action, i.e. with min and max value.

test_float()
Tests the case of one float action.

test_int()
Tests the case of one integer action.

test_lstm()
Tests the case of using internal states via an LSTM layer (for one integer action).

test_multi()
Tests the case of multiple actions of different type and shape.

tensorforce.tests.test_ppo_agent module

class tensorforce.tests.test_ppo_agent.**TestPPOAgent** (*methodName='runTest'*)
Bases: *tensorforce.tests.base_agent_test.BaseAgentTest*, *unittest.case.TestCase*

__init__ (*methodName='runTest'*)
Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

addCleanup (*function, *args, **kwargs*)
Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after `tearDown` on test failure or success.
Cleanup items are called even if `setUp` fails (unlike `tearDown`).

addTypeEqualityFunc (*typeobj, function*)
Add a type specific `assertEqual` style function to compare a type.
This method is for use by `TestCase` subclasses that need to register their own type equality functions to provide nicer error messages.

Parameters

- **typeobj** – The data type to call this function on when both values are of the same type in `assertEqual()`.
- **function** – The callable taking two arguments and an optional `msg=` argument that raises `self.failureException` with a useful error message when the two arguments are not equal.

agent

alias of `PPOAgent`

assertAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertAlmostEquals (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertDictContainsSubset (*expected, actual, msg=None*)

Checks whether `actual` is a superset of `expected`.

assertDictEqual (*d1, d2, msg=None*)**assertEqual** (*first, second, msg=None*)

Fail if the two objects are unequal as determined by the `'=='` operator.

assertEquals (*first, second, msg=None*)

Fail if the two objects are unequal as determined by the `'=='` operator.

assertFalse (*expr, msg=None*)

Check that the expression is false.

assertGreater (*a, b, msg=None*)

Just like `self.assertTrue(a > b)`, but with a nicer default message.

assertGreaterEqual (*a, b, msg=None*)

Just like `self.assertTrue(a >= b)`, but with a nicer default message.

assertIn (*member, container, msg=None*)

Just like `self.assertTrue(a in b)`, but with a nicer default message.

assertIs (*expr1, expr2, msg=None*)

Just like `self.assertTrue(a is b)`, but with a nicer default message.

assertIsInstance (*obj, cls, msg=None*)

Same as `self.assertTrue(isinstance(obj, cls))`, with a nicer default message.

assertIsNone (*obj, msg=None*)

Same as `self.assertTrue(obj is None)`, with a nicer default message.

assertIsNot (*expr1, expr2, msg=None*)

Just like `self.assertTrue(a is not b)`, but with a nicer default message.

assertIsNotNone (*obj, msg=None*)

Included for symmetry with `assertIsNone`.

assertItemsEqual (*expected_seq, actual_seq, msg=None*)

An unordered sequence specific comparison. It asserts that `actual_seq` and `expected_seq` have the same element counts. Equivalent to:

Asserts that each element has the same count in both sequences. .. rubric:: Example

- `[0, 1, 1]` and `[1, 0, 1]` compare equal.
- `[0, 0, 1]` and `[0, 1]` compare unequal.

assertLess (*a, b, msg=None*)

Just like `self.assertTrue(a < b)`, but with a nicer default message.

assertLessEqual (*a, b, msg=None*)

Just like `self.assertTrue(a <= b)`, but with a nicer default message.

assertListEqual (*list1, list2, msg=None*)

A list-specific equality assertion.

Parameters

- **list1** – The first list to compare.
- **list2** – The second list to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assertMultiLineEqual (*first, second, msg=None*)

Assert that two multi-line strings are equal.

assertNotAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotAlmostEquals (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotEqual (*first, second, msg=None*)

Fail if the two objects are equal as determined by the `'!='` operator.

assertNotEquals (*first, second, msg=None*)

Fail if the two objects are equal as determined by the `'!='` operator.

assertNotIn (*member, container, msg=None*)

Just like `self.assertTrue(a not in b)`, but with a nicer default message.

assertNotIsInstance (*obj, cls, msg=None*)

Included for symmetry with `assertIsInstance`.

assertNotRegexMatches (*text, unexpected_regex, msg=None*)

Fail the test if the text matches the regular expression.

assertRaises (*excClass, callableObj=None, *args, **kwargs*)

Fail unless an exception of class *excClass* is raised by *callableObj* when invoked with arguments *args* and keyword arguments *kwargs*. If a different type of exception is raised, it will not be caught, and the test case will be deemed to have suffered an error, exactly as for an unexpected exception.

If called with *callableObj* omitted or *None*, will return a context object used like this:

The context manager keeps a reference to the exception as the ‘exception’ attribute. This allows you to inspect the exception after the assertion:

assertRaisesRegex (*expected_exception, expected_regex, callable_obj=None, *args, **kwargs*)

Asserts that the message in a raised exception matches a regex.

Parameters

- **expected_exception** – Exception class expected to be raised.
- **expected_regex** – Regex (re pattern object or string) expected to be found in error message.
- **callable_obj** – Function to be called.
- **args** – Extra args.
- **kwargs** – Extra kwargs.

assertRegexMatches (*text, expected_regex, msg=None*)

Fail the test unless the text matches the regular expression.

assertSequenceEqual (*seq1, seq2, msg=None, seq_type=None*)

An equality assertion for ordered sequences (like lists and tuples).

For the purposes of this function, a valid ordered sequence type is one which can be indexed, has a length, and has an equality operator.

Parameters

- **seq1** – The first sequence to compare.
- **seq2** – The second sequence to compare.
- **seq_type** – The expected datatype of the sequences, or *None* if no datatype should be enforced.
- **msg** – Optional message to use on failure instead of a list of differences.

assertSetEqual (*set1, set2, msg=None*)

A set-specific equality assertion.

Parameters

- **set1** – The first set to compare.
- **set2** – The second set to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

`assertSetEqual` uses ducktyping to support different types of sets, and is optimized for sets specifically (parameters must support a difference method).

assertTrue (*expr, msg=None*)

Check that the expression is true.

assertTupleEqual (*tuple1, tuple2, msg=None*)

A tuple-specific equality assertion.

Parameters

- **tuple1** – The first tuple to compare.
- **tuple2** – The second tuple to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assert_ (*expr, msg=None*)

Check that the expression is true.

base_test_pass (*name, environment, network, **kwargs*)

Basic test loop, requires an Agent to achieve a certain performance on an environment.

Parameters

- **name** (*str*) – The name of the test.
- **environment** ([Environment](#)) – The Environment object to use for the test.
- **network** ([LayerBasedNetwork](#)) – The Network to use for the agent’s model.
- **kwargs** (*any*) – Agent arguments.

base_test_run (*name, environment, network, **kwargs*)

Run test, tests whether algorithm can run and update without compilation errors, not whether it passes.

Parameters

- **name** (*str*) – The name of the test.
- **environment** ([Environment](#)) – The Environment object to use for the test.
- **network** ([LayerBasedNetwork](#)) – The Network to use for the agent’s model.
- **kwargs** (*any*) – Agent arguments.

```
config = {'update_mode': {'frequency': 4, 'batch_size': 4, 'unit': 'episodes'}, 's
```

```
countTestCases ()
```

```
debug ()
```

Run the test without collecting errors in a TestResult

```
defaultTestResult ()
```

```
doCleanups ()
```

Execute all cleanup functions. Normally called for you after tearDown.

```
exclude_bool = False
```

```
exclude_bounded = False
```

```
exclude_float = False
```

```
exclude_int = False
```

```
exclude_lstm = False
```

```
exclude_multi = False
```

```
fail (msg=None)
```

Fail immediately, with the given message.

```
failIf (*args, **kwargs)
```

failIfAlmostEqual (**args*, ***kwargs*)

failIfEqual (**args*, ***kwargs*)

failUnless (**args*, ***kwargs*)

failUnlessAlmostEqual (**args*, ***kwargs*)

failUnlessEqual (**args*, ***kwargs*)

failUnlessRaises (**args*, ***kwargs*)

failureException
alias of `AssertionError`

id ()

longMessage = **False**

maxDiff = 640

multi_config = **None**

pass_threshold = 0.8

pre_run (*agent*, *environment*)
Called before `Runner.run`.

requires_network = **True**

run (*result=None*)

setUp ()
Hook method for setting up the test fixture before exercising it.

setUpClass ()
Hook method for setting up class fixture before running tests in the class.

shortDescription ()
Returns a one-line description of the test, or `None` if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

skipTest (*reason*)
Skip this test.

tearDown ()
Hook method for deconstructing the test fixture after testing it.

tearDownClass ()
Hook method for deconstructing the class fixture after running all tests in the class.

test_bool ()
Tests the case of one boolean action.

test_bounded_float ()
Tests the case of one bounded float action, i.e. with min and max value.

test_float ()
Tests the case of one float action.

test_int ()
Tests the case of one integer action.

test_lstm ()
Tests the case of using internal states via an LSTM layer (for one integer action).

test_multi()

Tests the case of multiple actions of different type and shape.

tensorforce.tests.test_quickstart_example module

class tensorforce.tests.test_quickstart_example.**TestQuickstartExample** (*methodName='runTest'*)
 Bases: unittest.case.TestCase

__init__ (*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

addCleanup (*function, *args, **kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after tearDown on test failure or success.

Cleanup items are called even if setUp fails (unlike tearDown).

addTypeEqualityFunc (*typeobj, function*)

Add a type specific assertEquals style function to compare a type.

This method is for use by TestCase subclasses that need to register their own type equality functions to provide nicer error messages.

Parameters

- **typeobj** – The data type to call this function on when both values are of the same type in assertEquals().
- **function** – The callable taking two arguments and an optional msg= argument that raises self.failureException with a useful error message when the two arguments are not equal.

assertAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertAlmostEquals (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertDictContainsSubset (*expected, actual, msg=None*)

Checks whether actual is a superset of expected.

assertDictEqual (*d1, d2, msg=None*)

assertEqual (*first, second, msg=None*)

Fail if the two objects are unequal as determined by the '==' operator.

assertEquals (*first, second, msg=None*)

Fail if the two objects are unequal as determined by the '==' operator.

assertFalse (*expr, msg=None*)

Check that the expression is false.

assertGreater (*a, b, msg=None*)

Just like self.assertTrue(a > b), but with a nicer default message.

assertGreaterEqual (*a, b, msg=None*)

Just like self.assertTrue(a >= b), but with a nicer default message.

assertIn (*member, container, msg=None*)

Just like self.assertTrue(a in b), but with a nicer default message.

assertIs (*expr1, expr2, msg=None*)

Just like self.assertTrue(a is b), but with a nicer default message.

assertIsInstance (*obj, cls, msg=None*)

Same as self.assertTrue(isinstance(obj, cls)), with a nicer default message.

assertIsNone (*obj, msg=None*)

Same as self.assertTrue(obj is None), with a nicer default message.

assertIsNot (*expr1, expr2, msg=None*)

Just like self.assertTrue(a is not b), but with a nicer default message.

assertIsNotNone (*obj, msg=None*)

Included for symmetry with assertIsNone.

assertItemsEqual (*expected_seq, actual_seq, msg=None*)

An unordered sequence specific comparison. It asserts that actual_seq and expected_seq have the same element counts. Equivalent to:

Asserts that each element has the same count in both sequences. .. rubric:: Example

- [0, 1, 1] and [1, 0, 1] compare equal.
- [0, 0, 1] and [0, 1] compare unequal.

assertLess (*a, b, msg=None*)

Just like self.assertTrue(a < b), but with a nicer default message.

assertLessEqual (*a, b, msg=None*)

Just like self.assertTrue(a <= b), but with a nicer default message.

assertListEqual (*list1, list2, msg=None*)

A list-specific equality assertion.

Parameters

- **list1** – The first list to compare.
- **list2** – The second list to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assertMultiLineEqual (*first, second, msg=None*)

Assert that two multi-line strings are equal.

assertNotAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotAlmostEquals (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotEqual (*first, second, msg=None*)

Fail if the two objects are equal as determined by the '!=' operator.

assertNotEquals (*first, second, msg=None*)

Fail if the two objects are equal as determined by the '!=' operator.

assertNotIn (*member, container, msg=None*)

Just like self.assertTrue(a not in b), but with a nicer default message.

assertNotIsInstance (*obj, cls, msg=None*)

Included for symmetry with assertIsInstance.

assertNotRegexpMatches (*text, unexpected_regexp, msg=None*)

Fail the test if the text matches the regular expression.

assertRaises (*excClass, callableObj=None, *args, **kwargs*)

Fail unless an exception of class excClass is raised by callableObj when invoked with arguments args and keyword arguments kwargs. If a different type of exception is raised, it will not be caught, and the test case will be deemed to have suffered an error, exactly as for an unexpected exception.

If called with callableObj omitted or None, will return a context object used like this:

The context manager keeps a reference to the exception as the 'exception' attribute. This allows you to inspect the exception after the assertion:

assertRaisesRegexp (*expected_exception, expected_regexp, callable_obj=None, *args, **kwargs*)

Asserts that the message in a raised exception matches a regexp.

Parameters

- **expected_exception** – Exception class expected to be raised.
- **expected_regexp** – Regexp (re pattern object or string) expected to be found in error message.
- **callable_obj** – Function to be called.
- **args** – Extra args.
- **kwargs** – Extra kwargs.

assertRegexpMatches (*text, expected_regexp, msg=None*)

Fail the test unless the text matches the regular expression.

assertSequenceEqual (*seq1, seq2, msg=None, seq_type=None*)

An equality assertion for ordered sequences (like lists and tuples).

For the purposes of this function, a valid ordered sequence type is one which can be indexed, has a length, and has an equality operator.

Parameters

- **seq1** – The first sequence to compare.
- **seq2** – The second sequence to compare.
- **seq_type** – The expected datatype of the sequences, or None if no datatype should be enforced.
- **msg** – Optional message to use on failure instead of a list of differences.

assertSetEqual (*set1, set2, msg=None*)

A set-specific equality assertion.

Parameters

- **set1** – The first set to compare.
- **set2** – The second set to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assertSetEqual uses ducktyping to support different types of sets, and is optimized for sets specifically (parameters must support a difference method).

assertTrue (*expr, msg=None*)

Check that the expression is true.

assertTupleEqual (*tuple1, tuple2, msg=None*)

A tuple-specific equality assertion.

Parameters

- **tuple1** – The first tuple to compare.
- **tuple2** – The second tuple to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assert_ (*expr, msg=None*)

Check that the expression is true.

countTestCases ()

debug ()

Run the test without collecting errors in a TestResult

defaultTestResult ()

doCleanups ()

Execute all cleanup functions. Normally called for you after tearDown.

fail (*msg=None*)

Fail immediately, with the given message.

failIf (**args, **kwargs*)

failIfAlmostEqual (**args, **kwargs*)

failIfEqual (**args, **kwargs*)

failUnless (**args, **kwargs*)

failUnlessAlmostEqual (**args, **kwargs*)

failUnlessEqual (**args, **kwargs*)

failUnlessRaises (**args, **kwargs*)

failureException
alias of AssertionError

id()

longMessage = False

maxDiff = 640

run (result=None)

setUp()
Hook method for setting up the test fixture before exercising it.

setUpClass()
Hook method for setting up class fixture before running tests in the class.

shortDescription()
Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

skipTest (reason)
Skip this test.

tearDown()
Hook method for deconstructing the test fixture after testing it.

tearDownClass()
Hook method for deconstructing the class fixture after running all tests in the class.

test_example()

tensorforce.tests.test_random_agent module

class tensorforce.tests.test_random_agent.**TestRandomAgent** (*methodName='runTest'*)
Bases: *tensorforce.tests.base_agent_test.BaseAgentTest*, *unittest.case.TestCase*

__init__ (*methodName='runTest'*)
Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

addCleanup (*function, *args, **kwargs*)
Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after `tearDown` on test failure or success.

Cleanup items are called even if `setUp` fails (unlike `tearDown`).

addTypeEqualityFunc (*typeobj, function*)
Add a type specific `assertEqual` style function to compare a type.

This method is for use by `TestCase` subclasses that need to register their own type equality functions to provide nicer error messages.

Parameters

- **typeobj** – The data type to call this function on when both values are of the same type in `assertEqual()`.
- **function** – The callable taking two arguments and an optional `msg=` argument that raises `self.failureException` with a useful error message when the two arguments are not equal.

agent

alias of RandomAgent

assertAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertAlmostEquals (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertDictContainsSubset (*expected, actual, msg=None*)

Checks whether actual is a superset of expected.

assertDictEqual (*d1, d2, msg=None*)**assertEqual** (*first, second, msg=None*)

Fail if the two objects are unequal as determined by the '==' operator.

assertEquals (*first, second, msg=None*)

Fail if the two objects are unequal as determined by the '==' operator.

assertFalse (*expr, msg=None*)

Check that the expression is false.

assertGreater (*a, b, msg=None*)

Just like self.assertTrue(a > b), but with a nicer default message.

assertGreaterEqual (*a, b, msg=None*)

Just like self.assertTrue(a >= b), but with a nicer default message.

assertIn (*member, container, msg=None*)

Just like self.assertTrue(a in b), but with a nicer default message.

assertIs (*expr1, expr2, msg=None*)

Just like self.assertTrue(a is b), but with a nicer default message.

assertIsInstance (*obj, cls, msg=None*)

Same as self.assertTrue(isinstance(obj, cls)), with a nicer default message.

assertIsNone (*obj, msg=None*)

Same as self.assertTrue(obj is None), with a nicer default message.

assertIsNot (*expr1, expr2, msg=None*)

Just like self.assertTrue(a is not b), but with a nicer default message.

assertIsNotNone (*obj, msg=None*)

Included for symmetry with assertIsNone.

assertItemsEqual (*expected_seq, actual_seq, msg=None*)

An unordered sequence specific comparison. It asserts that actual_seq and expected_seq have the same element counts. Equivalent to:

Asserts that each element has the same count in both sequences. .. rubric:: Example

- [0, 1, 1] and [1, 0, 1] compare equal.
- [0, 0, 1] and [0, 1] compare unequal.

assertLess (*a, b, msg=None*)

Just like self.assertTrue(*a < b*), but with a nicer default message.

assertLessEqual (*a, b, msg=None*)

Just like self.assertTrue(*a <= b*), but with a nicer default message.

assertListEqual (*list1, list2, msg=None*)

A list-specific equality assertion.

Parameters

- **list1** – The first list to compare.
- **list2** – The second list to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assertMultiLineEqual (*first, second, msg=None*)

Assert that two multi-line strings are equal.

assertNotAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotAlmostEquals (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotEqual (*first, second, msg=None*)

Fail if the two objects are equal as determined by the '!=' operator.

assertNotEquals (*first, second, msg=None*)

Fail if the two objects are equal as determined by the '!=' operator.

assertNotIn (*member, container, msg=None*)

Just like self.assertTrue(*a not in b*), but with a nicer default message.

assertNotIsInstance (*obj, cls, msg=None*)

Included for symmetry with `assertIsInstance`.

assertNotRegexpMatches (*text, unexpected_regexp, msg=None*)

Fail the test if the text matches the regular expression.

assertRaises (*excClass, callableObj=None, *args, **kwargs*)

Fail unless an exception of class `excClass` is raised by `callableObj` when invoked with arguments `args` and keyword arguments `kwargs`. If a different type of exception is raised, it will not be caught, and the test case will be deemed to have suffered an error, exactly as for an unexpected exception.

If called with `callableObj` omitted or `None`, will return a context object used like this:

The context manager keeps a reference to the exception as the ‘exception’ attribute. This allows you to inspect the exception after the assertion:

assertRaisesRegexp (*expected_exception, expected_regexp, callable_obj=None, *args, **kwargs*)

Asserts that the message in a raised exception matches a regexp.

Parameters

- **expected_exception** – Exception class expected to be raised.
- **expected_regexp** – Regexp (re pattern object or string) expected to be found in error message.
- **callable_obj** – Function to be called.
- **args** – Extra args.
- **kwargs** – Extra kwargs.

assertRegexpMatches (*text, expected_regexp, msg=None*)

Fail the test unless the text matches the regular expression.

assertSequenceEqual (*seq1, seq2, msg=None, seq_type=None*)

An equality assertion for ordered sequences (like lists and tuples).

For the purposes of this function, a valid ordered sequence type is one which can be indexed, has a length, and has an equality operator.

Parameters

- **seq1** – The first sequence to compare.
- **seq2** – The second sequence to compare.
- **seq_type** – The expected datatype of the sequences, or `None` if no datatype should be enforced.
- **msg** – Optional message to use on failure instead of a list of differences.

assertSetEqual (*set1, set2, msg=None*)

A set-specific equality assertion.

Parameters

- **set1** – The first set to compare.
- **set2** – The second set to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

`assertSetEqual` uses ducktyping to support different types of sets, and is optimized for sets specifically (parameters must support a difference method).

assertTrue (*expr, msg=None*)

Check that the expression is true.

assertTupleEqual (*tuple1, tuple2, msg=None*)

A tuple-specific equality assertion.

Parameters

- **tuple1** – The first tuple to compare.
- **tuple2** – The second tuple to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assert_ (*expr*, *msg=None*)

Check that the expression is true.

base_test_pass (*name*, *environment*, *network*, ***kwargs*)

Basic test loop, requires an Agent to achieve a certain performance on an environment.

Parameters

- **name** (*str*) – The name of the test.
- **environment** (*Environment*) – The Environment object to use for the test.
- **network** (*LayerBasedNetwork*) – The Network to use for the agent’s model.
- **kwargs** (*any*) – Agent arguments.

base_test_run (*name*, *environment*, *network*, ***kwargs*)

Run test, tests whether algorithm can run and update without compilation errors, not whether it passes.

Parameters

- **name** (*str*) – The name of the test.
- **environment** (*Environment*) – The Environment object to use for the test.
- **network** (*LayerBasedNetwork*) – The Network to use for the agent’s model.
- **kwargs** (*any*) – Agent arguments.

config = {}

countTestCases ()

debug ()

Run the test without collecting errors in a TestResult

defaultTestResult ()

doCleanups ()

Execute all cleanup functions. Normally called for you after `tearDown`.

exclude_bool = False

exclude_bounded = False

exclude_float = False

exclude_int = False

exclude_lstm = True

exclude_multi = False

fail (*msg=None*)

Fail immediately, with the given message.

failIf (**args*, ***kwargs*)

failIfAlmostEqual (**args*, ***kwargs*)

failIfEqual (**args*, ***kwargs*)

failUnless (**args*, ***kwargs*)

failUnlessAlmostEqual (**args*, ***kwargs*)

failUnlessEqual (**args*, ***kwargs*)

failUnlessRaises (**args*, ***kwargs*)

failureException
alias of `AssertionError`

id()

longMessage = False

maxDiff = 640

multi_config = None

pass_threshold = 0.0

pre_run (*agent, environment*)
Called before `Runner.run`.

requires_network = False

run (*result=None*)

setUp()
Hook method for setting up the test fixture before exercising it.

setUpClass()
Hook method for setting up class fixture before running tests in the class.

shortDescription()
Returns a one-line description of the test, or `None` if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

skipTest (*reason*)
Skip this test.

tearDown()
Hook method for deconstructing the test fixture after testing it.

tearDownClass()
Hook method for deconstructing the class fixture after running all tests in the class.

test_bool()
Tests the case of one boolean action.

test_bounded_float()
Tests the case of one bounded float action, i.e. with min and max value.

test_float()
Tests the case of one float action.

test_int()
Tests the case of one integer action.

test_lstm()
Tests the case of using internal states via an LSTM layer (for one integer action).

test_multi()
Tests the case of multiple actions of different type and shape.

tensorforce.tests.test_reward_estimation module

```
class tensorforce.tests.test_reward_estimation.TestRewardEstimation (methodName='runTest')  
    Bases: unittest.case.TestCase
```

__init__ (*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

addCleanup (*function, *args, **kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after `tearDown` on test failure or success.

Cleanup items are called even if `setUp` fails (unlike `tearDown`).

addTypeEqualityFunc (*typeobj, function*)

Add a type specific `assertEqual` style function to compare a type.

This method is for use by `TestCase` subclasses that need to register their own type equality functions to provide nicer error messages.

Parameters

- **typeobj** – The data type to call this function on when both values are of the same type in `assertEqual()`.
- **function** – The callable taking two arguments and an optional `msg=` argument that raises `self.failureException` with a useful error message when the two arguments are not equal.

assertAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertAlmostEquals (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertDictContainsSubset (*expected, actual, msg=None*)

Checks whether `actual` is a superset of `expected`.

assertDictEqual (*d1, d2, msg=None*)

assertEqual (*first, second, msg=None*)

Fail if the two objects are unequal as determined by the `'=='` operator.

assertEquals (*first, second, msg=None*)

Fail if the two objects are unequal as determined by the `'=='` operator.

assertFalse (*expr, msg=None*)

Check that the expression is false.

assertGreater (*a, b, msg=None*)

Just like `self.assertTrue(a > b)`, but with a nicer default message.

assertGreaterEqual (*a, b, msg=None*)

Just like `self.assertTrue(a >= b)`, but with a nicer default message.

assertIn (*member, container, msg=None*)

Just like self.assertTrue(a in b), but with a nicer default message.

assertIs (*expr1, expr2, msg=None*)

Just like self.assertTrue(a is b), but with a nicer default message.

assertIsInstance (*obj, cls, msg=None*)

Same as self.assertTrue(isinstance(obj, cls)), with a nicer default message.

assertIsNone (*obj, msg=None*)

Same as self.assertTrue(obj is None), with a nicer default message.

assertIsNot (*expr1, expr2, msg=None*)

Just like self.assertTrue(a is not b), but with a nicer default message.

assertIsNotNone (*obj, msg=None*)

Included for symmetry with assertIsNone.

assertItemsEqual (*expected_seq, actual_seq, msg=None*)

An unordered sequence specific comparison. It asserts that actual_seq and expected_seq have the same element counts. Equivalent to:

Asserts that each element has the same count in both sequences. .. rubric:: Example

- [0, 1, 1] and [1, 0, 1] compare equal.
- [0, 0, 1] and [0, 1] compare unequal.

assertLess (*a, b, msg=None*)

Just like self.assertTrue(a < b), but with a nicer default message.

assertLessEqual (*a, b, msg=None*)

Just like self.assertTrue(a <= b), but with a nicer default message.

assertListEqual (*list1, list2, msg=None*)

A list-specific equality assertion.

Parameters

- **list1** – The first list to compare.
- **list2** – The second list to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assertMultiLineEqual (*first, second, msg=None*)

Assert that two multi-line strings are equal.

assertNotAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotAlmostEquals (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotEqual (*first, second, msg=None*)

Fail if the two objects are equal as determined by the ‘!=’ operator.

assertNotEquals (*first, second, msg=None*)

Fail if the two objects are equal as determined by the ‘!=’ operator.

assertNotIn (*member, container, msg=None*)

Just like self.assertTrue(a not in b), but with a nicer default message.

assertNotIsInstance (*obj, cls, msg=None*)

Included for symmetry with assertIsInstance.

assertNotRegexpMatches (*text, unexpected_regexp, msg=None*)

Fail the test if the text matches the regular expression.

assertRaises (*excClass, callableObj=None, *args, **kwargs*)

Fail unless an exception of class excClass is raised by callableObj when invoked with arguments args and keyword arguments kwargs. If a different type of exception is raised, it will not be caught, and the test case will be deemed to have suffered an error, exactly as for an unexpected exception.

If called with callableObj omitted or None, will return a context object used like this:

The context manager keeps a reference to the exception as the ‘exception’ attribute. This allows you to inspect the exception after the assertion:

assertRaisesRegexp (*expected_exception, expected_regexp, callable_obj=None, *args, **kwargs*)

Asserts that the message in a raised exception matches a regexp.

Parameters

- **expected_exception** – Exception class expected to be raised.
- **expected_regexp** – Regexp (re pattern object or string) expected to be found in error message.
- **callable_obj** – Function to be called.
- **args** – Extra args.
- **kwargs** – Extra kwargs.

assertRegexpMatches (*text, expected_regexp, msg=None*)

Fail the test unless the text matches the regular expression.

assertSequenceEqual (*seq1, seq2, msg=None, seq_type=None*)

An equality assertion for ordered sequences (like lists and tuples).

For the purposes of this function, a valid ordered sequence type is one which can be indexed, has a length, and has an equality operator.

Parameters

- **seq1** – The first sequence to compare.
- **seq2** – The second sequence to compare.
- **seq_type** – The expected datatype of the sequences, or None if no datatype should be enforced.
- **msg** – Optional message to use on failure instead of a list of differences.

assertSetEqual (*set1, set2, msg=None*)

A set-specific equality assertion.

Parameters

- **set1** – The first set to compare.
- **set2** – The second set to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

`assertSetEqual` uses ducktyping to support different types of sets, and is optimized for sets specifically (parameters must support a difference method).

assertTrue (*expr*, *msg=None*)

Check that the expression is true.

assertTupleEqual (*tuple1*, *tuple2*, *msg=None*)

A tuple-specific equality assertion.

Parameters

- **tuple1** – The first tuple to compare.
- **tuple2** – The second tuple to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assert_ (*expr*, *msg=None*)

Check that the expression is true.

countTestCases ()

debug ()

Run the test without collecting errors in a `TestResult`

defaultTestResult ()

doCleanups ()

Execute all cleanup functions. Normally called for you after `tearDown`.

fail (*msg=None*)

Fail immediately, with the given message.

failIf (**args*, ***kwargs*)

failIfAlmostEqual (**args*, ***kwargs*)

failIfEqual (**args*, ***kwargs*)

failUnless (**args*, ***kwargs*)

failUnlessAlmostEqual (**args*, ***kwargs*)

failUnlessEqual (**args*, ***kwargs*)

failUnlessRaises (**args*, ***kwargs*)

failureException

alias of `AssertionError`

id ()

longMessage = **False**

maxDiff = 640

run (*result=None*)

setUp ()

Hook method for setting up the test fixture before exercising it.

setUpClass()

Hook method for setting up class fixture before running tests in the class.

shortDescription()

Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

skipTest(reason)

Skip this test.

tearDown()

Hook method for deconstructing the test fixture after testing it.

tearDownClass()

Hook method for deconstructing the class fixture after running all tests in the class.

test_baseline()**test_basic()****test_gae()****tensorforce.tests.test_trpo_agent module**

class tensorforce.tests.test_trpo_agent.**TestTRPOAgent** (*methodName='runTest'*)

Bases: *tensorforce.tests.base_agent_test.BaseAgentTest*, *unittest.case.TestCase*

__init__ (*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

addCleanup (*function, *args, **kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after tearDown on test failure or success.

Cleanup items are called even if setUp fails (unlike tearDown).

addTypeEqualityFunc (*typeobj, function*)

Add a type specific assertEquals style function to compare a type.

This method is for use by TestCase subclasses that need to register their own type equality functions to provide nicer error messages.

Parameters

- **typeobj** – The data type to call this function on when both values are of the same type in assertEquals().
- **function** – The callable taking two arguments and an optional msg= argument that raises self.failureException with a useful error message when the two arguments are not equal.

agent

alias of TRPOAgent

assertAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertAlmostEquals (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertDictContainsSubset (*expected, actual, msg=None*)

Checks whether actual is a superset of expected.

assertDictEqual (*d1, d2, msg=None*)

assertEqual (*first, second, msg=None*)

Fail if the two objects are unequal as determined by the '==' operator.

assertEquals (*first, second, msg=None*)

Fail if the two objects are unequal as determined by the '==' operator.

assertFalse (*expr, msg=None*)

Check that the expression is false.

assertGreater (*a, b, msg=None*)

Just like self.assertTrue(a > b), but with a nicer default message.

assertGreaterEqual (*a, b, msg=None*)

Just like self.assertTrue(a >= b), but with a nicer default message.

assertIn (*member, container, msg=None*)

Just like self.assertTrue(a in b), but with a nicer default message.

assertIs (*expr1, expr2, msg=None*)

Just like self.assertTrue(a is b), but with a nicer default message.

assertIsInstance (*obj, cls, msg=None*)

Same as self.assertTrue(isinstance(obj, cls)), with a nicer default message.

assertIsNone (*obj, msg=None*)

Same as self.assertTrue(obj is None), with a nicer default message.

assertIsNot (*expr1, expr2, msg=None*)

Just like self.assertTrue(a is not b), but with a nicer default message.

assertIsNotNone (*obj, msg=None*)

Included for symmetry with assertIsNone.

assertItemsEqual (*expected_seq, actual_seq, msg=None*)

An unordered sequence specific comparison. It asserts that actual_seq and expected_seq have the same element counts. Equivalent to:

Asserts that each element has the same count in both sequences. .. rubric:: Example

- [0, 1, 1] and [1, 0, 1] compare equal.
- [0, 0, 1] and [0, 1] compare unequal.

assertLess (*a, b, msg=None*)

Just like self.assertTrue(a < b), but with a nicer default message.

assertLessEqual (*a, b, msg=None*)

Just like self.assertTrue(a <= b), but with a nicer default message.

assertListEqual (*list1, list2, msg=None*)

A list-specific equality assertion.

Parameters

- **list1** – The first list to compare.
- **list2** – The second list to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assertMultiLineEqual (*first, second, msg=None*)

Assert that two multi-line strings are equal.

assertNotAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotAlmostEquals (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotEqual (*first, second, msg=None*)

Fail if the two objects are equal as determined by the '!=' operator.

assertNotEquals (*first, second, msg=None*)

Fail if the two objects are equal as determined by the '!=' operator.

assertNotIn (*member, container, msg=None*)

Just like self.assertTrue(a not in b), but with a nicer default message.

assertNotIsInstance (*obj, cls, msg=None*)

Included for symmetry with assertIsInstance.

assertNotRegexpMatches (*text, unexpected_regexp, msg=None*)

Fail the test if the text matches the regular expression.

assertRaises (*excClass, callableObj=None, *args, **kwargs*)

Fail unless an exception of class excClass is raised by callableObj when invoked with arguments args and keyword arguments kwargs. If a different type of exception is raised, it will not be caught, and the test case will be deemed to have suffered an error, exactly as for an unexpected exception.

If called with callableObj omitted or None, will return a context object used like this:

The context manager keeps a reference to the exception as the 'exception' attribute. This allows you to inspect the exception after the assertion:

assertRaisesRegexp (*expected_exception, expected_regexp, callable_obj=None, *args, **kwargs*)

Asserts that the message in a raised exception matches a regexp.

Parameters

- **expected_exception** – Exception class expected to be raised.
- **expected_regexp** – Regexp (re pattern object or string) expected to be found in error message.
- **callable_obj** – Function to be called.
- **args** – Extra args.
- **kwargs** – Extra kwargs.

assertRegexpMatches (*text, expected_regexp, msg=None*)

Fail the test unless the text matches the regular expression.

assertSequenceEqual (*seq1, seq2, msg=None, seq_type=None*)

An equality assertion for ordered sequences (like lists and tuples).

For the purposes of this function, a valid ordered sequence type is one which can be indexed, has a length, and has an equality operator.

Parameters

- **seq1** – The first sequence to compare.
- **seq2** – The second sequence to compare.
- **seq_type** – The expected datatype of the sequences, or None if no datatype should be enforced.
- **msg** – Optional message to use on failure instead of a list of differences.

assertSetEqual (*set1, set2, msg=None*)

A set-specific equality assertion.

Parameters

- **set1** – The first set to compare.
- **set2** – The second set to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assertSetEqual uses ducktyping to support different types of sets, and is optimized for sets specifically (parameters must support a difference method).

assertTrue (*expr, msg=None*)

Check that the expression is true.

assertTupleEqual (*tuple1, tuple2, msg=None*)

A tuple-specific equality assertion.

Parameters

- **tuple1** – The first tuple to compare.
- **tuple2** – The second tuple to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assert_ (*expr, msg=None*)

Check that the expression is true.

base_test_pass (*name, environment, network, **kwargs*)

Basic test loop, requires an Agent to achieve a certain performance on an environment.

Parameters

- **name** (*str*) – The name of the test.
- **environment** (*Environment*) – The Environment object to use for the test.
- **network** (*LayerBasedNetwork*) – The Network to use for the agent’s model.
- **kwargs** (*any*) – Agent arguments.

base_test_run (*name, environment, network, **kwargs*)

Run test, tests whether algorithm can run and update without compilation errors, not whether it passes.

Parameters

- **name** (*str*) – The name of the test.
- **environment** (*Environment*) – The Environment object to use for the test.
- **network** (*LayerBasedNetwork*) – The Network to use for the agent’s model.
- **kwargs** (*any*) – Agent arguments.

config = {'update_mode': {'frequency': 4, 'batch_size': 4, 'unit': 'episodes'}, 'l

countTestCases ()

debug ()

Run the test without collecting errors in a TestResult

defaultTestResult ()

doCleanups ()

Execute all cleanup functions. Normally called for you after tearDown.

exclude_bool = False

exclude_bounded = False

exclude_float = False

exclude_int = False

exclude_lstm = False

exclude_multi = False

fail (*msg=None*)

Fail immediately, with the given message.

failIf (**args, **kwargs*)

failIfAlmostEqual (**args, **kwargs*)

failIfEqual (**args, **kwargs*)

failUnless (**args, **kwargs*)

failUnlessAlmostEqual (**args, **kwargs*)

failUnlessEqual (**args, **kwargs*)

failUnlessRaises (**args, **kwargs*)

failureException

alias of AssertionError

id ()

longMessage = False

maxDiff = 640

multi_config = None

pass_threshold = 0.8

pre_run (*agent, environment*)
Called before `Runner.run`.

requires_network = True

run (*result=None*)

setUp ()
Hook method for setting up the test fixture before exercising it.

setUpClass ()
Hook method for setting up class fixture before running tests in the class.

shortDescription ()
Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

skipTest (*reason*)
Skip this test.

tearDown ()
Hook method for deconstructing the test fixture after testing it.

tearDownClass ()
Hook method for deconstructing the class fixture after running all tests in the class.

test_bool ()
Tests the case of one boolean action.

test_bounded_float ()
Tests the case of one bounded float action, i.e. with min and max value.

test_float ()
Tests the case of one float action.

test_int ()
Tests the case of one integer action.

test_lstm ()
Tests the case of using internal states via an LSTM layer (for one integer action).

test_multi ()
Tests the case of multiple actions of different type and shape.

tensorforce.tests.test_tutorial_code module

class tensorforce.tests.test_tutorial_code.**TestTutorialCode** (*methodName='runTest'*)
Bases: `unittest.case.TestCase`

Validation of random code snippets as to be notified when old blog posts need to be changed.

class **MyClient** (**args, **kwargs*)
Bases: `object`

__init__ (**args, **kwargs*)

execute (*action*)

get_state ()

__init__ (*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

addCleanup (*function, *args, **kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after `tearDown` on test failure or success.

Cleanup items are called even if `setUp` fails (unlike `tearDown`).

addTypeEqualityFunc (*typeobj, function*)

Add a type specific `assertEqual` style function to compare a type.

This method is for use by `TestCase` subclasses that need to register their own type equality functions to provide nicer error messages.

Parameters

- **typeobj** – The data type to call this function on when both values are of the same type in `assertEqual()`.
- **function** – The callable taking two arguments and an optional `msg=` argument that raises `self.failureException` with a useful error message when the two arguments are not equal.

assertAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertAlmostEquals (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertDictContainsSubset (*expected, actual, msg=None*)

Checks whether `actual` is a superset of `expected`.

assertDictEqual (*d1, d2, msg=None*)

assertEqual (*first, second, msg=None*)

Fail if the two objects are unequal as determined by the `'=='` operator.

assertEquals (*first, second, msg=None*)

Fail if the two objects are unequal as determined by the `'=='` operator.

assertFalse (*expr, msg=None*)

Check that the expression is false.

assertGreater (*a, b, msg=None*)

Just like `self.assertTrue(a > b)`, but with a nicer default message.

assertGreaterEqual (*a, b, msg=None*)

Just like `self.assertTrue(a >= b)`, but with a nicer default message.

assertIn (*member, container, msg=None*)

Just like self.assertTrue(a in b), but with a nicer default message.

assertIs (*expr1, expr2, msg=None*)

Just like self.assertTrue(a is b), but with a nicer default message.

assertIsInstance (*obj, cls, msg=None*)

Same as self.assertTrue(isinstance(obj, cls)), with a nicer default message.

assertIsNone (*obj, msg=None*)

Same as self.assertTrue(obj is None), with a nicer default message.

assertIsNot (*expr1, expr2, msg=None*)

Just like self.assertTrue(a is not b), but with a nicer default message.

assertIsNotNone (*obj, msg=None*)

Included for symmetry with assertIsNone.

assertItemsEqual (*expected_seq, actual_seq, msg=None*)

An unordered sequence specific comparison. It asserts that actual_seq and expected_seq have the same element counts. Equivalent to:

Asserts that each element has the same count in both sequences. .. rubric:: Example

- [0, 1, 1] and [1, 0, 1] compare equal.
- [0, 0, 1] and [0, 1] compare unequal.

assertLess (*a, b, msg=None*)

Just like self.assertTrue(a < b), but with a nicer default message.

assertLessEqual (*a, b, msg=None*)

Just like self.assertTrue(a <= b), but with a nicer default message.

assertListEqual (*list1, list2, msg=None*)

A list-specific equality assertion.

Parameters

- **list1** – The first list to compare.
- **list2** – The second list to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assertMultiLineEqual (*first, second, msg=None*)

Assert that two multi-line strings are equal.

assertNotAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotAlmostEquals (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotEqual (*first, second, msg=None*)

Fail if the two objects are equal as determined by the ‘!=’ operator.

assertNotEquals (*first, second, msg=None*)

Fail if the two objects are equal as determined by the ‘!=’ operator.

assertNotIn (*member, container, msg=None*)

Just like self.assertTrue(a not in b), but with a nicer default message.

assertNotIsInstance (*obj, cls, msg=None*)

Included for symmetry with assertIsInstance.

assertNotRegexpMatches (*text, unexpected_regexp, msg=None*)

Fail the test if the text matches the regular expression.

assertRaises (*excClass, callableObj=None, *args, **kwargs*)

Fail unless an exception of class excClass is raised by callableObj when invoked with arguments args and keyword arguments kwargs. If a different type of exception is raised, it will not be caught, and the test case will be deemed to have suffered an error, exactly as for an unexpected exception.

If called with callableObj omitted or None, will return a context object used like this:

The context manager keeps a reference to the exception as the ‘exception’ attribute. This allows you to inspect the exception after the assertion:

assertRaisesRegexp (*expected_exception, expected_regexp, callable_obj=None, *args, **kwargs*)

Asserts that the message in a raised exception matches a regexp.

Parameters

- **expected_exception** – Exception class expected to be raised.
- **expected_regexp** – Regexp (re pattern object or string) expected to be found in error message.
- **callable_obj** – Function to be called.
- **args** – Extra args.
- **kwargs** – Extra kwargs.

assertRegexpMatches (*text, expected_regexp, msg=None*)

Fail the test unless the text matches the regular expression.

assertSequenceEqual (*seq1, seq2, msg=None, seq_type=None*)

An equality assertion for ordered sequences (like lists and tuples).

For the purposes of this function, a valid ordered sequence type is one which can be indexed, has a length, and has an equality operator.

Parameters

- **seq1** – The first sequence to compare.
- **seq2** – The second sequence to compare.
- **seq_type** – The expected datatype of the sequences, or None if no datatype should be enforced.
- **msg** – Optional message to use on failure instead of a list of differences.

assertSetEqual (*set1, set2, msg=None*)

A set-specific equality assertion.

Parameters

- **set1** – The first set to compare.
- **set2** – The second set to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

`assertSetEqual` uses ducktyping to support different types of sets, and is optimized for sets specifically (parameters must support a difference method).

assertTrue (*expr*, *msg=None*)

Check that the expression is true.

assertTupleEqual (*tuple1*, *tuple2*, *msg=None*)

A tuple-specific equality assertion.

Parameters

- **tuple1** – The first tuple to compare.
- **tuple2** – The second tuple to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assert_ (*expr*, *msg=None*)

Check that the expression is true.

countTestCases ()

debug ()

Run the test without collecting errors in a `TestResult`

defaultTestResult ()

doCleanups ()

Execute all cleanup functions. Normally called for you after `tearDown`.

fail (*msg=None*)

Fail immediately, with the given message.

failIf (**args*, ***kwargs*)

failIfAlmostEqual (**args*, ***kwargs*)

failIfEqual (**args*, ***kwargs*)

failUnless (**args*, ***kwargs*)

failUnlessAlmostEqual (**args*, ***kwargs*)

failUnlessEqual (**args*, ***kwargs*)

failUnlessRaises (**args*, ***kwargs*)

failureException

alias of `AssertionError`

id ()

longMessage = **False**

maxDiff = 640

run (*result=None*)

setUp ()

Hook method for setting up the test fixture before exercising it.

setUpClass()
Hook method for setting up class fixture before running tests in the class.

shortDescription()
Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

skipTest(reason)
Skip this test.

tearDown()
Hook method for deconstructing the test fixture after testing it.

tearDownClass()
Hook method for deconstructing the class fixture after running all tests in the class.

test_blogpost_introduction()
Test of introduction blog post examples.

test_blogpost_introduction_runner()

test_reinforceio_homepage()
Code example from the homepage and README.md.

tensorforce.tests.test_vpg_agent module

class tensorforce.tests.test_vpg_agent.**TestVPGAgent** (*methodName='runTest'*)
Bases: *tensorforce.tests.base_agent_test.BaseAgentTest*, *unittest.case.TestCase*

__init__ (*methodName='runTest'*)
Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

addCleanup (*function, *args, **kwargs*)
Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after **tearDown** on test failure or success.

Cleanup items are called even if **setUp** fails (unlike **tearDown**).

addTypeEqualityFunc (*typeobj, function*)
Add a type specific **assertEqual** style function to compare a type.

This method is for use by **TestCase** subclasses that need to register their own type equality functions to provide nicer error messages.

Parameters

- **typeobj** – The data type to call this function on when both values are of the same type in **assertEqual()**.
- **function** – The callable taking two arguments and an optional **msg=** argument that raises **self.failureException** with a useful error message when the two arguments are not equal.

agent
alias of **VPGAgent**

assertAlmostEqual (*first, second, places=None, msg=None, delta=None*)
Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal

places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertAlmostEquals (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertDictContainsSubset (*expected, actual, msg=None*)

Checks whether actual is a superset of expected.

assertDictEqual (*d1, d2, msg=None*)

assertEqual (*first, second, msg=None*)

Fail if the two objects are unequal as determined by the '==' operator.

assertEquals (*first, second, msg=None*)

Fail if the two objects are unequal as determined by the '==' operator.

assertFalse (*expr, msg=None*)

Check that the expression is false.

assertGreater (*a, b, msg=None*)

Just like self.assertTrue(a > b), but with a nicer default message.

assertGreaterEqual (*a, b, msg=None*)

Just like self.assertTrue(a >= b), but with a nicer default message.

assertIn (*member, container, msg=None*)

Just like self.assertTrue(a in b), but with a nicer default message.

assertIs (*expr1, expr2, msg=None*)

Just like self.assertTrue(a is b), but with a nicer default message.

assertIsInstance (*obj, cls, msg=None*)

Same as self.assertTrue(isinstance(obj, cls)), with a nicer default message.

assertIsNone (*obj, msg=None*)

Same as self.assertTrue(obj is None), with a nicer default message.

assertIsNot (*expr1, expr2, msg=None*)

Just like self.assertTrue(a is not b), but with a nicer default message.

assertIsNotNone (*obj, msg=None*)

Included for symmetry with assertIsNone.

assertItemsEqual (*expected_seq, actual_seq, msg=None*)

An unordered sequence specific comparison. It asserts that actual_seq and expected_seq have the same element counts. Equivalent to:

Asserts that each element has the same count in both sequences. .. rubric:: Example

- [0, 1, 1] and [1, 0, 1] compare equal.
- [0, 0, 1] and [0, 1] compare unequal.

assertLess (*a, b, msg=None*)

Just like self.assertTrue(*a < b*), but with a nicer default message.

assertLessEqual (*a, b, msg=None*)

Just like self.assertTrue(*a <= b*), but with a nicer default message.

assertListEqual (*list1, list2, msg=None*)

A list-specific equality assertion.

Parameters

- **list1** – The first list to compare.
- **list2** – The second list to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assertMultiLineEqual (*first, second, msg=None*)

Assert that two multi-line strings are equal.

assertNotAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotAlmostEquals (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotEqual (*first, second, msg=None*)

Fail if the two objects are equal as determined by the '!=' operator.

assertNotEquals (*first, second, msg=None*)

Fail if the two objects are equal as determined by the '!=' operator.

assertNotIn (*member, container, msg=None*)

Just like self.assertTrue(*a not in b*), but with a nicer default message.

assertNotIsInstance (*obj, cls, msg=None*)

Included for symmetry with `assertIsInstance`.

assertNotRegexpMatches (*text, unexpected_regexp, msg=None*)

Fail the test if the text matches the regular expression.

assertRaises (*excClass, callableObj=None, *args, **kwargs*)

Fail unless an exception of class `excClass` is raised by `callableObj` when invoked with arguments `args` and keyword arguments `kwargs`. If a different type of exception is raised, it will not be caught, and the test case will be deemed to have suffered an error, exactly as for an unexpected exception.

If called with `callableObj` omitted or `None`, will return a context object used like this:

The context manager keeps a reference to the exception as the 'exception' attribute. This allows you to inspect the exception after the assertion:

assertRaisesRegexp (*expected_exception, expected_regexp, callable_obj=None, *args, **kwargs*)

Asserts that the message in a raised exception matches a regexp.

Parameters

- **expected_exception** – Exception class expected to be raised.
- **expected_regexp** – Regexp (re pattern object or string) expected to be found in error message.
- **callable_obj** – Function to be called.
- **args** – Extra args.
- **kwargs** – Extra kwargs.

assertRegexpMatches (*text, expected_regexp, msg=None*)

Fail the test unless the text matches the regular expression.

assertSequenceEqual (*seq1, seq2, msg=None, seq_type=None*)

An equality assertion for ordered sequences (like lists and tuples).

For the purposes of this function, a valid ordered sequence type is one which can be indexed, has a length, and has an equality operator.

Parameters

- **seq1** – The first sequence to compare.
- **seq2** – The second sequence to compare.
- **seq_type** – The expected datatype of the sequences, or None if no datatype should be enforced.
- **msg** – Optional message to use on failure instead of a list of differences.

assertSetEqual (*set1, set2, msg=None*)

A set-specific equality assertion.

Parameters

- **set1** – The first set to compare.
- **set2** – The second set to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

`assertSetEqual` uses ducktyping to support different types of sets, and is optimized for sets specifically (parameters must support a difference method).

assertTrue (*expr, msg=None*)

Check that the expression is true.

assertTupleEqual (*tuple1, tuple2, msg=None*)

A tuple-specific equality assertion.

Parameters

- **tuple1** – The first tuple to compare.
- **tuple2** – The second tuple to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assert_ (*expr, msg=None*)

Check that the expression is true.

base_test_pass (*name, environment, network, **kwargs*)

Basic test loop, requires an Agent to achieve a certain performance on an environment.

Parameters

- **name** (*str*) – The name of the test.
- **environment** (*Environment*) – The Environment object to use for the test.
- **network** (*LayerBasedNetwork*) – The Network to use for the agent’s model.
- **kwargs** (*any*) – Agent arguments.

base_test_run (*name, environment, network, **kwargs*)

Run test, tests whether algorithm can run and update without compilation errors, not whether it passes.

Parameters

- **name** (*str*) – The name of the test.
- **environment** (*Environment*) – The Environment object to use for the test.
- **network** (*LayerBasedNetwork*) – The Network to use for the agent’s model.
- **kwargs** (*any*) – Agent arguments.

config = {'update_mode': {'frequency': 4, 'batch_size': 4, 'unit': 'episodes'}, 'o

countTestCases ()

debug ()

Run the test without collecting errors in a TestResult

defaultTestResult ()

doCleanups ()

Execute all cleanup functions. Normally called for you after tearDown.

exclude_bool = False

exclude_bounded = False

exclude_float = False

exclude_int = False

exclude_lstm = False

exclude_multi = False

fail (*msg=None*)

Fail immediately, with the given message.

failIf (**args, **kwargs*)

failIfAlmostEqual (**args, **kwargs*)

failIfEqual (**args, **kwargs*)

failUnless (**args, **kwargs*)

failUnlessAlmostEqual (**args, **kwargs*)

failUnlessEqual (**args, **kwargs*)

failUnlessRaises (**args, **kwargs*)

failureException

alias of AssertionError

id()

longMessage = False

maxDiff = 640

multi_config = None

pass_threshold = 0.8

pre_run (*agent, environment*)
Called before `Runner.run`.

requires_network = True

run (*result=None*)

setUp()
Hook method for setting up the test fixture before exercising it.

setUpClass()
Hook method for setting up class fixture before running tests in the class.

shortDescription()
Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

skipTest (*reason*)
Skip this test.

tearDown()
Hook method for deconstructing the test fixture after testing it.

tearDownClass()
Hook method for deconstructing the class fixture after running all tests in the class.

test_bool()
Tests the case of one boolean action.

test_bounded_float()
Tests the case of one bounded float action, i.e. with min and max value.

test_float()
Tests the case of one float action.

test_int()
Tests the case of one integer action.

test_lstm()
Tests the case of using internal states via an LSTM layer (for one integer action).

test_multi()
Tests the case of multiple actions of different type and shape.

tensorforce.tests.test_vpg_baselines module

class tensorforce.tests.test_vpg_baselines.**TestVPGBaselines** (*methodName='runTest'*)
Bases: `tensorforce.tests.base_test.BaseTest`, `unittest.case.TestCase`

__init__ (*methodName='runTest'*)
Create an instance of the class that will use the named test method when executed. Raises a `ValueError` if the instance does not have a method with the specified name.

addCleanup (*function, *args, **kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after `tearDown` on test failure or success.

Cleanup items are called even if `setUp` fails (unlike `tearDown`).

addTypeEqualityFunc (*typeobj, function*)

Add a type specific `assertEqual` style function to compare a type.

This method is for use by `TestCase` subclasses that need to register their own type equality functions to provide nicer error messages.

Parameters

- **typeobj** – The data type to call this function on when both values are of the same type in `assertEqual()`.
- **function** – The callable taking two arguments and an optional `msg=` argument that raises `self.failureException` with a useful error message when the two arguments are not equal.

agent

alias of `VPGAgent`

assertAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertAlmostEquals (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertDictContainsSubset (*expected, actual, msg=None*)

Checks whether `actual` is a superset of `expected`.

assertDictEqual (*d1, d2, msg=None*)

assertEqual (*first, second, msg=None*)

Fail if the two objects are unequal as determined by the `'=='` operator.

assertEquals (*first, second, msg=None*)

Fail if the two objects are unequal as determined by the `'=='` operator.

assertFalse (*expr, msg=None*)

Check that the expression is false.

assertGreater (*a, b, msg=None*)

Just like `self.assertTrue(a > b)`, but with a nicer default message.

assertGreaterEqual (*a, b, msg=None*)

Just like `self.assertTrue(a >= b)`, but with a nicer default message.

assertIn (*member, container, msg=None*)

Just like self.assertTrue(a in b), but with a nicer default message.

assertIs (*expr1, expr2, msg=None*)

Just like self.assertTrue(a is b), but with a nicer default message.

assertIsInstance (*obj, cls, msg=None*)

Same as self.assertTrue(isinstance(obj, cls)), with a nicer default message.

assertIsNone (*obj, msg=None*)

Same as self.assertTrue(obj is None), with a nicer default message.

assertIsNot (*expr1, expr2, msg=None*)

Just like self.assertTrue(a is not b), but with a nicer default message.

assertIsNotNone (*obj, msg=None*)

Included for symmetry with assertIsNone.

assertItemsEqual (*expected_seq, actual_seq, msg=None*)

An unordered sequence specific comparison. It asserts that actual_seq and expected_seq have the same element counts. Equivalent to:

Asserts that each element has the same count in both sequences. .. rubric:: Example

- [0, 1, 1] and [1, 0, 1] compare equal.
- [0, 0, 1] and [0, 1] compare unequal.

assertLess (*a, b, msg=None*)

Just like self.assertTrue(a < b), but with a nicer default message.

assertLessEqual (*a, b, msg=None*)

Just like self.assertTrue(a <= b), but with a nicer default message.

assertListEqual (*list1, list2, msg=None*)

A list-specific equality assertion.

Parameters

- **list1** – The first list to compare.
- **list2** – The second list to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assertMultiLineEqual (*first, second, msg=None*)

Assert that two multi-line strings are equal.

assertNotAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotAlmostEquals (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotEqual (*first, second, msg=None*)

Fail if the two objects are equal as determined by the '!=' operator.

assertNotEquals (*first, second, msg=None*)

Fail if the two objects are equal as determined by the '!=' operator.

assertNotIn (*member, container, msg=None*)

Just like self.assertTrue(a not in b), but with a nicer default message.

assertNotIsInstance (*obj, cls, msg=None*)

Included for symmetry with assertIsInstance.

assertNotRegexpMatches (*text, unexpected_regexp, msg=None*)

Fail the test if the text matches the regular expression.

assertRaises (*excClass, callableObj=None, *args, **kwargs*)

Fail unless an exception of class excClass is raised by callableObj when invoked with arguments args and keyword arguments kwargs. If a different type of exception is raised, it will not be caught, and the test case will be deemed to have suffered an error, exactly as for an unexpected exception.

If called with callableObj omitted or None, will return a context object used like this:

The context manager keeps a reference to the exception as the 'exception' attribute. This allows you to inspect the exception after the assertion:

assertRaisesRegexp (*expected_exception, expected_regexp, callable_obj=None, *args, **kwargs*)

Asserts that the message in a raised exception matches a regexp.

Parameters

- **expected_exception** – Exception class expected to be raised.
- **expected_regexp** – Regexp (re pattern object or string) expected to be found in error message.
- **callable_obj** – Function to be called.
- **args** – Extra args.
- **kwargs** – Extra kwargs.

assertRegexpMatches (*text, expected_regexp, msg=None*)

Fail the test unless the text matches the regular expression.

assertSequenceEqual (*seq1, seq2, msg=None, seq_type=None*)

An equality assertion for ordered sequences (like lists and tuples).

For the purposes of this function, a valid ordered sequence type is one which can be indexed, has a length, and has an equality operator.

Parameters

- **seq1** – The first sequence to compare.
- **seq2** – The second sequence to compare.
- **seq_type** – The expected datatype of the sequences, or None if no datatype should be enforced.
- **msg** – Optional message to use on failure instead of a list of differences.

assertSetEqual (*set1, set2, msg=None*)

A set-specific equality assertion.

Parameters

- **set1** – The first set to compare.
- **set2** – The second set to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

`assertSetEqual` uses ducktyping to support different types of sets, and is optimized for sets specifically (parameters must support a difference method).

assertTrue (*expr*, *msg=None*)

Check that the expression is true.

assertTupleEqual (*tuple1*, *tuple2*, *msg=None*)

A tuple-specific equality assertion.

Parameters

- **tuple1** – The first tuple to compare.
- **tuple2** – The second tuple to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assert_ (*expr*, *msg=None*)

Check that the expression is true.

base_test_pass (*name*, *environment*, *network*, ***kwargs*)

Basic test loop, requires an Agent to achieve a certain performance on an environment.

Parameters

- **name** (*str*) – The name of the test.
- **environment** (`Environment`) – The Environment object to use for the test.
- **network** (`LayerBasedNetwork`) – The Network to use for the agent’s model.
- **kwargs** (*any*) – Agent arguments.

base_test_run (*name*, *environment*, *network*, ***kwargs*)

Run test, tests whether algorithm can run and update without compilation errors, not whether it passes.

Parameters

- **name** (*str*) – The name of the test.
- **environment** (`Environment`) – The Environment object to use for the test.
- **network** (`LayerBasedNetwork`) – The Network to use for the agent’s model.
- **kwargs** (*any*) – Agent arguments.

countTestCases ()

debug ()

Run the test without collecting errors in a `TestResult`

defaultTestResult ()

doCleanups ()

Execute all cleanup functions. Normally called for you after `tearDown`.

fail (*msg=None*)

Fail immediately, with the given message.

failIf (**args*, ***kwargs*)

```

failIfAlmostEqual (*args, **kwargs)
failIfEqual (*args, **kwargs)
failUnless (*args, **kwargs)
failUnlessAlmostEqual (*args, **kwargs)
failUnlessEqual (*args, **kwargs)
failUnlessRaises (*args, **kwargs)
failureException
    alias of AssertionError
id()
longMessage = False
maxDiff = 640
pass_threshold = 0.8
pre_run (agent, environment)
    Called before Runner.run.
requires_network = True
run (result=None)
setUp()
    Hook method for setting up the test fixture before exercising it.
setUpClass()
    Hook method for setting up class fixture before running tests in the class.
shortDescription()
    Returns a one-line description of the test, or None if no description has been provided.

    The default implementation of this method returns the first line of the specified test method's docstring.
skipTest (reason)
    Skip this test.
tearDown()
    Hook method for deconstructing the test fixture after testing it.
tearDownClass()
    Hook method for deconstructing the class fixture after running all tests in the class.
test_baseline_no_optimizer()
test_gae_baseline()
test_multi_baseline()
test_network_baseline()
test_states_baseline()

```

tensorforce.tests.test_vpg_optimizers module

```

class tensorforce.tests.test_vpg_optimizers.TestVPGOptimizers (methodName='runTest')
    Bases: tensorforce.tests.base\_test.BaseTest, unittest.case.TestCase

```

__init__ (*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

addCleanup (*function, *args, **kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after `tearDown` on test failure or success.

Cleanup items are called even if `setUp` fails (unlike `tearDown`).

addTypeEqualityFunc (*typeobj, function*)

Add a type specific `assertEqual` style function to compare a type.

This method is for use by `TestCase` subclasses that need to register their own type equality functions to provide nicer error messages.

Parameters

- **typeobj** – The data type to call this function on when both values are of the same type in `assertEqual()`.
- **function** – The callable taking two arguments and an optional `msg=` argument that raises `self.failureException` with a useful error message when the two arguments are not equal.

agent

alias of `VPGAgent`

assertAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertAlmostEquals (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertDictContainsSubset (*expected, actual, msg=None*)

Checks whether `actual` is a superset of `expected`.

assertDictEqual (*d1, d2, msg=None*)

assertEqual (*first, second, msg=None*)

Fail if the two objects are unequal as determined by the `'=='` operator.

assertEquals (*first, second, msg=None*)

Fail if the two objects are unequal as determined by the `'=='` operator.

assertFalse (*expr, msg=None*)

Check that the expression is false.

assertGreater (*a, b, msg=None*)

Just like `self.assertTrue(a > b)`, but with a nicer default message.

assertGreaterEqual (*a, b, msg=None*)

Just like `self.assertTrue(a >= b)`, but with a nicer default message.

assertIn (*member, container, msg=None*)

Just like `self.assertTrue(a in b)`, but with a nicer default message.

assertIs (*expr1, expr2, msg=None*)

Just like `self.assertTrue(a is b)`, but with a nicer default message.

assertIsInstance (*obj, cls, msg=None*)

Same as `self.assertTrue(isinstance(obj, cls))`, with a nicer default message.

assertIsNone (*obj, msg=None*)

Same as `self.assertTrue(obj is None)`, with a nicer default message.

assertIsNot (*expr1, expr2, msg=None*)

Just like `self.assertTrue(a is not b)`, but with a nicer default message.

assertIsNotNone (*obj, msg=None*)

Included for symmetry with `assertIsNone`.

assertItemsEqual (*expected_seq, actual_seq, msg=None*)

An unordered sequence specific comparison. It asserts that `actual_seq` and `expected_seq` have the same element counts. Equivalent to:

Asserts that each element has the same count in both sequences. .. rubric:: Example

- `[0, 1, 1]` and `[1, 0, 1]` compare equal.
- `[0, 0, 1]` and `[0, 1]` compare unequal.

assertLess (*a, b, msg=None*)

Just like `self.assertTrue(a < b)`, but with a nicer default message.

assertLessEqual (*a, b, msg=None*)

Just like `self.assertTrue(a <= b)`, but with a nicer default message.

assertListEqual (*list1, list2, msg=None*)

A list-specific equality assertion.

Parameters

- **list1** – The first list to compare.
- **list2** – The second list to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assertMultiLineEqual (*first, second, msg=None*)

Assert that two multi-line strings are equal.

assertNotAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotAlmostEquals (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotEqual (*first, second, msg=None*)

Fail if the two objects are equal as determined by the '!=' operator.

assertNotEquals (*first, second, msg=None*)

Fail if the two objects are equal as determined by the '!=' operator.

assertNotIn (*member, container, msg=None*)

Just like self.assertTrue(a not in b), but with a nicer default message.

assertNotIsInstance (*obj, cls, msg=None*)

Included for symmetry with assertIsInstance.

assertNotRegexpMatches (*text, unexpected_regexp, msg=None*)

Fail the test if the text matches the regular expression.

assertRaises (*excClass, callableObj=None, *args, **kwargs*)

Fail unless an exception of class excClass is raised by callableObj when invoked with arguments args and keyword arguments kwargs. If a different type of exception is raised, it will not be caught, and the test case will be deemed to have suffered an error, exactly as for an unexpected exception.

If called with callableObj omitted or None, will return a context object used like this:

The context manager keeps a reference to the exception as the 'exception' attribute. This allows you to inspect the exception after the assertion:

assertRaisesRegexp (*expected_exception, expected_regexp, callable_obj=None, *args, **kwargs*)

Asserts that the message in a raised exception matches a regexp.

Parameters

- **expected_exception** – Exception class expected to be raised.
- **expected_regexp** – Regexp (re pattern object or string) expected to be found in error message.
- **callable_obj** – Function to be called.
- **args** – Extra args.
- **kwargs** – Extra kwargs.

assertRegexpMatches (*text, expected_regexp, msg=None*)

Fail the test unless the text matches the regular expression.

assertSequenceEqual (*seq1, seq2, msg=None, seq_type=None*)

An equality assertion for ordered sequences (like lists and tuples).

For the purposes of this function, a valid ordered sequence type is one which can be indexed, has a length, and has an equality operator.

Parameters

- **seq1** – The first sequence to compare.
- **seq2** – The second sequence to compare.
- **seq_type** – The expected datatype of the sequences, or None if no datatype should be enforced.
- **msg** – Optional message to use on failure instead of a list of differences.

assertSetEqual (*set1, set2, msg=None*)

A set-specific equality assertion.

Parameters

- **set1** – The first set to compare.
- **set2** – The second set to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

`assertSetEqual` uses ducktyping to support different types of sets, and is optimized for sets specifically (parameters must support a difference method).

assertTrue (*expr, msg=None*)

Check that the expression is true.

assertTupleEqual (*tuple1, tuple2, msg=None*)

A tuple-specific equality assertion.

Parameters

- **tuple1** – The first tuple to compare.
- **tuple2** – The second tuple to compare.
- **msg** – Optional message to use on failure instead of a list of differences.

assert_ (*expr, msg=None*)

Check that the expression is true.

base_test_pass (*name, environment, network, **kwargs*)

Basic test loop, requires an Agent to achieve a certain performance on an environment.

Parameters

- **name** (*str*) – The name of the test.
- **environment** (`Environment`) – The Environment object to use for the test.
- **network** (`LayerBasedNetwork`) – The Network to use for the agent’s model.
- **kwargs** (*any*) – Agent arguments.

base_test_run (*name, environment, network, **kwargs*)

Run test, tests whether algorithm can run and update without compilation errors, not whether it passes.

Parameters

- **name** (*str*) – The name of the test.
- **environment** (`Environment`) – The Environment object to use for the test.
- **network** (`LayerBasedNetwork`) – The Network to use for the agent’s model.
- **kwargs** (*any*) – Agent arguments.

countTestCases ()

debug ()

Run the test without collecting errors in a `TestResult`

defaultTestResult ()

doCleanups ()

Execute all cleanup functions. Normally called for you after `tearDown`.

fail (*msg=None*)
Fail immediately, with the given message.

failIf (**args, **kwargs*)

failIfAlmostEqual (**args, **kwargs*)

failIfEqual (**args, **kwargs*)

failUnless (**args, **kwargs*)

failUnlessAlmostEqual (**args, **kwargs*)

failUnlessEqual (**args, **kwargs*)

failUnlessRaises (**args, **kwargs*)

failureException
alias of `AssertionError`

id ()

longMessage = **False**

maxDiff = **640**

pass_threshold = **0.8**

pre_run (*agent, environment*)
Called before `Runner.run`.

requires_network = **True**

run (*result=None*)

setUp ()
Hook method for setting up the test fixture before exercising it.

setUpClass ()
Hook method for setting up class fixture before running tests in the class.

shortDescription ()
Returns a one-line description of the test, or `None` if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

skipTest (*reason*)
Skip this test.

tearDown ()
Hook method for deconstructing the test fixture after testing it.

tearDownClass ()
Hook method for deconstructing the class fixture after running all tests in the class.

test_adam ()

test_clipped_step ()

test_evolutionary ()

test_multi_step ()

test_natural_gradient ()

test_optimized_step ()

test_subsampling_step ()

Module contents

1.6.2 Submodules

1.6.3 `tensorforce.exception` module

exception `tensorforce.exception.TensorForceError`

Bases: `exceptions.Exception`

TensorForce error

__init__

`x.init(...)` initializes x; see `help(type(x))` for signature

args

message

1.6.4 `tensorforce.meta_parameter_recorder` module

class `tensorforce.meta_parameter_recorder.MetaParameterRecorder` (*current_frame*)

Bases: `object`

Class to record MetaParameters as well as Summary/Description for TensorBoard (TEXT & FILE will come later).

General:

- **format_type**: used to configure data conversion for TensorBoard=0, TEXT & JSON (not Implemented), etc

__init__ (*current_frame*)

Init the MetaParameterRecord with “Agent” parameters by passing `inspect.currentframe()` from Agent Class.

The Init will search back to find the parent class to capture all passed parameters and store them in “`self.meta_params`”.

NOTE: Currently only optimized for TensorBoard output.

TODO: Add JSON Export, TEXT EXPORT

Parameters **current_frame** – Frame value from class to obtain metaparameters[= `inspect.currentframe()`]

build_metagraph_list ()

Convert MetaParams into TF Summary Format and create summary_op.

Returns Merged TF Op for TEXT summary elements, should only be executed once to reduce data duplication.

convert_data_to_string (*data*, *indent=0*, *format_type=0*, *separator=None*, *eol=None*)

convert_dictionary_to_string (*data*, *indent=0*, *format_type=0*, *separator=None*, *eol=None*)

convert_list_to_string (*data*, *indent=0*, *format_type=0*, *eol=None*, *count=True*)

convert_ndarray_to_md (*data*, *format_type=0*, *eol=None*)

merge_custom (*custom_dict*)

text_output (*format_type=1*)

1.6.5 tensorflow.util module

class tensorflow.util.SavableComponent

Bases: object

Component that can save and restore its own state.

__init__

x.init(...) initializes x; see help(type(x)) for signature

get_savable_variables()

Returns the list of all the variables this component is responsible to save and restore.

Returns The list of variables that will be saved or restored.

register_saver_ops()

Registers the saver operations to the graph in context.

restore(sess, save_path)

Restores the values of the managed variables from disk location.

Parameters

- **sess** – The session for which to save the managed variables.
- **save_path** – The path used to save the data to.

save(sess, save_path, timestep=None)

Saves this component's managed variables.

Parameters

- **sess** – The session for which to save the managed variables.
- **save_path** – The path to save data to.
- **timestep** – Optional, the timestep to append to the file name.

Returns Checkpoint path where the model was saved.

tensorflow.util.get_object(obj, predefined_objects=None, default_object=None, kwargs=None)

Utility method to map some kind of object specification to its content, e.g. optimizer or baseline specifications to the respective classes.

Parameters

- **obj** – A specification dict (value for key 'type' optionally specifies the object, options as follows), a module path (e.g., my_module.MyClass), a key in predefined_objects, or a callable (e.g., the class type object).
- **predefined_objects** – Dict containing predefined set of objects, accessible via their key
- **default_object** – Default object is no other is specified
- **kwargs** – Arguments for object creation

Returns: The retrieved object

tensorflow.util.map_tensors(fn, tensors)

tensorflow.util.numpy_dtype(dtype)

Translates dtype specifications in configurations to numpy data types. :param dtype: String describing a numerical type (e.g. 'float') or numerical type primitive.

Returns: Numpy data type

`tensorforce.util.prepare_kwargs (raw, string_parameter='name')`

Utility method to convert raw string/diction input into a dictionary to pass into a function. Always returns a dictionary.

Parameters **raw** – string or dictionary, string is assumed to be the name of the activation activation function. Dictionary will be passed through unchanged.

Returns: kwargs dictionary for ****kwargs**

`tensorforce.util.prod (xs)`

Computes the product along the elements in an iterable. Returns 1 for empty iterable.

Parameters **xs** – Iterable containing numbers.

Returns: Product along iterable.

`tensorforce.util.rank (x)`

`tensorforce.util.shape (x, unknown=-1)`

`tensorforce.util.strip_name_scope (name, base_scope)`

`tensorforce.util.tf_dtype (dtype)`

Translates dtype specifications in configurations to tensorflow data types.

Parameters **dtype** – String describing a numerical type (e.g. 'float'), numpy data type, or numerical type primitive.

Returns: TensorFlow data type

1.6.6 Module contents

exception `tensorforce.TensorForceError`

Bases: `exceptions.Exception`

TensorForce error

__init__

`x.init(...)` initializes x; see `help(type(x))` for signature

args

message

CHAPTER 2

More information

You can find more information at our [TensorForce GitHub repository](#).

We have a separate repository available for benchmarking our algorithm implementations [here](<https://github.com/reinforceio/tensorforce-benchmark>).

t

tensorforce, 359

tensorforce.agents, 50

tensorforce.agents.agent, 26

tensorforce.agents.constant_agent, 27

tensorforce.agents.dqfd_agent, 30

tensorforce.agents.dqn_agent, 32

tensorforce.agents.dqn_nstep_agent, 34

tensorforce.agents.learning_agent, 37

tensorforce.agents.naf_agent, 39

tensorforce.agents.ppo_agent, 42

tensorforce.agents.random_agent, 44

tensorforce.agents.trpo_agent, 46

tensorforce.agents.vpg_agent, 48

tensorforce.contrib, 83

tensorforce.contrib.ale, 74

tensorforce.contrib.deepmind_lab, 75

tensorforce.contrib.maze_explorer, 76

tensorforce.contrib.openai_gym, 77

tensorforce.contrib.openai_universe, 78

tensorforce.contrib.remote_environment, 79

tensorforce.contrib.state_settable_environment, 80

tensorforce.contrib.unreal_engine, 81

tensorforce.core, 181

tensorforce.core.baselines, 88

tensorforce.core.baselines.aggregated_baseline, 83

tensorforce.core.baselines.baseline, 84

tensorforce.core.baselines.cnn_baseline, 85

tensorforce.core.baselines.mlp_baseline, 86

tensorforce.core.baselines.network_baseline, 87

tensorforce.core.distributions, 96

tensorforce.core.distributions.bernoulli, 92

tensorforce.core.distributions.beta, 93

tensorforce.core.distributions.categorical, 94

tensorforce.core.distributions.distribution, 94

tensorforce.core.distributions.gaussian, 96

tensorforce.core.explorations, 102

tensorforce.core.explorations.constant, 100

tensorforce.core.explorations.epsilon_anneal, 100

tensorforce.core.explorations.epsilon_decay, 101

tensorforce.core.explorations.exploration, 101

tensorforce.core.explorations.ornstein_uhlenbeck_process, 102

tensorforce.core.memories, 108

tensorforce.core.memories.memory, 104

tensorforce.core.memories.prioritized_replay, 106

tensorforce.core.memories.replay, 107

tensorforce.core.networks, 129

tensorforce.core.networks.complex_network, 114

tensorforce.core.networks.layer, 116

tensorforce.core.networks.network, 127

tensorforce.core.optimizers, 166

tensorforce.core.optimizers.clipped_step, 151

tensorforce.core.optimizers.evolutionary, 153

tensorforce.core.optimizers.global_optimizer, 154

tensorforce.core.optimizers.meta_optimizer, 156

tensorforce.core.optimizers.multi_step, 157

tensorforce.core.optimizers.natural_gradient,

158
tensorforce.core.optimizers.optimized_step, 317
160
tensorforce.core.optimizers.optimizer, 326
162
tensorforce.core.optimizers.solvers, 147
tensorforce.core.optimizers.solvers.conjugate_gradient, 326
142
tensorforce.core.optimizers.solvers.iterative, 341
144
tensorforce.core.optimizers.solvers.linestaircase, 346
145
tensorforce.core.optimizers.solvers.solver, 351
147
tensorforce.core.optimizers.synchronization,
163
tensorforce.core.optimizers.tf_optimizer,
164
tensorforce.environments, 182
tensorforce.environments.environment,
181
tensorforce.exception, 357
tensorforce.execution, 186
tensorforce.execution.runner, 183
tensorforce.execution.threaded_runner,
184
tensorforce.meta_parameter_recorder, 357
tensorforce.models, 234
tensorforce.models.constant_model, 188
tensorforce.models.distribution_model,
191
tensorforce.models.model, 196
tensorforce.models.pg_log_prob_model,
200
tensorforce.models.pg_model, 205
tensorforce.models.pg_prob_ratio_model,
210
tensorforce.models.q_demo_model, 214
tensorforce.models.q_model, 219
tensorforce.models.q_naf_model, 223
tensorforce.models.q_nstep_model, 227
tensorforce.models.random_model, 231
tensorforce.tests, 357
tensorforce.tests.base_agent_test, 282
tensorforce.tests.base_test, 283
tensorforce.tests.test_constant_agent,
284
tensorforce.tests.test_dqfd_agent, 289
tensorforce.tests.test_dqn_agent, 295
tensorforce.tests.test_dqn_nstep_agent,
300
tensorforce.tests.test_naf_agent, 306
tensorforce.tests.test_ppo_agent, 311
tensorforce.tests.test_quickstart_example,

Symbols

- `__init__` (tensorforce.TensorForceError attribute), 359
- `__init__` (tensorforce.contrib.state_settable_environment.StateSettableEnvironment attribute), 80
- `__init__` (tensorforce.environments.Environment attribute), 182
- `__init__` (tensorforce.environments.environment.Environment attribute), 181
- `__init__` (tensorforce.exception.TensorForceError attribute), 357
- `__init__` (tensorforce.tests.base_agent_test.BaseAgentTest attribute), 282
- `__init__` (tensorforce.tests.base_test.BaseTest attribute), 283
- `__init__` (tensorforce.util.SavableComponent attribute), 358
- `__init__()` (tensorforce.agents.Agent method), 50
- `__init__()` (tensorforce.agents.ConstantAgent method), 52
- `__init__()` (tensorforce.agents.DDPGAgent method), 72
- `__init__()` (tensorforce.agents.DQFDAgent method), 58
- `__init__()` (tensorforce.agents.DQNAgent method), 60
- `__init__()` (tensorforce.agents.DQNNstepAgent method), 62
- `__init__()` (tensorforce.agents.LearningAgent method), 55
- `__init__()` (tensorforce.agents.NAFAgent method), 64
- `__init__()` (tensorforce.agents.PPOAgent method), 66
- `__init__()` (tensorforce.agents.RandomAgent method), 53
- `__init__()` (tensorforce.agents.TRPOAgent method), 68
- `__init__()` (tensorforce.agents.VPGAgent method), 70
- `__init__()` (tensorforce.agents.agent.Agent method), 26
- `__init__()` (tensorforce.agents.constant_agent.ConstantAgent method), 28
- `__init__()` (tensorforce.agents.dqfd_agent.DQFDAgent method), 30
- `__init__()` (tensorforce.agents.dqn_agent.DQNAgent method), 32
- `__init__()` (tensorforce.agents.dqn_nstep_agent.DQNNstepAgent method), 35
- `__init__()` (tensorforce.agents.learning_agent.LearningAgent method), 37
- `__init__()` (tensorforce.agents.naf_agent.NAFAgent method), 39
- `__init__()` (tensorforce.agents.ppo_agent.PPOAgent method), 42
- `__init__()` (tensorforce.agents.random_agent.RandomAgent method), 44
- `__init__()` (tensorforce.agents.trpo_agent.TRPOAgent method), 46
- `__init__()` (tensorforce.agents.vpg_agent.VPGAgent method), 48
- `__init__()` (tensorforce.contrib.ale.ALE method), 74
- `__init__()` (tensorforce.contrib.deepmind_lab.DeepMindLab method), 75
- `__init__()` (tensorforce.contrib.maze_explorer.MazeExplorer method), 76
- `__init__()` (tensorforce.contrib.openai_gym.OpenAIGym method), 77
- `__init__()` (tensorforce.contrib.openai_universe.OpenAIUniverse method), 78
- `__init__()` (tensorforce.contrib.remote_environment.MsgPackNumpyProtocol method), 79
- `__init__()` (tensorforce.contrib.remote_environment.RemoteEnvironment method), 79
- `__init__()` (tensorforce.contrib.unreal_engine.UE4Environment method), 81
- `__init__()` (tensorforce.core.baselines.AggregatedBaseline method), 89
- `__init__()` (tensorforce.core.baselines.Baseline method), 88
- `__init__()` (tensorforce.core.baselines.CNNBaseline method), 91
- `__init__()` (tensorforce.core.baselines.MLPBaseline method), 90
- `__init__()` (tensorforce.core.baselines.NetworkBaseline method), 90
- `__init__()` (tensorforce.core.baselines.aggregated_baseline.AggregatedBaseline method), 83

<code>__init__()</code> (tensorflow.core.baselines.baseline.Baseline method), 84	<code>__init__()</code> (tensorflow.core.memories.PrioritizedReplay method), 112
<code>__init__()</code> (tensorflow.core.baselines.cnn_baseline.CNNBaseline method), 85	<code>__init__()</code> (tensorflow.core.memories.Queue method), 109
<code>__init__()</code> (tensorflow.core.baselines.mlp_baseline.MLPBaseline method), 86	<code>__init__()</code> (tensorflow.core.memories.Replay method), 111
<code>__init__()</code> (tensorflow.core.baselines.network_baseline.NetworkBaseline method), 87	<code>__init__()</code> (tensorflow.core.memories.memory.Memory method), 104
<code>__init__()</code> (tensorflow.core.distributions.Bernoulli method), 97	<code>__init__()</code> (tensorflow.core.memories.prioritized_replay.PrioritizedReplay method), 106
<code>__init__()</code> (tensorflow.core.distributions.Beta method), 99	<code>__init__()</code> (tensorflow.core.memories.replay.Replay method), 107
<code>__init__()</code> (tensorflow.core.distributions.Categorical method), 98	<code>__init__()</code> (tensorflow.core.networks.Conv1d method), 137
<code>__init__()</code> (tensorflow.core.distributions.Distribution method), 96	<code>__init__()</code> (tensorflow.core.networks.Conv2d method), 137
<code>__init__()</code> (tensorflow.core.distributions.Gaussian method), 99	<code>__init__()</code> (tensorflow.core.networks.Dense method), 135
<code>__init__()</code> (tensorflow.core.distributions.bernoulli.Bernoulli method), 92	<code>__init__()</code> (tensorflow.core.networks.Dropout method), 132
<code>__init__()</code> (tensorflow.core.distributions.beta.Beta method), 93	<code>__init__()</code> (tensorflow.core.networks.Dueling method), 136
<code>__init__()</code> (tensorflow.core.distributions.categorical.Categorical method), 94	<code>__init__()</code> (tensorflow.core.networks.Embedding method), 134
<code>__init__()</code> (tensorflow.core.distributions.distribution.Distribution method), 94	<code>__init__()</code> (tensorflow.core.networks.Flatten method), 132
<code>__init__()</code> (tensorflow.core.distributions.gaussian.Gaussian method), 96	<code>__init__()</code> (tensorflow.core.networks.InternalLstm method), 138
<code>__init__()</code> (tensorflow.core.explorations.Constant method), 102	<code>__init__()</code> (tensorflow.core.networks.Layer method), 129
<code>__init__()</code> (tensorflow.core.explorations.EpsilonAnneal method), 103	<code>__init__()</code> (tensorflow.core.networks.LayerBasedNetwork method), 141
<code>__init__()</code> (tensorflow.core.explorations.EpsilonDecay method), 103	<code>__init__()</code> (tensorflow.core.networks.LayeredNetwork method), 141
<code>__init__()</code> (tensorflow.core.explorations.Exploration method), 102	<code>__init__()</code> (tensorflow.core.networks.Linear method), 134
<code>__init__()</code> (tensorflow.core.explorations.GaussianNoise method), 103	<code>__init__()</code> (tensorflow.core.networks.Lstm method), 139
<code>__init__()</code> (tensorflow.core.explorations.OrnsteinUhlenbeckProcess method), 104	<code>__init__()</code> (tensorflow.core.networks.Network method), 140
<code>__init__()</code> (tensorflow.core.explorations.constant.Constant method), 100	<code>__init__()</code> (tensorflow.core.networks.Nonlinearity method), 131
<code>__init__()</code> (tensorflow.core.explorations.epsilon_anneal.EpsilonAnneal method), 100	<code>__init__()</code> (tensorflow.core.networks.Pool2d method), 133
<code>__init__()</code> (tensorflow.core.explorations.epsilon_decay.EpsilonDecay method), 101	<code>__init__()</code> (tensorflow.core.networks.TFLayer method), 130
<code>__init__()</code> (tensorflow.core.explorations.exploration.Exploration method), 101	<code>__init__()</code> (tensorflow.core.networks.complex_network.ComplexLayeredNetwork method), 114
<code>__init__()</code> (tensorflow.core.explorations.ornstein_uhlenbeck_ornstein_uhlenbeck.Process method), 102	<code>__init__()</code> (tensorflow.core.networks.complex_network.Input method), 115
<code>__init__()</code> (tensorflow.core.memories.Latest method), 110	<code>__init__()</code> (tensorflow.core.networks.complex_network.Output method), 115
<code>__init__()</code> (tensorflow.core.memories.Memory method), 108	<code>__init__()</code> (tensorflow.core.networks.layer.Conv1d method), 116
	<code>__init__()</code> (tensorflow.core.networks.layer.Conv2d method), 117

<code>__init__()</code> (tensorflow.core.networks.layer.Dense method), 118	<code>__init__()</code> (tensorflow.core.optimizers.evolutionary.Evolutionary method), 153
<code>__init__()</code> (tensorflow.core.networks.layer.Dropout method), 118	<code>__init__()</code> (tensorflow.core.optimizers.global_optimizer.GlobalOptimizer method), 154
<code>__init__()</code> (tensorflow.core.networks.layer.Dueling method), 119	<code>__init__()</code> (tensorflow.core.optimizers.meta_optimizer.MetaOptimizer method), 156
<code>__init__()</code> (tensorflow.core.networks.layer.Embedding method), 120	<code>__init__()</code> (tensorflow.core.optimizers.multi_step.MultiStep method), 157
<code>__init__()</code> (tensorflow.core.networks.layer.Flatten method), 121	<code>__init__()</code> (tensorflow.core.optimizers.natural_gradient.NaturalGradient method), 158
<code>__init__()</code> (tensorflow.core.networks.layer.InternalLstm method), 121	<code>__init__()</code> (tensorflow.core.optimizers.optimized_step.OptimizedStep method), 160
<code>__init__()</code> (tensorflow.core.networks.layer.Layer method), 122	<code>__init__()</code> (tensorflow.core.optimizers.optimizer.Optimizer method), 162
<code>__init__()</code> (tensorflow.core.networks.layer.Linear method), 123	<code>__init__()</code> (tensorflow.core.optimizers.solvers.ConjugateGradient method), 149
<code>__init__()</code> (tensorflow.core.networks.layer.Lstm method), 123	<code>__init__()</code> (tensorflow.core.optimizers.solvers.Iterative method), 147
<code>__init__()</code> (tensorflow.core.networks.layer.Nonlinearity method), 124	<code>__init__()</code> (tensorflow.core.optimizers.solvers.LineSearch method), 150
<code>__init__()</code> (tensorflow.core.networks.layer.Pool2d method), 125	<code>__init__()</code> (tensorflow.core.optimizers.solvers.Solver method), 147
<code>__init__()</code> (tensorflow.core.networks.layer.TFLayer method), 126	<code>__init__()</code> (tensorflow.core.optimizers.solvers.conjugate_gradient.ConjugateGradient method), 143
<code>__init__()</code> (tensorflow.core.networks.network.LayerBasedNetwork method), 127	<code>__init__()</code> (tensorflow.core.optimizers.solvers.iterative.Iterative method), 144
<code>__init__()</code> (tensorflow.core.networks.network.LayeredNetwork method), 127	<code>__init__()</code> (tensorflow.core.optimizers.solvers.line_search.LineSearch method), 145
<code>__init__()</code> (tensorflow.core.networks.network.Network method), 128	<code>__init__()</code> (tensorflow.core.optimizers.solvers.solver.Solver method), 147
<code>__init__()</code> (tensorflow.core.optimizers.ClippedStep method), 174	<code>__init__()</code> (tensorflow.core.optimizers.synchronization.Synchronization method), 163
<code>__init__()</code> (tensorflow.core.optimizers.Evolutionary method), 171	<code>__init__()</code> (tensorflow.core.optimizers.tf_optimizer.TFOptimizer method), 164
<code>__init__()</code> (tensorflow.core.optimizers.GlobalOptimizer method), 168	<code>__init__()</code> (tensorflow.environments.MinimalTest method), 183
<code>__init__()</code> (tensorflow.core.optimizers.MetaOptimizer method), 167	<code>__init__()</code> (tensorflow.execution.BaseRunner method), 186
<code>__init__()</code> (tensorflow.core.optimizers.MultiStep method), 175	<code>__init__()</code> (tensorflow.execution.Runner method), 187
<code>__init__()</code> (tensorflow.core.optimizers.NaturalGradient method), 172	<code>__init__()</code> (tensorflow.execution.ThreadedRunner method), 187
<code>__init__()</code> (tensorflow.core.optimizers.OptimizedStep method), 176	<code>__init__()</code> (tensorflow.execution.runner.Runner method), 184
<code>__init__()</code> (tensorflow.core.optimizers.Optimizer method), 166	<code>__init__()</code> (tensorflow.execution.threaded_runner.ThreadedRunner method), 184
<code>__init__()</code> (tensorflow.core.optimizers.SubsamplingStep method), 178	<code>__init__()</code> (tensorflow.meta_parameter_recorder.MetaParameterRecorder method), 357
<code>__init__()</code> (tensorflow.core.optimizers.Synchronization method), 179	<code>__init__()</code> (tensorflow.models.DPGTargetModel method), 257
<code>__init__()</code> (tensorflow.core.optimizers.TFOptimizer method), 169	<code>__init__()</code> (tensorflow.models.DistributionModel method), 244
<code>__init__()</code> (tensorflow.core.optimizers.clipped_step.ClippedStep method), 152	<code>__init__()</code> (tensorflow.models.MemoryModel method), 238
	<code>__init__()</code> (tensorflow.models.Model method), 234

`__init__()` (tensorforce.models.PGLogProbModel method), 261

`__init__()` (tensorforce.models.PGModel method), 248

`__init__()` (tensorforce.models.PGProbRatioModel method), 253

`__init__()` (tensorforce.models.QDemoModel method), 278

`__init__()` (tensorforce.models.QModel method), 265

`__init__()` (tensorforce.models.QNAFModel method), 273

`__init__()` (tensorforce.models.QNstepModel method), 269

`__init__()` (tensorforce.models.constant_model.ConstantModel method), 189

`__init__()` (tensorforce.models.distribution_model.DistributionModel method), 192

`__init__()` (tensorforce.models.model.Model method), 197

`__init__()` (tensorforce.models.pg_log_prob_model.PGLogProbModel method), 201

`__init__()` (tensorforce.models.pg_model.PGModel method), 205

`__init__()` (tensorforce.models.pg_prob_ratio_model.PGProbRatioModel method), 210

`__init__()` (tensorforce.models.q_demo_model.QDemoModel method), 214

`__init__()` (tensorforce.models.q_model.QModel method), 219

`__init__()` (tensorforce.models.q_naf_model.QNAFModel method), 223

`__init__()` (tensorforce.models.q_nstep_model.QNstepModel method), 228

`__init__()` (tensorforce.models.random_model.RandomModel method), 231

`__init__()` (tensorforce.tests.test_constant_agent.TestConstantAgent method), 284

`__init__()` (tensorforce.tests.test_dqfd_agent.TestDQFDAgent method), 289

`__init__()` (tensorforce.tests.test_dqn_agent.TestDQNAgent method), 295

`__init__()` (tensorforce.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 300

`__init__()` (tensorforce.tests.test_naf_agent.TestNAFAgent method), 306

`__init__()` (tensorforce.tests.test_ppo_agent.TestPPOAgent method), 311

`__init__()` (tensorforce.tests.test_quickstart_example.TestQuickstartExample method), 317

`__init__()` (tensorforce.tests.test_random_agent.TestRandomAgent method), 321

`__init__()` (tensorforce.tests.test_reward_estimation.TestRewardEstimation method), 326

`__init__()` (tensorforce.tests.test_trpo_agent.TestTRPOAgent method), 331

`__init__()` (tensorforce.tests.test_tutorial_code.TestTutorialCode method), 336

`__init__()` (tensorforce.tests.test_tutorial_code.TestTutorialCode.MyClient method), 336

`__init__()` (tensorforce.tests.test_vpg_agent.TestVPGAgent method), 341

`__init__()` (tensorforce.tests.test_vpg_baselines.TestVPGBaselines method), 346

`__init__()` (tensorforce.tests.test_vpg_optimizers.TestVPGOptimizers method), 351

A

`act()` (tensorforce.agents.Agent method), 50

`act()` (tensorforce.agents.agent.Agent method), 26

`act()` (tensorforce.agents.constant_agent.ConstantAgent method), 28

`act()` (tensorforce.agents.ConstantAgent method), 52

`act()` (tensorforce.agents.DDPGAgent method), 73

`act()` (tensorforce.agents.dqfd_agent.DQFDAgent method), 31

`act()` (tensorforce.agents.DQFDAgent method), 58

`act()` (tensorforce.agents.dqn_agent.DQNAgent method), 48

`act()` (tensorforce.agents.dqn_nstep_agent.DQNNstepAgent method), 35

`act()` (tensorforce.agents.DQNAgent method), 60

`act()` (tensorforce.agents.DQNNstepAgent method), 62

`act()` (tensorforce.agents.learning_agent.LearningAgent method), 38

`act()` (tensorforce.agents.LearningAgent method), 56

`act()` (tensorforce.agents.naf_agent.NAFAgent method), 40

`act()` (tensorforce.agents.NAFAgent method), 64

`act()` (tensorforce.agents.ppo_agent.PPOAgent method), 43

`act()` (tensorforce.agents.PPOAgent method), 66

`act()` (tensorforce.agents.random_agent.RandomAgent method), 44

`act()` (tensorforce.agents.RandomAgent method), 54

`act()` (tensorforce.agents.trpo_agent.TRPOAgent method), 47

`act()` (tensorforce.agents.TRPOAgent method), 69

`act()` (tensorforce.agents.vpg_agent.VPGAgent method), 49

`act()` (tensorforce.agents.VPGAgent method), 71

`act()` (tensorforce.models.constant_model.ConstantModel method), 189

`act()` (tensorforce.models.distribution_model.DistributionModel method), 192

`act()` (tensorforce.models.DistributionModel method), 192

`act()` (tensorforce.models.DPGTargetModel method), 257

`act()` (tensorforce.models.MemoryModel method), 238

`act()` (tensorforce.models.Model method), 235

- act() (tensorflow.models.model.Model method), 197
- act() (tensorflow.models.pg_log_prob_model.PGLogProbModel method), 201
- act() (tensorflow.models.pg_model.PGModel method), 205
- act() (tensorflow.models.pg_prob_ratio_model.PGProbRatioModel method), 128
- act() (tensorflow.models.PGLogProbModel method), 261
- act() (tensorflow.models.PGModel method), 248
- act() (tensorflow.models.PGProbRatioModel method), 253
- act() (tensorflow.models.q_demo_model.QDemoModel method), 215
- act() (tensorflow.models.q_model.QModel method), 219
- act() (tensorflow.models.q_naf_model.QNAFModel method), 223
- act() (tensorflow.models.q_nstep_model.QNstepModel method), 228
- act() (tensorflow.models.QDemoModel method), 278
- act() (tensorflow.models.QModel method), 265
- act() (tensorflow.models.QNAFModel method), 274
- act() (tensorflow.models.QNstepModel method), 269
- act() (tensorflow.models.random_model.RandomModel method), 232
- action_from_space() (tensorflow.contrib.openai_gym.OpenAIGym static method), 77
- action_names (tensorflow.contrib.ale.ALE attribute), 74
- actions (tensorflow.contrib.ale.ALE attribute), 74
- actions (tensorflow.contrib.deepmind_lab.DeepMindLab attribute), 76
- actions (tensorflow.contrib.maze_explorer.MazeExplorer attribute), 77
- actions (tensorflow.contrib.openai_gym.OpenAIGym attribute), 77
- actions (tensorflow.contrib.openai_universe.OpenAIUniverse attribute), 78
- actions (tensorflow.contrib.remote_environment.RemoteEnvironment attribute), 79
- actions (tensorflow.contrib.state_settable_environment.StateSettableEnvironment attribute), 80
- actions (tensorflow.environments.Environment attribute), 182
- actions (tensorflow.environments.environment.Environment attribute), 181
- actions (tensorflow.environments.MinimalTest attribute), 183
- actions() (tensorflow.contrib.unreal_engine.UE4Environment method), 82
- add_layer() (tensorflow.core.networks.complex_network.ComplexLayeredNetwork method), 114
- add_layer() (tensorflow.core.networks.LayerBasedNetwork method), 141
- add_layer() (tensorflow.core.networks.LayeredNetwork method), 141
- add_layer() (tensorflow.core.networks.network.LayerBasedNetwork method), 127
- add_layer() (tensorflow.core.networks.network.LayeredNetwork method), 128
- addCleanup() (tensorflow.tests.test_constant_agent.TestConstantAgent method), 284
- addCleanup() (tensorflow.tests.test_dqfd_agent.TestDQFDAGent method), 289
- addCleanup() (tensorflow.tests.test_dqn_agent.TestDQNAgent method), 295
- addCleanup() (tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 300
- addCleanup() (tensorflow.tests.test_naf_agent.TestNAFAGent method), 306
- addCleanup() (tensorflow.tests.test_ppo_agent.TestPPOAgent method), 311
- addCleanup() (tensorflow.tests.test_quickstart_example.TestQuickstartExample method), 317
- addCleanup() (tensorflow.tests.test_random_agent.TestRandomAgent method), 321
- addCleanup() (tensorflow.tests.test_reward_estimation.TestRewardEstimation method), 327
- addCleanup() (tensorflow.tests.test_trpo_agent.TestTRPOAgent method), 331
- addCleanup() (tensorflow.tests.test_tutorial_code.TestTutorialCode method), 337
- addCleanup() (tensorflow.tests.test_vpg_agent.TestVPGAGent method), 341
- addCleanup() (tensorflow.tests.test_vpg_baselines.TestVPGBaselines method), 346
- addCleanup() (tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers method), 352
- addTypeEqualityFunc() (tensorflow.tests.test_constant_agent.TestConstantAgent method), 284
- addTypeEqualityFunc() (tensorflow.tests.test_dqfd_agent.TestDQFDAGent method), 290
- addTypeEqualityFunc() (tensorflow.tests.test_dqn_agent.TestDQNAgent method), 295
- addTypeEqualityFunc() (tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 301
- addTypeEqualityFunc() (tensorflow.tests.test_naf_agent.TestNAFAGent method), 306
- addTypeEqualityFunc() (tensorflow.tests.test_ppo_agent.TestPPOAgent method), 311
- addTypeEqualityFunc() (tensorflow.tests.test_quickstart_example.TestQuickstartExample method), 317

method), 317

addTypeEqualityFunc() (tensorflow.tests.test_random_agent.TestRandomAgent method), 321

addTypeEqualityFunc() (tensorflow.tests.test_reward_estimation.TestRewardEstimation method), 327

addTypeEqualityFunc() (tensorflow.tests.test_trpo_agent.TestTRPOAgent method), 331

addTypeEqualityFunc() (tensorflow.tests.test_tutorial_code.TestTutorialCode method), 337

addTypeEqualityFunc() (tensorflow.tests.test_vpg_agent.TestVPGAgent method), 341

addTypeEqualityFunc() (tensorflow.tests.test_vpg_baselines.TestVPGBaselines method), 347

addTypeEqualityFunc() (tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers method), 352

Agent (class in tensorflow.agents), 50

Agent (class in tensorflow.agents.agent), 26

agent (tensorflow.tests.base_agent_test.BaseAgentTest attribute), 282

agent (tensorflow.tests.base_test.BaseTest attribute), 283

agent (tensorflow.tests.test_constant_agent.TestConstantAgent attribute), 284

agent (tensorflow.tests.test_dqfd_agent.TestDQFDAGent attribute), 290

agent (tensorflow.tests.test_dqn_agent.TestDQNAgent attribute), 295

agent (tensorflow.tests.test_dqn_nstep_agent.TestDQNNStepAgent attribute), 301

agent (tensorflow.tests.test_naf_agent.TestNAFAGent attribute), 306

agent (tensorflow.tests.test_ppo_agent.TestPPOAgent attribute), 312

agent (tensorflow.tests.test_random_agent.TestRandomAgent attribute), 321

agent (tensorflow.tests.test_trpo_agent.TestTRPOAgent attribute), 331

agent (tensorflow.tests.test_vpg_agent.TestVPGAgent attribute), 341

agent (tensorflow.tests.test_vpg_baselines.TestVPGBaselines attribute), 347

agent (tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers attribute), 352

agents (tensorflow.execution.threaded_runner.ThreadedRunner attribute), 185

agents (tensorflow.execution.ThreadedRunner attribute), 188

AggregatedBaseline (class in tensorflow.core.baselines), 89

AggregatedBaseline (class in tensorflow.core.baselines.aggregated_baseline), 83

ALE (class in tensorflow.contrib.ale), 74

apply_step() (tensorflow.core.optimizers.clipped_step.ClippedStep method), 152

apply_step() (tensorflow.core.optimizers.ClippedStep method), 174

apply_step() (tensorflow.core.optimizers.Evolutionary method), 171

apply_step() (tensorflow.core.optimizers.evolutionary.Evolutionary method), 153

apply_step() (tensorflow.core.optimizers.global_optimizer.GlobalOptimizer method), 154

apply_step() (tensorflow.core.optimizers.GlobalOptimizer method), 168

apply_step() (tensorflow.core.optimizers.meta_optimizer.MetaOptimizer method), 156

apply_step() (tensorflow.core.optimizers.MetaOptimizer method), 167

apply_step() (tensorflow.core.optimizers.multi_step.MultiStep method), 157

apply_step() (tensorflow.core.optimizers.MultiStep method), 175

apply_step() (tensorflow.core.optimizers.natural_gradient.NaturalGradient method), 159

apply_step() (tensorflow.core.optimizers.NaturalGradient method), 173

apply_step() (tensorflow.core.optimizers.optimized_step.OptimizedStep method), 160

apply_step() (tensorflow.core.optimizers.OptimizedStep method), 177

apply_step() (tensorflow.core.optimizers.Optimizer method), 166

apply_step() (tensorflow.core.optimizers.optimizer.Optimizer method), 162

apply_step() (tensorflow.core.optimizers.SubsamplingStep method), 178

apply_step() (tensorflow.core.optimizers.Synchronization method), 179

apply_step() (tensorflow.core.optimizers.synchronization.Synchronization method), 163

apply_step() (tensorflow.core.optimizers.tf_optimizer.TFOptimizer method), 165

apply_step() (tensorflow.core.optimizers.TFOptimizer method), 170

errors (tensorflow.exception.TensorForceError attribute), 357

errors (tensorflow.TensorForceError attribute), 359

as_local_model() (tensorflow.models.constant_model.ConstantModel method), 189

as_local_model() (tensorflow-

force.models.distribution_model.DistributionModel	assert_() (tensorflow.tests.test_dqn_agent.TestDQNAgent
method), 192	method), 298
as_local_model() (tensorflow.models.DistributionModel	assert_() (tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent
method), 244	method), 304
as_local_model() (tensorflow.models.DPGTargetModel	assert_() (tensorflow.tests.test_naf_agent.TestNAFAgent
method), 257	method), 309
as_local_model() (tensorflow.models.MemoryModel	assert_() (tensorflow.tests.test_ppo_agent.TestPPOAgent
method), 239	method), 315
as_local_model() (tensorflow.models.Model method),	assert_() (tensorflow.tests.test_quickstart_example.TestQuickstartExample
235	method), 320
as_local_model() (tensorflow.models.model.Model	assert_() (tensorflow.tests.test_random_agent.TestRandomAgent
method), 198	method), 324
as_local_model() (tensorflow.models.pg_log_prob_model.PGLogProbModel	assert_() (tensorflow.tests.test_reward_estimation.TestRewardEstimation
method), 201	method), 330
as_local_model() (tensorflow.models.pg_model.PGModel	assert_() (tensorflow.tests.test_trpo_agent.TestTRPOAgent
method), 205	method), 334
as_local_model() (tensorflow.models.pg_prob_ratio_model.PGProbRatioModel	assert_() (tensorflow.tests.test_tutorial_code.TestTutorialCode
method), 211	method), 340
as_local_model() (tensorflow.models.PGLogProbModel	assert_() (tensorflow.tests.test_vpg_agent.TestVPGAgent
method), 261	method), 344
as_local_model() (tensorflow.models.PGModel	assert_() (tensorflow.tests.test_vpg_baselines.TestVPGBaselines
method), 249	method), 350
as_local_model() (tensorflow.models.PGProbRatioModel	assert_() (tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers
method), 253	method), 355
as_local_model() (tensorflow.models.q_demo_model.QDemoModel	assertAlmostEqual() (tensor-
method), 215	force.tests.test_constant_agent.TestConstantAgent
as_local_model() (tensorflow.models.q_model.QModel	method), 284
method), 220	assertAlmostEqual() (tensor-
as_local_model() (tensorflow.models.q_naf_model.QNAFModel	force.tests.test_dqfd_agent.TestDQFDAgent
method), 224	method), 290
as_local_model() (tensorflow.models.q_nstep_model.QNstepModel	assertAlmostEqual() (tensor-
method), 228	force.tests.test_dqn_agent.TestDQNAgent
as_local_model() (tensorflow.models.QDemoModel	method), 295
method), 278	assertAlmostEqual() (tensor-
as_local_model() (tensorflow.models.QModel method),	force.tests.test_dqn_nstep_agent.TestDQNNstepAgent
266	method), 301
as_local_model() (tensorflow.models.QNAFModel	assertAlmostEqual() (tensor-
method), 274	force.tests.test_naf_agent.TestNAFAgent
as_local_model() (tensorflow.models.QNstepModel	method), 306
method), 270	assertAlmostEqual() (tensor-
as_local_model() (tensorflow.models.random_model.RandomModel	force.tests.test_ppo_agent.TestPPOAgent
method), 232	method), 312
assert_() (tensorflow.tests.test_constant_agent.TestConstantAgent	assertAlmostEqual() (tensor-
method), 287	force.tests.test_quickstart_example.TestQuickstartExample
assert_() (tensorflow.tests.test_dqfd_agent.TestDQFDAgent	method), 317
method), 293	assertAlmostEqual() (tensor-
	force.tests.test_random_agent.TestRandomAgent
	method), 322
	assertAlmostEqual() (tensor-
	force.tests.test_reward_estimation.TestRewardEstimation
	method), 327
	assertAlmostEqual() (tensor-
	force.tests.test_trpo_agent.TestTRPOAgent
	method), 331

<code>assertAlmostEqual()</code> (tensor- force.tests.test_tutorial_code.TestTutorialCode method), 337	<code>assertDictContainsSubset()</code> (tensor- force.tests.test_constant_agent.TestConstantAgent method), 285
<code>assertAlmostEqual()</code> (tensor- force.tests.test_vpg_agent.TestVPGAgent method), 341	<code>assertDictContainsSubset()</code> (tensor- force.tests.test_dqfd_agent.TestDQFDAGent method), 290
<code>assertAlmostEqual()</code> (tensor- force.tests.test_vpg_baselines.TestVPGBaselines method), 347	<code>assertDictContainsSubset()</code> (tensor- force.tests.test_dqn_agent.TestDQNAgent method), 296
<code>assertAlmostEqual()</code> (tensor- force.tests.test_vpg_optimizers.TestVPGOptimizers method), 352	<code>assertDictContainsSubset()</code> (tensor- force.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 301
<code>assertAlmostEquals()</code> (tensor- force.tests.test_constant_agent.TestConstantAgent method), 285	<code>assertDictContainsSubset()</code> (tensor- force.tests.test_naf_agent.TestNAFAgent method), 307
<code>assertAlmostEquals()</code> (tensor- force.tests.test_dqfd_agent.TestDQFDAGent method), 290	<code>assertDictContainsSubset()</code> (tensor- force.tests.test_ppo_agent.TestPPOAgent method), 312
<code>assertAlmostEquals()</code> (tensor- force.tests.test_dqn_agent.TestDQNAgent method), 295	<code>assertDictContainsSubset()</code> (tensor- force.tests.test_quickstart_example.TestQuickstartExample method), 317
<code>assertAlmostEquals()</code> (tensor- force.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 301	<code>assertDictContainsSubset()</code> (tensor- force.tests.test_random_agent.TestRandomAgent method), 322
<code>assertAlmostEquals()</code> (tensor- force.tests.test_naf_agent.TestNAFAgent method), 306	<code>assertDictContainsSubset()</code> (tensor- force.tests.test_reward_estimation.TestRewardEstimation method), 327
<code>assertAlmostEquals()</code> (tensor- force.tests.test_ppo_agent.TestPPOAgent method), 312	<code>assertDictContainsSubset()</code> (tensor- force.tests.test_trpo_agent.TestTRPOAgent method), 332
<code>assertAlmostEquals()</code> (tensor- force.tests.test_quickstart_example.TestQuickstartExample method), 317	<code>assertDictContainsSubset()</code> (tensor- force.tests.test_tutorial_code.TestTutorialCode method), 337
<code>assertAlmostEquals()</code> (tensor- force.tests.test_random_agent.TestRandomAgent method), 322	<code>assertDictContainsSubset()</code> (tensor- force.tests.test_vpg_agent.TestVPGAgent method), 342
<code>assertAlmostEquals()</code> (tensor- force.tests.test_reward_estimation.TestRewardEstimation method), 327	<code>assertDictContainsSubset()</code> (tensor- force.tests.test_vpg_baselines.TestVPGBaselines method), 347
<code>assertAlmostEquals()</code> (tensor- force.tests.test_trpo_agent.TestTRPOAgent method), 332	<code>assertDictContainsSubset()</code> (tensor- force.tests.test_vpg_optimizers.TestVPGOptimizers method), 352
<code>assertAlmostEquals()</code> (tensor- force.tests.test_tutorial_code.TestTutorialCode method), 337	<code>assertDictEqual()</code> (tensor- force.tests.test_constant_agent.TestConstantAgent method), 285
<code>assertAlmostEquals()</code> (tensor- force.tests.test_vpg_agent.TestVPGAgent method), 342	<code>assertDictEqual()</code> (tensor- force.tests.test_dqfd_agent.TestDQFDAGent method), 290
<code>assertAlmostEquals()</code> (tensor- force.tests.test_vpg_baselines.TestVPGBaselines method), 347	<code>assertDictEqual()</code> (tensor- force.tests.test_dqn_agent.TestDQNAgent method), 296
<code>assertAlmostEquals()</code> (tensor- force.tests.test_vpg_optimizers.TestVPGOptimizers method), 352	<code>assertDictEqual()</code> (tensor- force.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 301

`assertDictEqual()` (tensorforce.tests.test_naf_agent.TestNAFAgent method), 307
`assertDictEqual()` (tensorforce.tests.test_ppo_agent.TestPPOAgent method), 312
`assertDictEqual()` (tensorforce.tests.test_quickstart_example.TestQuickstartExample method), 317
`assertDictEqual()` (tensorforce.tests.test_random_agent.TestRandomAgent method), 322
`assertDictEqual()` (tensorforce.tests.test_reward_estimation.TestRewardEstimation method), 327
`assertDictEqual()` (tensorforce.tests.test_trpo_agent.TestTRPOAgent method), 332
`assertDictEqual()` (tensorforce.tests.test_tutorial_code.TestTutorialCode method), 337
`assertDictEqual()` (tensorforce.tests.test_vpg_agent.TestVPAGent method), 342
`assertDictEqual()` (tensorforce.tests.test_vpg_baselines.TestVPGBaselines method), 347
`assertDictEqual()` (tensorforce.tests.test_vpg_optimizers.TestVPGOptimizers method), 352
`assertEqual()` (tensorforce.tests.test_constant_agent.TestConstantAgent method), 285
`assertEqual()` (tensorforce.tests.test_dqfd_agent.TestDQFDAGent method), 290
`assertEqual()` (tensorforce.tests.test_dqn_agent.TestDQNAgent method), 296
`assertEqual()` (tensorforce.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 301
`assertEqual()` (tensorforce.tests.test_naf_agent.TestNAFAgent method), 307
`assertEqual()` (tensorforce.tests.test_ppo_agent.TestPPOAgent method), 312
`assertEqual()` (tensorforce.tests.test_quickstart_example.TestQuickstartExample method), 317
`assertEqual()` (tensorforce.tests.test_random_agent.TestRandomAgent method), 322
`assertEqual()` (tensorforce.tests.test_reward_estimation.TestRewardEstimation method), 327
`assertEqual()` (tensorforce.tests.test_trpo_agent.TestTRPOAgent method), 332
`assertEqual()` (tensorforce.tests.test_tutorial_code.TestTutorialCode method), 337
`assertEqual()` (tensorforce.tests.test_vpg_agent.TestVPAGent method), 342
`assertEqual()` (tensorforce.tests.test_vpg_baselines.TestVPGBaselines method), 347
`assertEqual()` (tensorforce.tests.test_vpg_optimizers.TestVPGOptimizers method), 352
`assertFalse()` (tensorforce.tests.test_constant_agent.TestConstantAgent method), 285
`assertFalse()` (tensorforce.tests.test_dqfd_agent.TestDQFDAGent method), 290
`assertFalse()` (tensorforce.tests.test_dqn_agent.TestDQNAgent method), 296
`assertFalse()` (tensorforce.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 301
`assertFalse()` (tensorforce.tests.test_naf_agent.TestNAFAgent method), 307
`assertFalse()` (tensorforce.tests.test_ppo_agent.TestPPOAgent method), 312
`assertFalse()` (tensorforce.tests.test_quickstart_example.TestQuickstartExample method), 317
`assertFalse()` (tensorforce.tests.test_random_agent.TestRandomAgent method), 322
`assertFalse()` (tensorforce.tests.test_reward_estimation.TestRewardEstimation method), 327
`assertFalse()` (tensorforce.tests.test_trpo_agent.TestTRPOAgent method), 332
`assertFalse()` (tensorforce.tests.test_tutorial_code.TestTutorialCode method), 337
`assertFalse()` (tensorforce.tests.test_vpg_agent.TestVPAGent method), 342
`assertFalse()` (tensorforce.tests.test_vpg_baselines.TestVPGBaselines method), 347
`assertFalse()` (tensorforce.tests.test_vpg_optimizers.TestVPGOptimizers method), 352

[illegible]

[assertIs\(\) \(tensorflow.tests.test_dqn_agent.TestDQNAgent method\), 296](#)
[assertIs\(\) \(tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent method\), 301](#)
[assertIs\(\) \(tensorflow.tests.test_naf_agent.TestNAFAgent method\), 307](#)
[assertIs\(\) \(tensorflow.tests.test_ppo_agent.TestPPOAgent method\), 312](#)
[assertIs\(\) \(tensorflow.tests.test_quickstart_example.TestQuickstartExample method\), 318](#)
[assertIs\(\) \(tensorflow.tests.test_random_agent.TestRandomAgent method\), 322](#)
[assertIs\(\) \(tensorflow.tests.test_reward_estimation.TestRewardEstimation method\), 328](#)
[assertIs\(\) \(tensorflow.tests.test_trpo_agent.TestTRPOAgent method\), 332](#)
[assertIs\(\) \(tensorflow.tests.test_tutorial_code.TestTutorialCode method\), 338](#)
[assertIs\(\) \(tensorflow.tests.test_vpg_agent.TestVPGAgent method\), 342](#)
[assertIs\(\) \(tensorflow.tests.test_vpg_baselines.TestVPGBaselines method\), 348](#)
[assertIs\(\) \(tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers method\), 353](#)
[assertIsNone\(\) \(tensorflow.tests.test_constant_agent.TestConstantAgent method\), 285](#)
[assertIsNone\(\) \(tensorflow.tests.test_dqfd_agent.TestDQFDAgent method\), 291](#)
[assertIsNone\(\) \(tensorflow.tests.test_dqn_agent.TestDQNAgent method\), 296](#)
[assertIsNone\(\) \(tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent method\), 302](#)
[assertIsNone\(\) \(tensorflow.tests.test_naf_agent.TestNAFAgent method\), 307](#)
[assertIsNone\(\) \(tensorflow.tests.test_ppo_agent.TestPPOAgent method\), 312](#)
[assertIsNone\(\) \(tensorflow.tests.test_quickstart_example.TestQuickstartExample method\), 318](#)
[assertIsNone\(\) \(tensorflow.tests.test_random_agent.TestRandomAgent method\), 322](#)
[assertIsNone\(\) \(tensorflow.tests.test_reward_estimation.TestRewardEstimation method\), 328](#)
[assertIsNone\(\) \(tensorflow.tests.test_trpo_agent.TestTRPOAgent method\), 332](#)
[assertIsNone\(\) \(tensorflow.tests.test_tutorial_code.TestTutorialCode method\), 338](#)
[assertIsNone\(\) \(tensorflow.tests.test_vpg_agent.TestVPGAgent method\), 342](#)
[assertIsNone\(\) \(tensorflow.tests.test_vpg_baselines.TestVPGBaselines method\), 348](#)
[assertIsNone\(\) \(tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers method\), 353](#)
[assertIsNot\(\) \(tensorflow.tests.test_constant_agent.TestConstantAgent method\), 285](#)
[assertIsNot\(\) \(tensorflow.tests.test_dqfd_agent.TestDQFDAgent method\), 291](#)
[assertIsNot\(\) \(tensorflow.tests.test_dqn_agent.TestDQNAgent method\), 296](#)
[assertIsNot\(\) \(tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent method\), 302](#)
[assertIsNot\(\) \(tensorflow.tests.test_naf_agent.TestNAFAgent method\), 307](#)
[assertIsNot\(\) \(tensorflow.tests.test_ppo_agent.TestPPOAgent method\), 312](#)
[assertIsNot\(\) \(tensorflow.tests.test_quickstart_example.TestQuickstartExample method\), 318](#)

`assertIsNot()` (tensorflow.tests.test_random_agent.TestRandomAgent method), 322

`assertIsNot()` (tensorflow.tests.test_reward_estimation.TestRewardEstimation method), 328

`assertIsNot()` (tensorflow.tests.test_trpo_agent.TestTRPOAgent method), 332

`assertIsNot()` (tensorflow.tests.test_tutorial_code.TestTutorialCode method), 338

`assertIsNot()` (tensorflow.tests.test_vpg_agent.TestVPAGent method), 342

`assertIsNot()` (tensorflow.tests.test_vpg_baselines.TestVPGBaselines method), 348

`assertIsNot()` (tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers method), 353

`assertIsNotNone()` (tensorflow.tests.test_constant_agent.TestConstantAgent method), 285

`assertIsNotNone()` (tensorflow.tests.test_dqfd_agent.TestDQFDAgent method), 291

`assertIsNotNone()` (tensorflow.tests.test_dqn_agent.TestDQNAgent method), 296

`assertIsNotNone()` (tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 302

`assertIsNotNone()` (tensorflow.tests.test_naf_agent.TestNAFAgent method), 307

`assertIsNotNone()` (tensorflow.tests.test_ppo_agent.TestPPOAgent method), 312

`assertIsNotNone()` (tensorflow.tests.test_quickstart_example.TestQuickstartExample method), 318

`assertIsNotNone()` (tensorflow.tests.test_random_agent.TestRandomAgent method), 322

`assertIsNotNone()` (tensorflow.tests.test_reward_estimation.TestRewardEstimation method), 328

`assertIsNotNone()` (tensorflow.tests.test_trpo_agent.TestTRPOAgent method), 332

`assertIsNotNone()` (tensorflow.tests.test_tutorial_code.TestTutorialCode method), 338

`assertIsNotNone()` (tensorflow.tests.test_vpg_agent.TestVPAGent method), 342

`assertIsNotNone()` (tensorflow.tests.test_vpg_baselines.TestVPGBaselines method), 348

`assertIsNotNone()` (tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers method), 353

`assertLess()` (tensorflow.tests.test_constant_agent.TestConstantAgent method), 285

`assertLess()` (tensorflow.tests.test_dqfd_agent.TestDQFDAgent method), 291

`assertLess()` (tensorflow.tests.test_dqn_agent.TestDQNAgent method), 296

`assertLess()` (tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 302

`assertLess()` (tensorflow.tests.test_naf_agent.TestNAFAgent method), 307

`assertItemsEqual()` (tensorflow.tests.test_constant_agent.TestConstantAgent method), 285

`assertItemsEqual()` (tensorflow.tests.test_dqfd_agent.TestDQFDAgent method), 291

`assertItemsEqual()` (tensorflow.tests.test_dqn_agent.TestDQNAgent method), 296

`assertItemsEqual()` (tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 302

`assertItemsEqual()` (tensorflow.tests.test_naf_agent.TestNAFAgent method), 307

`assertItemsEqual()` (tensorflow.tests.test_ppo_agent.TestPPOAgent method), 312

`assertItemsEqual()` (tensorflow.tests.test_quickstart_example.TestQuickstartExample method), 318

`assertItemsEqual()` (tensorflow.tests.test_random_agent.TestRandomAgent method), 322

`assertItemsEqual()` (tensorflow.tests.test_reward_estimation.TestRewardEstimation method), 328

`assertItemsEqual()` (tensorflow.tests.test_trpo_agent.TestTRPOAgent method), 332

`assertItemsEqual()` (tensorflow.tests.test_tutorial_code.TestTutorialCode method), 338

`assertItemsEqual()` (tensorflow.tests.test_vpg_agent.TestVPAGent method), 342

`assertItemsEqual()` (tensorflow.tests.test_vpg_baselines.TestVPGBaselines method), 348

`assertItemsEqual()` (tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers method), 353

[assertLess\(\) \(tensorflow.tests.test_ppo_agent.TestPPOAgent.assertLessEqual\(\) method\), 313](#)
[assertLess\(\) \(tensorflow.tests.test_quickstart_example.TestQuickstartExample.assertLessEqual\(\) method\), 318](#)
[assertLess\(\) \(tensorflow.tests.test_random_agent.TestRandomAgent.assertLessEqual\(\) method\), 323](#)
[assertLess\(\) \(tensorflow.tests.test_reward_estimation.TestRewardEstimation.assertLessEqual\(\) method\), 328](#)
[assertLess\(\) \(tensorflow.tests.test_trpo_agent.TestTRPOAgent.assertLessEqual\(\) method\), 332](#)
[assertLess\(\) \(tensorflow.tests.test_tutorial_code.TestTutorialCode.assertLessEqual\(\) method\), 338](#)
[assertLess\(\) \(tensorflow.tests.test_vpg_agent.TestVPGAgent.assertLessEqual\(\) method\), 342](#)
[assertLess\(\) \(tensorflow.tests.test_vpg_baselines.TestVPGBaselines.assertLessEqual\(\) method\), 348](#)
[assertLess\(\) \(tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers.assertLessEqual\(\) method\), 353](#)
[assertLessEqual\(\) \(tensorflow.tests.test_constant_agent.TestConstantAgent.assertLessEqual\(\) method\), 285](#)
[assertLessEqual\(\) \(tensorflow.tests.test_dqfd_agent.TestDQFDAgent.assertLessEqual\(\) method\), 291](#)
[assertLessEqual\(\) \(tensorflow.tests.test_dqn_agent.TestDQNAgent.assertLessEqual\(\) method\), 296](#)
[assertLessEqual\(\) \(tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent.assertLessEqual\(\) method\), 302](#)
[assertLessEqual\(\) \(tensorflow.tests.test_naf_agent.TestNAFAgent.assertLessEqual\(\) method\), 307](#)
[assertLessEqual\(\) \(tensorflow.tests.test_ppo_agent.TestPPOAgent.assertLessEqual\(\) method\), 313](#)
[assertLessEqual\(\) \(tensorflow.tests.test_quickstart_example.TestQuickstartExample.assertLessEqual\(\) method\), 318](#)
[assertLessEqual\(\) \(tensorflow.tests.test_random_agent.TestRandomAgent.assertLessEqual\(\) method\), 323](#)
[assertLessEqual\(\) \(tensorflow.tests.test_reward_estimation.TestRewardEstimation.assertLessEqual\(\) method\), 328](#)
[assertLessEqual\(\) \(tensorflow.tests.test_trpo_agent.TestTRPOAgent.assertLessEqual\(\) method\), 333](#)
[assertLessEqual\(\) \(tensorflow.tests.test_tutorial_code.TestTutorialCode.assertLessEqual\(\) method\), 338](#)
[assertLessEqual\(\) \(tensorflow.tests.test_vpg_agent.TestVPGAgent.assertLessEqual\(\) method\), 343](#)
[assertLessEqual\(\) \(tensorflow.tests.test_vpg_baselines.TestVPGBaselines.assertLessEqual\(\) method\), 348](#)
[assertLessEqual\(\) \(tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers.assertLessEqual\(\) method\), 353](#)
[assertMultiLineEqual\(\) \(tensorflow.tests.test_constant_agent.TestConstantAgent.assertMultiLineEqual\(\) method\), 286](#)
[assertMultiLineEqual\(\) \(tensorflow.tests.test_dqfd_agent.TestDQFDAgent.assertMultiLineEqual\(\) method\), 291](#)

<code>assertMultiLineEqual()</code> force.tests.test_dqn_agent.TestDQNAgent method), 297	(tensor-	<code>assertNotAlmostEqual()</code> force.tests.test_quickstart_example.TestQuickstartExample method), 318	(tensor-
<code>assertMultiLineEqual()</code> force.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 302	(tensor-	<code>assertNotAlmostEqual()</code> force.tests.test_random_agent.TestRandomAgent method), 323	(tensor-
<code>assertMultiLineEqual()</code> force.tests.test_naf_agent.TestNAFAgent method), 307	(tensor-	<code>assertNotAlmostEqual()</code> force.tests.test_reward_estimation.TestRewardEstimation method), 328	(tensor-
<code>assertMultiLineEqual()</code> force.tests.test_ppo_agent.TestPPOAgent method), 313	(tensor-	<code>assertNotAlmostEqual()</code> force.tests.test_trpo_agent.TestTRPOAgent method), 333	(tensor-
<code>assertMultiLineEqual()</code> force.tests.test_quickstart_example.TestQuickstartExample method), 318	(tensor-	<code>assertNotAlmostEqual()</code> force.tests.test_tutorial_code.TestTutorialCode method), 338	(tensor-
<code>assertMultiLineEqual()</code> force.tests.test_random_agent.TestRandomAgent method), 323	(tensor-	<code>assertNotAlmostEqual()</code> force.tests.test_vpg_agent.TestVPGAgent method), 343	(tensor-
<code>assertMultiLineEqual()</code> force.tests.test_reward_estimation.TestRewardEstimation method), 328	(tensor-	<code>assertNotAlmostEqual()</code> force.tests.test_vpg_baselines.TestVPGBaselines method), 348	(tensor-
<code>assertMultiLineEqual()</code> force.tests.test_trpo_agent.TestTRPOAgent method), 333	(tensor-	<code>assertNotAlmostEqual()</code> force.tests.test_vpg_optimizers.TestVPGOptimizers method), 353	(tensor-
<code>assertMultiLineEqual()</code> force.tests.test_tutorial_code.TestTutorialCode method), 338	(tensor-	<code>assertNotAlmostEquals()</code> force.tests.test_constant_agent.TestConstantAgent method), 286	(tensor-
<code>assertMultiLineEqual()</code> force.tests.test_vpg_agent.TestVPGAgent method), 343	(tensor-	<code>assertNotAlmostEquals()</code> force.tests.test_dqfd_agent.TestDQFDAGENT method), 291	(tensor-
<code>assertMultiLineEqual()</code> force.tests.test_vpg_baselines.TestVPGBaselines method), 348	(tensor-	<code>assertNotAlmostEquals()</code> force.tests.test_dqn_agent.TestDQNAgent method), 297	(tensor-
<code>assertMultiLineEqual()</code> force.tests.test_vpg_optimizers.TestVPGOptimizers method), 353	(tensor-	<code>assertNotAlmostEquals()</code> force.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 302	(tensor-
<code>assertNotAlmostEqual()</code> force.tests.test_constant_agent.TestConstantAgent method), 286	(tensor-	<code>assertNotAlmostEquals()</code> force.tests.test_naf_agent.TestNAFAgent method), 308	(tensor-
<code>assertNotAlmostEqual()</code> force.tests.test_dqfd_agent.TestDQFDAGENT method), 291	(tensor-	<code>assertNotAlmostEquals()</code> force.tests.test_ppo_agent.TestPPOAgent method), 313	(tensor-
<code>assertNotAlmostEqual()</code> force.tests.test_dqn_agent.TestDQNAgent method), 297	(tensor-	<code>assertNotAlmostEquals()</code> force.tests.test_quickstart_example.TestQuickstartExample method), 319	(tensor-
<code>assertNotAlmostEqual()</code> force.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 302	(tensor-	<code>assertNotAlmostEquals()</code> force.tests.test_random_agent.TestRandomAgent method), 323	(tensor-
<code>assertNotAlmostEqual()</code> force.tests.test_naf_agent.TestNAFAgent method), 308	(tensor-	<code>assertNotAlmostEquals()</code> force.tests.test_reward_estimation.TestRewardEstimation method), 328	(tensor-
<code>assertNotAlmostEqual()</code> force.tests.test_ppo_agent.TestPPOAgent method), 313	(tensor-	<code>assertNotAlmostEquals()</code> force.tests.test_trpo_agent.TestTRPOAgent method), 333	(tensor-

<code>assertNotAlmostEquals()</code> (tensorflow.tests.test_tutorial_code.TestTutorialCode method), 338	<code>assertNotEquals()</code> (tensorflow.tests.test_constant_agent.TestConstantAgent method), 286
<code>assertNotAlmostEquals()</code> (tensorflow.tests.test_vpg_agent.TestVPGAgent method), 343	<code>assertNotEquals()</code> (tensorflow.tests.test_dqfd_agent.TestDQFDAgent method), 292
<code>assertNotAlmostEquals()</code> (tensorflow.tests.test_vpg_baselines.TestVPGBaselines method), 348	<code>assertNotEquals()</code> (tensorflow.tests.test_dqn_agent.TestDQNAgent method), 297
<code>assertNotAlmostEquals()</code> (tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers method), 353	<code>assertNotEquals()</code> (tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 302
<code>assertNotEqual()</code> (tensorflow.tests.test_constant_agent.TestConstantAgent method), 286	<code>assertNotEquals()</code> (tensorflow.tests.test_naf_agent.TestNAFAgent method), 308
<code>assertNotEqual()</code> (tensorflow.tests.test_dqfd_agent.TestDQFDAgent method), 292	<code>assertNotEquals()</code> (tensorflow.tests.test_ppo_agent.TestPPOAgent method), 313
<code>assertNotEqual()</code> (tensorflow.tests.test_dqn_agent.TestDQNAgent method), 297	<code>assertNotEquals()</code> (tensorflow.tests.test_quickstart_example.TestQuickstartExample method), 319
<code>assertNotEqual()</code> (tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 302	<code>assertNotEquals()</code> (tensorflow.tests.test_random_agent.TestRandomAgent method), 323
<code>assertNotEqual()</code> (tensorflow.tests.test_naf_agent.TestNAFAgent method), 308	<code>assertNotEquals()</code> (tensorflow.tests.test_reward_estimation.TestRewardEstimation method), 329
<code>assertNotEqual()</code> (tensorflow.tests.test_ppo_agent.TestPPOAgent method), 313	<code>assertNotEquals()</code> (tensorflow.tests.test_trpo_agent.TestTRPOAgent method), 333
<code>assertNotEqual()</code> (tensorflow.tests.test_quickstart_example.TestQuickstartExample method), 319	<code>assertNotEquals()</code> (tensorflow.tests.test_tutorial_code.TestTutorialCode method), 339
<code>assertNotEqual()</code> (tensorflow.tests.test_random_agent.TestRandomAgent method), 323	<code>assertNotEquals()</code> (tensorflow.tests.test_vpg_agent.TestVPGAgent method), 343
<code>assertNotEqual()</code> (tensorflow.tests.test_reward_estimation.TestRewardEstimation method), 329	<code>assertNotEquals()</code> (tensorflow.tests.test_vpg_baselines.TestVPGBaselines method), 349
<code>assertNotEqual()</code> (tensorflow.tests.test_trpo_agent.TestTRPOAgent method), 333	<code>assertNotEquals()</code> (tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers method), 354
<code>assertNotEqual()</code> (tensorflow.tests.test_tutorial_code.TestTutorialCode method), 339	<code>assertNotIn()</code> (tensorflow.tests.test_constant_agent.TestConstantAgent method), 286
<code>assertNotEqual()</code> (tensorflow.tests.test_vpg_agent.TestVPGAgent method), 343	<code>assertNotIn()</code> (tensorflow.tests.test_dqfd_agent.TestDQFDAgent method), 292
<code>assertNotEqual()</code> (tensorflow.tests.test_vpg_baselines.TestVPGBaselines method), 349	<code>assertNotIn()</code> (tensorflow.tests.test_dqn_agent.TestDQNAgent method), 297
<code>assertNotEqual()</code> (tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers method), 354	<code>assertNotIn()</code> (tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 303
	<code>assertNotIn()</code> (tensorflow.tests.test_naf_agent.TestNAFAgent method), 308
	<code>assertNotIn()</code> (tensorflow.tests.test_ppo_agent.TestPPOAgent method), 313

[assertNotIn\(\) \(tensorflow.tests.test_quickstart_example.TestQuickstartExample method\), 319](#)
[assertNotIn\(\) \(tensorflow.tests.test_random_agent.TestRandomAgent method\), 323](#)
[assertNotIn\(\) \(tensorflow.tests.test_reward_estimation.TestRewardEstimation method\), 329](#)
[assertNotIn\(\) \(tensorflow.tests.test_trpo_agent.TestTRPOAgent method\), 333](#)
[assertNotIn\(\) \(tensorflow.tests.test_tutorial_code.TestTutorialCode method\), 339](#)
[assertNotIn\(\) \(tensorflow.tests.test_vpg_agent.TestVPAGent method\), 343](#)
[assertNotIn\(\) \(tensorflow.tests.test_vpg_baselines.TestVPGBaselines method\), 349](#)
[assertNotIn\(\) \(tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers method\), 354](#)
[assertNotIsInstance\(\) \(tensorflow.tests.test_constant_agent.TestConstantAgent method\), 286](#)
[assertNotIsInstance\(\) \(tensorflow.tests.test_dqfd_agent.TestDQFDAgent method\), 292](#)
[assertNotIsInstance\(\) \(tensorflow.tests.test_dqn_agent.TestDQNAgent method\), 297](#)
[assertNotIsInstance\(\) \(tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent method\), 303](#)
[assertNotIsInstance\(\) \(tensorflow.tests.test_naf_agent.TestNAFAgent method\), 308](#)
[assertNotIsInstance\(\) \(tensorflow.tests.test_ppo_agent.TestPPOAgent method\), 313](#)
[assertNotIsInstance\(\) \(tensorflow.tests.test_quickstart_example.TestQuickstartExample method\), 319](#)
[assertNotIsInstance\(\) \(tensorflow.tests.test_random_agent.TestRandomAgent method\), 323](#)
[assertNotIsInstance\(\) \(tensorflow.tests.test_reward_estimation.TestRewardEstimation method\), 329](#)
[assertNotIsInstance\(\) \(tensorflow.tests.test_trpo_agent.TestTRPOAgent method\), 333](#)
[assertNotIsInstance\(\) \(tensorflow.tests.test_tutorial_code.TestTutorialCode method\), 339](#)
[assertNotIsInstance\(\) \(tensorflow.tests.test_vpg_agent.TestVPAGent method\), 343](#)
[assertNotIsInstance\(\) \(tensorflow.tests.test_vpg_baselines.TestVPGBaselines method\), 349](#)
[assertNotIsInstance\(\) \(tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers method\), 354](#)
[assertRaises\(\) \(tensorflow.tests.test_constant_agent.TestConstantAgent method\), 286](#)
[assertRaises\(\) \(tensorflow.tests.test_dqfd_agent.TestDQFDAgent method\), 292](#)
[assertRaises\(\) \(tensorflow.tests.test_dqn_agent.TestDQNAgent method\), 297](#)
[assertRaises\(\) \(tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent method\), 303](#)
[assertRaises\(\) \(tensorflow.tests.test_naf_agent.TestNAFAgent method\), 308](#)
[assertRaises\(\) \(tensorflow.tests.test_ppo_agent.TestPPOAgent method\), 313](#)
[assertRaises\(\) \(tensorflow.tests.test_quickstart_example.TestQuickstartExample method\), 319](#)
[assertRaises\(\) \(tensorflow.tests.test_random_agent.TestRandomAgent method\), 323](#)
[assertRaises\(\) \(tensorflow.tests.test_reward_estimation.TestRewardEstimation method\), 329](#)
[assertRaises\(\) \(tensorflow.tests.test_trpo_agent.TestTRPOAgent method\), 333](#)
[assertRaises\(\) \(tensorflow.tests.test_tutorial_code.TestTutorialCode method\), 339](#)
[assertRaises\(\) \(tensorflow.tests.test_vpg_agent.TestVPAGent method\), 343](#)
[assertRaises\(\) \(tensorflow.tests.test_vpg_baselines.TestVPGBaselines method\), 349](#)
[assertRaises\(\) \(tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers method\), 354](#)

assertRaises() (tensorflow.tests.test_naf_agent.TestNAFAgent method), 308	force.tests.test_vpg_agent.TestVPGAgent method), 343
assertRaises() (tensorflow.tests.test_ppo_agent.TestPPOAgent method), 314	assertRaisesRegexp() (tensor- force.tests.test_vpg_baselines.TestVPGBaselines method), 349
assertRaises() (tensorflow.tests.test_quickstart_example.TestQuickstartExample method), 319	assertRaisesRegexp() (tensor- force.tests.test_vpg_optimizers.TestVPGOptimizers method), 354
assertRaises() (tensorflow.tests.test_random_agent.TestRandomAgent method), 323	assertRaisesRegexp() (tensor- force.tests.test_reward_estimation.TestRewardEstimation method), 329
assertRaises() (tensorflow.tests.test_reward_estimation.TestRewardEstimation method), 329	force.tests.test_constant_agent.TestConstantAgent method), 287
assertRaises() (tensorflow.tests.test_trpo_agent.TestTRPOAgent method), 333	assertRegexpMatches() (tensor- force.tests.test_dqfd_agent.TestDQFDAgent method), 292
assertRaises() (tensorflow.tests.test_tutorial_code.TestTutorialCode method), 339	assertRegexpMatches() (tensor- force.tests.test_dqn_agent.TestDQNAgent method), 298
assertRaises() (tensorflow.tests.test_vpg_agent.TestVPGAgent method), 343	assertRegexpMatches() (tensor- force.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 303
assertRaises() (tensorflow.tests.test_vpg_baselines.TestVPGBaselines method), 349	assertRegexpMatches() (tensor- force.tests.test_naf_agent.TestNAFAgent method), 308
assertRaises() (tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers method), 354	assertRegexpMatches() (tensor- force.tests.test_ppo_agent.TestPPOAgent method), 314
assertRaisesRegexp() (tensor- force.tests.test_constant_agent.TestConstantAgent method), 286	assertRegexpMatches() (tensor- force.tests.test_quickstart_example.TestQuickstartExample method), 319
assertRaisesRegexp() (tensor- force.tests.test_dqfd_agent.TestDQFDAgent method), 292	assertRegexpMatches() (tensor- force.tests.test_random_agent.TestRandomAgent method), 324
assertRaisesRegexp() (tensor- force.tests.test_dqn_agent.TestDQNAgent method), 297	assertRegexpMatches() (tensor- force.tests.test_reward_estimation.TestRewardEstimation method), 329
assertRaisesRegexp() (tensor- force.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 303	assertRegexpMatches() (tensor- force.tests.test_trpo_agent.TestTRPOAgent method), 334
assertRaisesRegexp() (tensor- force.tests.test_naf_agent.TestNAFAgent method), 308	assertRegexpMatches() (tensor- force.tests.test_tutorial_code.TestTutorialCode method), 339
assertRaisesRegexp() (tensor- force.tests.test_ppo_agent.TestPPOAgent method), 314	assertRegexpMatches() (tensor- force.tests.test_vpg_agent.TestVPGAgent method), 344
assertRaisesRegexp() (tensor- force.tests.test_quickstart_example.TestQuickstartExample method), 319	assertRegexpMatches() (tensor- force.tests.test_vpg_baselines.TestVPGBaselines method), 349
assertRaisesRegexp() (tensor- force.tests.test_random_agent.TestRandomAgent method), 324	assertRegexpMatches() (tensor- force.tests.test_vpg_optimizers.TestVPGOptimizers method), 354
assertRaisesRegexp() (tensor- force.tests.test_reward_estimation.TestRewardEstimation method), 329	assertSequenceEqual() (tensor- force.tests.test_constant_agent.TestConstantAgent method), 287
assertRaisesRegexp() (tensor- force.tests.test_trpo_agent.TestTRPOAgent method), 333	assertSequenceEqual() (tensor-
assertRaisesRegexp() (tensor- force.tests.test_tutorial_code.TestTutorialCode method), 339	
assertRaisesRegexp() (tensor-	

force.tests.test_dqfd_agent.TestDQFDAgent method), 292	force.tests.test_ppo_agent.TestPPOAgent method), 314
assertSequenceEqual() (tensor- force.tests.test_dqn_agent.TestDQNAgent method), 298	assertSetEqual() (tensor- force.tests.test_quickstart_example.TestQuickstartExample method), 320
assertSequenceEqual() (tensor- force.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 303	assertSetEqual() (tensor- force.tests.test_random_agent.TestRandomAgent method), 324
assertSequenceEqual() (tensor- force.tests.test_naf_agent.TestNAFAgent method), 309	assertSetEqual() (tensor- force.tests.test_reward_estimation.TestRewardEstimation method), 329
assertSequenceEqual() (tensor- force.tests.test_ppo_agent.TestPPOAgent method), 314	assertSetEqual() (tensor- force.tests.test_trpo_agent.TestTRPOAgent method), 334
assertSequenceEqual() (tensor- force.tests.test_quickstart_example.TestQuickstartExample method), 319	assertSetEqual() (tensor- force.tests.test_tutorial_code.TestTutorialCode method), 339
assertSequenceEqual() (tensor- force.tests.test_random_agent.TestRandomAgent method), 324	assertSetEqual() (tensor- force.tests.test_vpg_agent.TestVPGAgent method), 344
assertSequenceEqual() (tensor- force.tests.test_reward_estimation.TestRewardEstimation method), 329	assertSetEqual() (tensor- force.tests.test_vpg_baselines.TestVPGBaselines method), 349
assertSequenceEqual() (tensor- force.tests.test_trpo_agent.TestTRPOAgent method), 334	assertSetEqual() (tensor- force.tests.test_vpg_optimizers.TestVPGOptimizers method), 354
assertSequenceEqual() (tensor- force.tests.test_tutorial_code.TestTutorialCode method), 339	assertTrue() (tensorforce.tests.test_constant_agent.TestConstantAgent method), 287
assertSequenceEqual() (tensor- force.tests.test_vpg_agent.TestVPGAgent method), 344	assertTrue() (tensorforce.tests.test_dqfd_agent.TestDQFDAgent method), 293
assertSequenceEqual() (tensor- force.tests.test_vpg_baselines.TestVPGBaselines method), 349	assertTrue() (tensorforce.tests.test_dqn_agent.TestDQNAgent method), 298
assertSequenceEqual() (tensor- force.tests.test_vpg_optimizers.TestVPGOptimizers method), 354	assertTrue() (tensorforce.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 304
assertSetEqual() (tensor- force.tests.test_constant_agent.TestConstantAgent method), 287	assertTrue() (tensorforce.tests.test_naf_agent.TestNAFAgent method), 309
assertSetEqual() (tensor- force.tests.test_dqfd_agent.TestDQFDAgent method), 292	assertTrue() (tensorforce.tests.test_ppo_agent.TestPPOAgent method), 314
assertSetEqual() (tensor- force.tests.test_dqn_agent.TestDQNAgent method), 298	assertTrue() (tensorforce.tests.test_quickstart_example.TestQuickstartExam method), 320
assertSetEqual() (tensor- force.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 303	assertTrue() (tensorforce.tests.test_random_agent.TestRandomAgent method), 324
assertSetEqual() (tensor- force.tests.test_naf_agent.TestNAFAgent method), 309	assertTrue() (tensorforce.tests.test_reward_estimation.TestRewardEstimation method), 330
assertSetEqual() (tensor- force.tests.test_ppo_agent.TestPPOAgent method), 314	assertTrue() (tensorforce.tests.test_trpo_agent.TestTRPOAgent method), 334
assertSetEqual() (tensor- force.tests.test_quickstart_example.TestQuickstartExample method), 319	assertTrue() (tensorforce.tests.test_tutorial_code.TestTutorialCode method), 340
assertSetEqual() (tensor- force.tests.test_random_agent.TestRandomAgent method), 324	assertTrue() (tensorforce.tests.test_vpg_agent.TestVPGAgent method), 344
assertSetEqual() (tensor- force.tests.test_reward_estimation.TestRewardEstimation method), 329	assertTrue() (tensorforce.tests.test_vpg_baselines.TestVPGBaselines method), 350
assertSetEqual() (tensor- force.tests.test_trpo_agent.TestTRPOAgent method), 334	assertTrue() (tensorforce.tests.test_vpg_optimizers.TestVPGOptimizers method), 355

`assertTupleEqual()` (tensorforce.tests.test_constant_agent.TestConstantAgent) `base_test_pass()` (tensorforce.tests.test_constant_agent.TestConstantAgent) `method()`, 293
`assertTupleEqual()` (tensorforce.tests.test_dqn_agent.TestDQNAgent) `base_test_pass()` (tensorforce.tests.test_dqn_agent.TestDQNAgent) `method()`, 298
`assertTupleEqual()` (tensorforce.tests.test_dqfd_agent.TestDQFDAgent) `base_test_pass()` (tensorforce.tests.test_dqfd_agent.TestDQFDAgent) `method()`, 293
`assertTupleEqual()` (tensorforce.tests.test_dqn_nstep_agent.TestDQNNstepAgent) `base_test_pass()` (tensorforce.tests.test_dqn_nstep_agent.TestDQNNstepAgent) `method()`, 304
`assertTupleEqual()` (tensorforce.tests.test_naf_agent.TestNAFAgent) `base_test_pass()` (tensorforce.tests.test_naf_agent.TestNAFAgent) `method()`, 309
`assertTupleEqual()` (tensorforce.tests.test_ppo_agent.TestPPOAgent) `base_test_pass()` (tensorforce.tests.test_ppo_agent.TestPPOAgent) `method()`, 315
`assertTupleEqual()` (tensorforce.tests.test_random_agent.TestRandomAgent) `base_test_pass()` (tensorforce.tests.test_random_agent.TestRandomAgent) `method()`, 325
`assertTupleEqual()` (tensorforce.tests.test_trpo_agent.TestTRPOAgent) `base_test_pass()` (tensorforce.tests.test_trpo_agent.TestTRPOAgent) `method()`, 334
`assertTupleEqual()` (tensorforce.tests.test_quickstart_example.TestQuickstartExample) `base_test_pass()` (tensorforce.tests.test_quickstart_example.TestQuickstartExample) `method()`, 320
`assertTupleEqual()` (tensorforce.tests.test_vpg_agent.TestVPGAgent) `base_test_pass()` (tensorforce.tests.test_vpg_agent.TestVPGAgent) `method()`, 344
`assertTupleEqual()` (tensorforce.tests.test_vpg_baselines.TestVPGBaselines) `base_test_pass()` (tensorforce.tests.test_vpg_baselines.TestVPGBaselines) `method()`, 350
`assertTupleEqual()` (tensorforce.tests.test_vpg_optimizers.TestVPGOptimizers) `base_test_pass()` (tensorforce.tests.test_vpg_optimizers.TestVPGOptimizers) `method()`, 355
`assertTupleEqual()` (tensorforce.tests.test_trpo_agent.TestTRPOAgent) `base_test_run()` (tensorforce.tests.test_trpo_agent.TestTRPOAgent) `method()`, 334
`assertTupleEqual()` (tensorforce.tests.test_tutorial_code.TestTutorialCode) `base_test_run()` (tensorforce.tests.test_tutorial_code.TestTutorialCode) `method()`, 340
`assertTupleEqual()` (tensorforce.tests.test_vpg_agent.TestVPGAgent) `base_test_run()` (tensorforce.tests.test_vpg_agent.TestVPGAgent) `method()`, 344
`assertTupleEqual()` (tensorforce.tests.test_vpg_baselines.TestVPGBaselines) `base_test_run()` (tensorforce.tests.test_vpg_baselines.TestVPGBaselines) `method()`, 350
`assertTupleEqual()` (tensorforce.tests.test_vpg_optimizers.TestVPGOptimizers) `base_test_run()` (tensorforce.tests.test_vpg_optimizers.TestVPGOptimizers) `method()`, 355

B

`base_test_pass()` (tensorforce.tests.base_agent_test.BaseAgentTest) `base_test_run()` (tensorforce.tests.base_agent_test.BaseAgentTest) `method()`, 282
`base_test_pass()` (tensorforce.tests.base_test.BaseTest) `base_test_run()` (tensorforce.tests.base_test.BaseTest) `method()`, 283
`base_test_pass()` (tensorforce.tests.test_constant_agent.TestConstantAgent) `base_test_run()` (tensorforce.tests.test_constant_agent.TestConstantAgent) `method()`, 287
`base_test_pass()` (tensorforce.tests.test_dqfd_agent.TestDQFDAgent) `base_test_run()` (tensorforce.tests.test_dqfd_agent.TestDQFDAgent) `method()`, 325

- base_test_run() (tensorforce.tests.test_trpo_agent.TestTRPOAgent method), 335
- base_test_run() (tensorforce.tests.test_vpg_agent.TestVPGAgent method), 345
- base_test_run() (tensorforce.tests.test_vpg_baselines.TestVPGBaselines method), 350
- base_test_run() (tensorforce.tests.test_vpg_optimizers.TestVPGOptimizers method), 355
- BaseAgentTest (class in tensorforce.tests.base_agent_test), 282
- Baseline (class in tensorforce.core.baselines), 88
- Baseline (class in tensorforce.core.baselines.baseline), 84
- baseline_optimizer_arguments() (tensorforce.models.pg_log_prob_model.PGLogProbModel method), 201
- baseline_optimizer_arguments() (tensorforce.models.pg_model.PGModel method), 206
- baseline_optimizer_arguments() (tensorforce.models.pg_prob_ratio_model.PGProbRatioModel method), 211
- baseline_optimizer_arguments() (tensorforce.models.PGLogProbModel method), 261
- baseline_optimizer_arguments() (tensorforce.models.PGModel method), 249
- baseline_optimizer_arguments() (tensorforce.models.PGProbRatioModel method), 253
- BaseRunner (class in tensorforce.execution), 186
- BaseTest (class in tensorforce.tests.base_test), 283
- Bernoulli (class in tensorforce.core.distributions), 97
- Bernoulli (class in tensorforce.core.distributions.bernoulli), 92
- Beta (class in tensorforce.core.distributions), 99
- Beta (class in tensorforce.core.distributions.beta), 93
- build_metagraph_list() (tensorforce.meta_parameter_recorder.MetaParameterRecorder method), 357
- C**
- Categorical (class in tensorforce.core.distributions), 98
- Categorical (class in tensorforce.core.distributions.categorical), 94
- ClippedStep (class in tensorforce.core.optimizers), 174
- ClippedStep (class in tensorforce.core.optimizers.clipped_step), 151
- clone_worker_agent() (in module tensorforce.execution.threaded_runner), 185
- close() (tensorforce.agents.Agent method), 51
- close() (tensorforce.agents.agent.Agent method), 26
- close() (tensorforce.agents.constant_agent.ConstantAgent method), 28
- close() (tensorforce.agents.ConstantAgent method), 53
- close() (tensorforce.agents.DDPGAgent method), 73
- close() (tensorforce.agents.dqfd_agent.DQFDAGent method), 31
- close() (tensorforce.agents.DQFDAGent method), 59
- close() (tensorforce.agents.dqn_agent.DQNAGent method), 33
- close() (tensorforce.agents.dqn_nstep_agent.DQNNstepAgent method), 35
- close() (tensorforce.agents.DQNAGent method), 61
- close() (tensorforce.agents.DQNNstepAgent method), 63
- close() (tensorforce.agents.learning_agent.LearningAgent method), 38
- close() (tensorforce.agents.LearningAgent method), 56
- close() (tensorforce.agents.naf_agent.NAFAGent method), 40
- close() (tensorforce.agents.NAFAGent method), 65
- close() (tensorforce.agents.ppo_agent.PPOAGent method), 43
- close() (tensorforce.agents.PPOAGent method), 67
- close() (tensorforce.agents.random_agent.RandomAgent method), 45
- close() (tensorforce.agents.RandomAgent method), 54
- close() (tensorforce.agents.trpo_agent.TRPOAGent method), 47
- close() (tensorforce.agents.TRPOAGent method), 69
- close() (tensorforce.agents.vpg_agent.VPGAGent method), 49
- close() (tensorforce.agents.VPGAGent method), 71
- close() (tensorforce.contrib.ale.ALE method), 75
- close() (tensorforce.contrib.deepmind_lab.DeepMindLab method), 76
- close() (tensorforce.contrib.maze_explorer.MazeExplorer method), 77
- close() (tensorforce.contrib.openai_gym.OpenAIGym method), 77
- close() (tensorforce.contrib.openai_universe.OpenAIUniverse method), 78
- close() (tensorforce.contrib.remote_environment.RemoteEnvironment method), 79
- close() (tensorforce.contrib.state_settable_environment.StateSettableEnvironment method), 80
- close() (tensorforce.contrib.unreal_engine.UE4Environment method), 82
- close() (tensorforce.environments.Environment method), 182
- close() (tensorforce.environments.environment.Environment method), 181
- close() (tensorforce.environments.MinimalTest method), 183
- close() (tensorforce.execution.BaseRunner method), 186

- close() (tensorflow.execution.Runner method), 187
- close() (tensorflow.execution.runner.Runner method), 184
- close() (tensorflow.execution.threaded_runner.ThreadedRunner method), 185
- close() (tensorflow.execution.ThreadedRunner method), 188
- close() (tensorflow.models.constant_model.ConstantModel method), 189
- close() (tensorflow.models.distribution_model.DistributionModel method), 192
- close() (tensorflow.models.DistributionModel method), 244
- close() (tensorflow.models.DPGTargetModel method), 257
- close() (tensorflow.models.MemoryModel method), 239
- close() (tensorflow.models.Model method), 235
- close() (tensorflow.models.model.Model method), 198
- close() (tensorflow.models.pg_log_prob_model.PGLogProbModel method), 201
- close() (tensorflow.models.pg_model.PGModel method), 206
- close() (tensorflow.models.pg_prob_ratio_model.PGProbRatioModel method), 211
- close() (tensorflow.models.PGLogProbModel method), 262
- close() (tensorflow.models.PGModel method), 249
- close() (tensorflow.models.PGProbRatioModel method), 253
- close() (tensorflow.models.q_demo_model.QDemoModel method), 215
- close() (tensorflow.models.q_model.QModel method), 220
- close() (tensorflow.models.q_naf_model.QNAFModel method), 224
- close() (tensorflow.models.q_nstep_model.QNstepModel method), 228
- close() (tensorflow.models.QDemoModel method), 278
- close() (tensorflow.models.QModel method), 266
- close() (tensorflow.models.QNAFModel method), 274
- close() (tensorflow.models.QNstepModel method), 270
- close() (tensorflow.models.random_model.RandomModel method), 232
- CNNBaseline (class in tensorflow.core.baselines), 91
- CNNBaseline (class in tensorflow.core.baselines.cnn_baseline), 85
- ComplexLayeredNetwork (class in tensorflow.core.networks.complex_network), 114
- COMPONENT_BASELINE (tensorflow.models.pg_log_prob_model.PGLogProbModel attribute), 200
- COMPONENT_BASELINE (tensorflow.models.pg_model.PGModel attribute), 205
- COMPONENT_BASELINE (tensorflow.models.pg_prob_ratio_model.PGProbRatioModel attribute), 210
- COMPONENT_BASELINE (tensorflow.models.PGLogProbModel attribute), 261
- COMPONENT_BASELINE (tensorflow.models.PGModel attribute), 248
- COMPONENT_BASELINE (tensorflow.models.PGProbRatioModel attribute), 253
- COMPONENT_CRITIC (tensorflow.models.DPGTargetModel attribute), 257
- COMPONENT_DISTRIBUTION (tensorflow.models.distribution_model.DistributionModel attribute), 191
- COMPONENT_DISTRIBUTION (tensorflow.models.DistributionModel attribute), 244
- COMPONENT_DISTRIBUTION (tensorflow.models.DPGTargetModel attribute), 257
- COMPONENT_DISTRIBUTION (tensorflow.models.pg_log_prob_model.PGLogProbModel attribute), 201
- COMPONENT_DISTRIBUTION (tensorflow.models.pg_model.PGModel attribute), 205
- COMPONENT_DISTRIBUTION (tensorflow.models.pg_prob_ratio_model.PGProbRatioModel attribute), 210
- COMPONENT_DISTRIBUTION (tensorflow.models.PGLogProbModel attribute), 261
- COMPONENT_DISTRIBUTION (tensorflow.models.PGModel attribute), 248
- COMPONENT_DISTRIBUTION (tensorflow.models.PGProbRatioModel attribute), 253
- COMPONENT_DISTRIBUTION (tensorflow.models.q_demo_model.QDemoModel attribute), 214
- COMPONENT_DISTRIBUTION (tensorflow.models.q_model.QModel attribute), 219
- COMPONENT_DISTRIBUTION (tensorflow.models.q_naf_model.QNAFModel attribute), 223
- COMPONENT_DISTRIBUTION (tensorflow.models.q_nstep_model.QNstepModel attribute), 227
- COMPONENT_DISTRIBUTION (tensorflow.models.QDemoModel attribute), 278

- COMPONENT_DISTRIBUTION (tensorforce.models.QModel attribute), 265
- COMPONENT_DISTRIBUTION (tensorforce.models.QNAFModel attribute), 273
- COMPONENT_DISTRIBUTION (tensorforce.models.QNstepModel attribute), 269
- COMPONENT_NETWORK (tensorforce.models.distribution_model.DistributionModel attribute), 192
- COMPONENT_NETWORK (tensorforce.models.DistributionModel attribute), 244
- COMPONENT_NETWORK (tensorforce.models.DPGTargetModel attribute), 257
- COMPONENT_NETWORK (tensorforce.models.pg_log_prob_model.PGLogProbModel attribute), 201
- COMPONENT_NETWORK (tensorforce.models.pg_model.PGModel attribute), 205
- COMPONENT_NETWORK (tensorforce.models.pg_prob_ratio_model.PGProbRatioModel attribute), 210
- COMPONENT_NETWORK (tensorforce.models.PGLogProbModel attribute), 261
- COMPONENT_NETWORK (tensorforce.models.PGModel attribute), 248
- COMPONENT_NETWORK (tensorforce.models.PGProbRatioModel attribute), 253
- COMPONENT_NETWORK (tensorforce.models.q_demo_model.QDemoModel attribute), 214
- COMPONENT_NETWORK (tensorforce.models.q_model.QModel attribute), 219
- COMPONENT_NETWORK (tensorforce.models.q_naf_model.QNAFModel attribute), 223
- COMPONENT_NETWORK (tensorforce.models.q_nstep_model.QNstepModel attribute), 228
- COMPONENT_NETWORK (tensorforce.models.QDemoModel attribute), 278
- COMPONENT_NETWORK (tensorforce.models.QModel attribute), 265
- COMPONENT_NETWORK (tensorforce.models.QNAFModel attribute), 273
- COMPONENT_NETWORK (tensorforce.models.QNstepModel attribute), 269
- COMPONENT_TARGET_DISTRIBUTION (tensorforce.models.DPGTargetModel attribute), 257
- COMPONENT_TARGET_DISTRIBUTION (tensorforce.models.q_demo_model.QDemoModel attribute), 214
- COMPONENT_TARGET_DISTRIBUTION (tensorforce.models.q_model.QModel attribute), 219
- COMPONENT_TARGET_DISTRIBUTION (tensorforce.models.q_naf_model.QNAFModel attribute), 223
- COMPONENT_TARGET_DISTRIBUTION (tensorforce.models.q_nstep_model.QNstepModel attribute), 228
- COMPONENT_TARGET_DISTRIBUTION (tensorforce.models.QDemoModel attribute), 278
- COMPONENT_TARGET_DISTRIBUTION (tensorforce.models.QModel attribute), 265
- COMPONENT_TARGET_DISTRIBUTION (tensorforce.models.QNAFModel attribute), 273
- COMPONENT_TARGET_DISTRIBUTION (tensorforce.models.QNstepModel attribute), 269
- config (tensorforce.tests.base_agent_test.BaseAgentTest attribute), 283
- config (tensorforce.tests.test_constant_agent.TestConstantAgent attribute), 288
- config (tensorforce.tests.test_dqfd_agent.TestDQFDAgent attribute), 293
- config (tensorforce.tests.test_dqn_agent.TestDQNAgent attribute), 299
- config (tensorforce.tests.test_dqn_nstep_agent.TestDQNNstepAgent attribute), 304

[config \(tensorflow.tests.test_naf_agent.TestNAFAgent attribute\), 310](#)
[config \(tensorflow.tests.test_ppo_agent.TestPPOAgent attribute\), 315](#)
[config \(tensorflow.tests.test_random_agent.TestRandomAgent attribute\), 325](#)
[config \(tensorflow.tests.test_trpo_agent.TestTRPOAgent attribute\), 335](#)
[config \(tensorflow.tests.test_vpg_agent.TestVPGAgent attribute\), 345](#)
[configure\(\) \(tensorflow.contrib.openai_universe.OpenAIUniverse method\), 78](#)
[ConjugateGradient \(class in tensorflow.core.optimizers.solvers\), 148](#)
[ConjugateGradient \(class in tensorflow.core.optimizers.solvers.conjugate_gradient\), 142](#)
[connect\(\) \(tensorflow.contrib.remote_environment.RemoteEnvironment method\), 80](#)
[connect\(\) \(tensorflow.contrib.unreal_engine.UE4Environment method\), 82](#)
[Constant \(class in tensorflow.core.explorations\), 102](#)
[Constant \(class in tensorflow.core.explorations.constant\), 100](#)
[ConstantAgent \(class in tensorflow.agents\), 52](#)
[ConstantAgent \(class in tensorflow.agents.constant_agent\), 27](#)
[ConstantModel \(class in tensorflow.models.constant_model\), 188](#)
[Conv1d \(class in tensorflow.core.networks\), 137](#)
[Conv1d \(class in tensorflow.core.networks.layer\), 116](#)
[Conv2d \(class in tensorflow.core.networks\), 137](#)
[Conv2d \(class in tensorflow.core.networks.layer\), 117](#)
[convert_data_to_string\(\) \(tensorflow.meta_parameter_recorder.MetaParameterRecorder method\), 357](#)
[convert_dictionary_to_string\(\) \(tensorflow.meta_parameter_recorder.MetaParameterRecorder method\), 357](#)
[convert_list_to_string\(\) \(tensorflow.meta_parameter_recorder.MetaParameterRecorder method\), 357](#)
[convert_ndarray_to_md\(\) \(tensorflow.meta_parameter_recorder.MetaParameterRecorder method\), 357](#)
[countTestCases\(\) \(tensorflow.tests.test_constant_agent.TestConstantAgent method\), 288](#)
[countTestCases\(\) \(tensorflow.tests.test_dqfd_agent.TestDQFDAGent method\), 293](#)
[countTestCases\(\) \(tensorflow.tests.test_dqn_agent.TestDQNAgent method\), 299](#)
[countTestCases\(\) \(tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent method\), 304](#)
[countTestCases\(\) \(tensorflow.tests.test_naf_agent.TestNAFAgent method\), 310](#)
[countTestCases\(\) \(tensorflow.tests.test_ppo_agent.TestPPOAgent method\), 315](#)
[countTestCases\(\) \(tensorflow.tests.test_quickstart_example.TestQuickstartExample method\), 320](#)
[countTestCases\(\) \(tensorflow.tests.test_random_agent.TestRandomAgent method\), 325](#)
[countTestCases\(\) \(tensorflow.tests.test_reward_estimation.TestRewardEstimation method\), 330](#)
[countTestCases\(\) \(tensorflow.tests.test_trpo_agent.TestTRPOAgent method\), 335](#)
[countTestCases\(\) \(tensorflow.tests.test_tutorial_code.TestTutorialCode method\), 340](#)
[countTestCases\(\) \(tensorflow.tests.test_vpg_agent.TestVPGAgent method\), 345](#)
[countTestCases\(\) \(tensorflow.tests.test_vpg_baselines.TestVPGBaselines method\), 350](#)
[countTestCases\(\) \(tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers method\), 355](#)
[create_act_operations\(\) \(tensorflow.models.constant_model.ConstantModel method\), 189](#)
[create_act_operations\(\) \(tensorflow.models.distribution_model.DistributionModel method\), 192](#)
[create_act_operations\(\) \(tensorflow.models.DistributionModel method\), 244](#)
[create_act_operations\(\) \(tensorflow.models.DPGTargetModel method\), 257](#)
[create_act_operations\(\) \(tensorflow.models.MemoryModel method\), 239](#)
[create_act_operations\(\) \(tensorflow.models.Model method\), 235](#)
[create_act_operations\(\) \(tensorflow.models.model.Model method\), 198](#)
[create_act_operations\(\) \(tensorflow.models.pg_log_prob_model.PGLogProbModel method\), 201](#)

`create_act_operations()` (tensorflow.models.pg_model.PGModel method), 211
`create_act_operations()` (tensorflow.models.pg_prob_ratio_model.PGProbRatioModel method), 211
`create_act_operations()` (tensorflow.models.PGLogProbModel method), 262
`create_act_operations()` (tensorflow.models.PGModel method), 249
`create_act_operations()` (tensorflow.models.PGProbRatioModel method), 253
`create_act_operations()` (tensorflow.models.q_demo_model.QDemoModel method), 215
`create_act_operations()` (tensorflow.models.q_model.QModel method), 220
`create_act_operations()` (tensorflow.models.q_naf_model.QNAFModel method), 224
`create_act_operations()` (tensorflow.models.q_nstep_model.QNstepModel method), 228
`create_act_operations()` (tensorflow.models.QDemoModel method), 278
`create_act_operations()` (tensorflow.models.QModel method), 266
`create_act_operations()` (tensorflow.models.QNAFModel method), 274
`create_act_operations()` (tensorflow.models.QNstepModel method), 270
`create_act_operations()` (tensorflow.models.random_model.RandomModel method), 232
`create_distributions()` (tensorflow.models.distribution_model.DistributionModel method), 192
`create_distributions()` (tensorflow.models.DistributionModel method), 244
`create_distributions()` (tensorflow.models.DPGTargetModel method), 257
`create_distributions()` (tensorflow.models.pg_log_prob_model.PGLogProbModel method), 201
`create_distributions()` (tensorflow.models.pg_model.PGModel method), 206
`create_distributions()` (tensorflow.models.pg_prob_ratio_model.PGProbRatioModel method), 206
`create_distributions()` (tensorflow.models.PGLogProbModel method), 262
`create_distributions()` (tensorflow.models.PGModel method), 249
`create_distributions()` (tensorflow.models.PGProbRatioModel method), 254
`create_distributions()` (tensorflow.models.q_demo_model.QDemoModel method), 215
`create_distributions()` (tensorflow.models.q_model.QModel method), 220
`create_distributions()` (tensorflow.models.q_naf_model.QNAFModel method), 224
`create_distributions()` (tensorflow.models.q_nstep_model.QNstepModel method), 228
`create_distributions()` (tensorflow.models.QDemoModel method), 278
`create_distributions()` (tensorflow.models.QModel method), 266
`create_distributions()` (tensorflow.models.QNAFModel method), 274
`create_distributions()` (tensorflow.models.QNstepModel method), 270
`create_observe_operations()` (tensorflow.models.constant_model.ConstantModel method), 189
`create_observe_operations()` (tensorflow.models.distribution_model.DistributionModel method), 192
`create_observe_operations()` (tensorflow.models.DistributionModel method), 244
`create_observe_operations()` (tensorflow.models.DPGTargetModel method), 257
`create_observe_operations()` (tensorflow.models.MemoryModel method), 239
`create_observe_operations()` (tensorflow.models.Model method), 235
`create_observe_operations()` (tensorflow.models.model.Model method), 198
`create_observe_operations()` (tensorflow.models.pg_log_prob_model.PGLogProbModel method), 201
`create_observe_operations()` (tensorflow.models.pg_model.PGModel method), 206
`create_observe_operations()` (tensorflow.models.pg_prob_ratio_model.PGProbRatioModel method), 206

[debug\(\)](#) (tensorforce.tests.test_naf_agent.TestNAFAgent method), 340
[defaultTestResult\(\)](#) (tensorforce.tests.test_constant_agent.TestConstantAgent method), 288
[defaultTestResult\(\)](#) (tensorforce.tests.test_dqfd_agent.TestDQFDAgent method), 293
[defaultTestResult\(\)](#) (tensorforce.tests.test_dqn_agent.TestDQNAgent method), 299
[defaultTestResult\(\)](#) (tensorforce.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 304
[defaultTestResult\(\)](#) (tensorforce.tests.test_naf_agent.TestNAFAgent method), 310
[defaultTestResult\(\)](#) (tensorforce.tests.test_ppo_agent.TestPPOAgent method), 315
[defaultTestResult\(\)](#) (tensorforce.tests.test_quickstart_example.TestQuickstartExample method), 320
[defaultTestResult\(\)](#) (tensorforce.tests.test_random_agent.TestRandomAgent method), 325
[defaultTestResult\(\)](#) (tensorforce.tests.test_reward_estimation.TestRewardEstimation method), 330
[defaultTestResult\(\)](#) (tensorforce.tests.test_trpo_agent.TestTRPOAgent method), 335
[defaultTestResult\(\)](#) (tensorforce.tests.test_tutorial_code.TestTutorialCode method), 340
[demo_update\(\)](#) (tensorforce.models.q_demo_model.QDemoModel method), 215
[Dense](#) (class in tensorforce.core.networks), 135
[Dense](#) (class in tensorforce.core.networks.layer), 118
[disconnect\(\)](#) (tensorforce.contrib.remote_environment.RemoteEnvironment method), 80
[disconnect\(\)](#) (tensorforce.contrib.unreal_engine.UE4Environment method), 82
[discretize_action_space_desc\(\)](#) (tensorforce.contrib.unreal_engine.UE4Environment method), 82
[DistributedTFRunner](#) (in module tensorforce.execution), 187
[DistributedTFRunner](#) (in module tensorforce.execution.runner), 183
[Distribution](#) (class in tensorforce.core.distributions), 96
[Distribution](#) (class in tensorforce.core.distributions.distribution), 94
[DistributionModel](#) (class in tensorforce.models), 243
[DistributionModel](#) (class in tensorforce.models.distribution_model), 191
[doCleanups\(\)](#) (tensorforce.tests.test_constant_agent.TestConstantAgent method), 288
[doCleanups\(\)](#) (tensorforce.tests.test_dqfd_agent.TestDQFDAgent method), 293
[doCleanups\(\)](#) (tensorforce.tests.test_dqn_agent.TestDQNAgent method), 299
[doCleanups\(\)](#) (tensorforce.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 304
[doCleanups\(\)](#) (tensorforce.tests.test_naf_agent.TestNAFAgent method), 310
[doCleanups\(\)](#) (tensorforce.tests.test_ppo_agent.TestPPOAgent method), 315
[doCleanups\(\)](#) (tensorforce.tests.test_quickstart_example.TestQuickstartExample method), 320
[doCleanups\(\)](#) (tensorforce.tests.test_random_agent.TestRandomAgent method), 325
[doCleanups\(\)](#) (tensorforce.tests.test_reward_estimation.TestRewardEstimation method), 330
[doCleanups\(\)](#) (tensorforce.tests.test_trpo_agent.TestTRPOAgent method), 335
[doCleanups\(\)](#) (tensorforce.tests.test_tutorial_code.TestTutorialCode method), 340

- doCleanups() (tensorflow.tests.test_tutorial_code.TestTutorialCode method), 340
- doCleanups() (tensorflow.tests.test_vpg_agent.TestVPGAgent method), 345
- doCleanups() (tensorflow.tests.test_vpg_baselines.TestVPGBaselinesforce.core.explorations.epsilon_decay), 101
- doCleanups() (tensorflow.tests.test_vpg_optimizers.TestVPGOptimizersforce.core.optimizers.evolutionary), 171
- DPGTargetModel (class in tensorflow.models), 257
- DQFDAgent (class in tensorflow.agents), 57
- DQFDAgent (class in tensorflow.agents.dqfd_agent), 30
- DQNAgent (class in tensorflow.agents), 60
- DQNAgent (class in tensorflow.agents.dqn_agent), 32
- DQNNstepAgent (class in tensorflow.agents.dqn_nstep_agent), 34
- Dropout (class in tensorflow.core.networks), 131
- Dropout (class in tensorflow.core.networks.layer), 118
- Dueling (class in tensorflow.core.networks), 136
- Dueling (class in tensorflow.core.networks.layer), 119
- ## E
- Embedding (class in tensorflow.core.networks), 133
- Embedding (class in tensorflow.core.networks.layer), 120
- Environment (class in tensorflow.environments), 182
- Environment (class in tensorflow.environments.environment), 181
- environments (tensorflow.execution.threaded_runner.ThreadedRunner attribute), 185
- environments (tensorflow.execution.ThreadedRunner attribute), 188
- episode (tensorflow.execution.BaseRunner attribute), 186
- episode (tensorflow.execution.Runner attribute), 187
- episode (tensorflow.execution.runner.Runner attribute), 184
- episode (tensorflow.execution.threaded_runner.ThreadedRunner attribute), 185
- episode (tensorflow.execution.ThreadedRunner attribute), 188
- episode_lengths (tensorflow.execution.threaded_runner.ThreadedRunner attribute), 185
- episode_lengths (tensorflow.execution.ThreadedRunner attribute), 188
- episode_timestep (tensorflow.execution.Runner attribute), 187
- episode_timestep (tensorflow.execution.runner.Runner attribute), 184
- EpsilonAnneal (class in tensorflow.core.explorations), 103
- EpsilonAnneal (class in tensorflow.core.explorations.epsilon_anneal), 103
- EpsilonDecay (class in tensorflow.core.explorations), 103
- EpsilonDecay (class in tensorflow.core.explorations.epsilon_decay), 101
- Evolutionary (class in tensorflow.core.optimizers), 171
- Evolutionary (class in tensorflow.core.optimizers.evolutionary), 153
- exclude_bool (tensorflow.tests.base_agent_test.BaseAgentTest attribute), 283
- exclude_bool (tensorflow.tests.test_constant_agent.TestConstantAgent attribute), 288
- exclude_bool (tensorflow.tests.test_dqfd_agent.TestDQFDAgent attribute), 293
- exclude_bool (tensorflow.tests.test_dqn_agent.TestDQNAgent attribute), 299
- exclude_bool (tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent attribute), 304
- exclude_bool (tensorflow.tests.test_naf_agent.TestNAFAgent attribute), 310
- exclude_bool (tensorflow.tests.test_ppo_agent.TestPPOAgent attribute), 315
- exclude_bool (tensorflow.tests.test_random_agent.TestRandomAgent attribute), 325
- exclude_bool (tensorflow.tests.test_trpo_agent.TestTRPOAgent attribute), 335
- exclude_bounded (tensorflow.tests.base_agent_test.BaseAgentTest attribute), 283
- exclude_bounded (tensorflow.tests.test_constant_agent.TestConstantAgent attribute), 288
- exclude_bounded (tensorflow.tests.test_dqfd_agent.TestDQFDAgent attribute), 293
- exclude_bounded (tensorflow.tests.test_dqn_agent.TestDQNAgent attribute), 299
- exclude_bounded (tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent attribute), 304
- exclude_bounded (tensorflow.tests.test_naf_agent.TestNAFAgent attribute), 310
- exclude_bounded (tensorflow.tests.test_ppo_agent.TestPPOAgent attribute), 315
- exclude_bounded (tensorflow.tests.test_random_agent.TestRandomAgent attribute), 325
- exclude_bounded (tensorflow.tests.test_trpo_agent.TestTRPOAgent attribute), 335

attribute), 335

exclude_bounded (tensorflow.tests.test_vpg_agent.TestVPGAgent attribute), 345

exclude_float (tensorflow.tests.base_agent_test.BaseAgentTest attribute), 283

exclude_float (tensorflow.tests.test_constant_agent.TestConstantAgent attribute), 288

exclude_float (tensorflow.tests.test_dqfd_agent.TestDQFDAGent attribute), 293

exclude_float (tensorflow.tests.test_dqn_agent.TestDQNAgent attribute), 299

exclude_float (tensorflow.tests.test_dqn_nstep_agent.TestDQNNStepAgent attribute), 304

exclude_float (tensorflow.tests.test_naf_agent.TestNAFAgent attribute), 310

exclude_float (tensorflow.tests.test_ppo_agent.TestPPOAgent attribute), 315

exclude_float (tensorflow.tests.test_random_agent.TestRandomAgent attribute), 325

exclude_float (tensorflow.tests.test_trpo_agent.TestTRPOAgent attribute), 335

exclude_float (tensorflow.tests.test_vpg_agent.TestVPGAgent attribute), 345

exclude_int (tensorflow.tests.base_agent_test.BaseAgentTest attribute), 283

exclude_int (tensorflow.tests.test_constant_agent.TestConstantAgent attribute), 288

exclude_int (tensorflow.tests.test_dqfd_agent.TestDQFDAGent attribute), 294

exclude_int (tensorflow.tests.test_dqn_agent.TestDQNAgent attribute), 299

exclude_int (tensorflow.tests.test_dqn_nstep_agent.TestDQNNStepAgent attribute), 304

exclude_int (tensorflow.tests.test_naf_agent.TestNAFAgent attribute), 310

exclude_int (tensorflow.tests.test_ppo_agent.TestPPOAgent attribute), 315

exclude_int (tensorflow.tests.test_random_agent.TestRandomAgent attribute), 325

exclude_int (tensorflow.tests.test_trpo_agent.TestTRPOAgent attribute), 335

exclude_int (tensorflow.tests.test_vpg_agent.TestVPGAgent attribute), 345

exclude_lstm (tensorflow.tests.base_agent_test.BaseAgentTest attribute), 283

exclude_lstm (tensorflow.tests.test_constant_agent.TestConstantAgent attribute), 288

exclude_lstm (tensorflow.tests.test_dqfd_agent.TestDQFDAGent attribute), 294

exclude_lstm (tensorflow.tests.test_dqn_agent.TestDQNAgent attribute), 299

exclude_lstm (tensorflow.tests.test_dqn_nstep_agent.TestDQNNStepAgent attribute), 304

exclude_lstm (tensorflow.tests.test_naf_agent.TestNAFAgent attribute), 310

exclude_lstm (tensorflow.tests.test_ppo_agent.TestPPOAgent attribute), 315

exclude_lstm (tensorflow.tests.test_random_agent.TestRandomAgent attribute), 325

exclude_lstm (tensorflow.tests.test_trpo_agent.TestTRPOAgent attribute), 335

exclude_lstm (tensorflow.tests.test_vpg_agent.TestVPGAgent attribute), 345

exclude_multi (tensorflow.tests.base_agent_test.BaseAgentTest attribute), 283

exclude_multi (tensorflow.tests.test_dqfd_agent.TestDQFDAGent attribute), 294

exclude_multi (tensorflow.tests.test_dqn_agent.TestDQNAgent attribute), 299

exclude_multi (tensorflow.tests.test_dqn_nstep_agent.TestDQNNStepAgent attribute), 304

exclude_multi (tensorflow.tests.test_naf_agent.TestNAFAgent attribute), 310

exclude_multi (tensorflow.tests.test_ppo_agent.TestPPOAgent attribute), 315

exclude_multi (tensorflow.tests.test_random_agent.TestRandomAgent attribute), 325

exclude_multi (tensorflow.tests.test_trpo_agent.TestTRPOAgent attribute), 335

exclude_multi (tensorflow.tests.test_vpg_agent.TestVPGAgent attribute), 345

execute() (tensorflow.contrib.ale.ALE method), 75

execute() (tensorflow.contrib.deepmind_lab.DeepMindLab method), 76

execute() (tensorflow.contrib.maze_explorer.MazeExplorer method), 77

execute() (tensorflow.contrib.openai_gym.OpenAIGym method), 77

execute() (tensorflow.contrib.openai_universe.OpenAIUniverse method), 78

execute() (tensorflow.contrib.remote_environment.RemoteEnvironment method), 80

execute() (tensorflow.contrib.state_settable_environment.StateSettableEnvironment method), 80

execute() (tensorflow.contrib.unreal_engine.UE4Environment method), 82

execute() (tensorflow.environments.Environment method), 182

execute() (tensorflow.environments.environment.Environment method), 181

execute() (tensorflow.environments.MinimalTest method), 183

execute() (tensorflow.tests.test_tutorial_code.TestTutorialCode.MyClient method), 336

Exploration (class in tensorflow.core.explorations), 102

Exploration (class in tensorflow.core.explorations.exploration), 101

extract_observation() (tensorflow.contrib.unreal_engine.UEnv4Environment static method), 82

F

fail() (tensorflow.tests.test_constant_agent.TestConstantAgent method), 288

fail() (tensorflow.tests.test_dqfd_agent.TestDQFDAGent method), 294

fail() (tensorflow.tests.test_dqn_agent.TestDQNAgent method), 299

fail() (tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 305

fail() (tensorflow.tests.test_naf_agent.TestNAFAGent method), 310

fail() (tensorflow.tests.test_ppo_agent.TestPPOAgent method), 315

fail() (tensorflow.tests.test_quickstart_example.TestQuickstartExample method), 320

fail() (tensorflow.tests.test_random_agent.TestRandomAgent method), 325

fail() (tensorflow.tests.test_reward_estimation.TestRewardEstimation method), 330

fail() (tensorflow.tests.test_trpo_agent.TestTRPOAgent method), 335

fail() (tensorflow.tests.test_tutorial_code.TestTutorialCode method), 340

fail() (tensorflow.tests.test_vpg_agent.TestVPGAgent method), 345

fail() (tensorflow.tests.test_vpg_baselines.TestVPGBaselines method), 350

fail() (tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers method), 355

failIf() (tensorflow.tests.test_constant_agent.TestConstantAgent method), 288

failIf() (tensorflow.tests.test_dqfd_agent.TestDQFDAGent method), 294

failIf() (tensorflow.tests.test_dqn_agent.TestDQNAgent method), 299

failIf() (tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 305

failIf() (tensorflow.tests.test_naf_agent.TestNAFAGent method), 310

failIf() (tensorflow.tests.test_ppo_agent.TestPPOAgent method), 315

failIf() (tensorflow.tests.test_quickstart_example.TestQuickstartExample method), 320

failIf() (tensorflow.tests.test_random_agent.TestRandomAgent method), 325

failIf() (tensorflow.tests.test_reward_estimation.TestRewardEstimation method), 330

failIf() (tensorflow.tests.test_trpo_agent.TestTRPOAgent method), 335

failIf() (tensorflow.tests.test_tutorial_code.TestTutorialCode method), 340

failIf() (tensorflow.tests.test_vpg_agent.TestVPGAgent method), 345

failIf() (tensorflow.tests.test_vpg_baselines.TestVPGBaselines method), 350

failIf() (tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers method), 356

failIfAlmostEqual() (tensorflow.tests.test_constant_agent.TestConstantAgent method), 288

failIfAlmostEqual() (tensorflow.tests.test_dqfd_agent.TestDQFDAGent method), 294

failIfAlmostEqual() (tensorflow.tests.test_dqn_agent.TestDQNAgent method), 299

failIfAlmostEqual() (tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 305

failIfAlmostEqual() (tensorflow.tests.test_naf_agent.TestNAFAGent method), 310

failIfAlmostEqual() (tensorflow.tests.test_ppo_agent.TestPPOAgent method), 315

failIfAlmostEqual() (tensorflow.tests.test_quickstart_example.TestQuickstartExample method), 320

failIfAlmostEqual() (tensorflow.tests.test_random_agent.TestRandomAgent method), 325

failIfAlmostEqual() (tensorflow.tests.test_reward_estimation.TestRewardEstimation method), 330

failIfAlmostEqual() (tensorflow.tests.test_trpo_agent.TestTRPOAgent method), 335

failIfAlmostEqual() (tensorflow.tests.test_tutorial_code.TestTutorialCode method), 340

failIfAlmostEqual() (tensorflow.tests.test_vpg_agent.TestVPGAgent method), 345

failIfAlmostEqual() (tensorflow.tests.test_vpg_baselines.TestVPGBaselines method), 350

failIfAlmostEqual() (tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers method), 356

failIfEqual() (tensorflow.tests.test_constant_agent.TestConstantAgent method), 288

`failIfEqual()` (tensorforce.tests.test_dqfd_agent.TestDQFDAGent method), 294
`failUnlessAlmostEqual()` (tensorforce.tests.test_constant_agent.TestConstantAgent method), 288
`failIfEqual()` (tensorforce.tests.test_dqn_agent.TestDQNAgent method), 299
`failUnlessAlmostEqual()` (tensorforce.tests.test_dqfd_agent.TestDQFDAGent method), 294
`failIfEqual()` (tensorforce.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 305
`failUnlessAlmostEqual()` (tensorforce.tests.test_naf_agent.TestNAFAGent method), 310
`failIfEqual()` (tensorforce.tests.test_ppo_agent.TestPPOAgent method), 316
`failUnlessAlmostEqual()` (tensorforce.tests.test_quickstart_example.TestQuickstartExample method), 320
`failIfEqual()` (tensorforce.tests.test_random_agent.TestRandomAgent method), 325
`failUnlessAlmostEqual()` (tensorforce.tests.test_naf_agent.TestNAFAGent method), 310
`failIfEqual()` (tensorforce.tests.test_reward_estimation.TestRewardEstimation method), 330
`failUnlessAlmostEqual()` (tensorforce.tests.test_trpo_agent.TestTRPOAgent method), 335
`failIfEqual()` (tensorforce.tests.test_ppo_agent.TestPPOAgent method), 316
`failUnlessAlmostEqual()` (tensorforce.tests.test_tutorial_code.TestTutorialCode method), 340
`failIfEqual()` (tensorforce.tests.test_vpg_agent.TestVPGAgent method), 345
`failUnlessAlmostEqual()` (tensorforce.tests.test_vpg_optimizers.TestVPGOptimizers method), 356
`failUnless()` (tensorforce.tests.test_constant_agent.TestConstantAgent method), 288
`failUnlessAlmostEqual()` (tensorforce.tests.test_dqfd_agent.TestDQFDAGent method), 294
`failUnless()` (tensorforce.tests.test_dqn_agent.TestDQNAgent method), 299
`failUnlessAlmostEqual()` (tensorforce.tests.test_tutorial_code.TestTutorialCode method), 340
`failUnless()` (tensorforce.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 305
`failUnlessAlmostEqual()` (tensorforce.tests.test_naf_agent.TestNAFAGent method), 310
`failUnless()` (tensorforce.tests.test_ppo_agent.TestPPOAgent method), 316
`failUnlessAlmostEqual()` (tensorforce.tests.test_vpg_optimizers.TestVPGOptimizers method), 356
`failUnless()` (tensorforce.tests.test_quickstart_example.TestQuickstartExample method), 320
`failUnlessAlmostEqual()` (tensorforce.tests.test_random_agent.TestRandomAgent method), 325
`failUnless()` (tensorforce.tests.test_reward_estimation.TestRewardEstimation method), 330
`failUnlessAlmostEqual()` (tensorforce.tests.test_trpo_agent.TestTRPOAgent method), 335
`failUnlessEqual()` (tensorforce.tests.test_dqfd_agent.TestDQFDAGent method), 294
`failUnless()` (tensorforce.tests.test_vpg_agent.TestVPGAgent method), 345
`failUnlessEqual()` (tensorforce.tests.test_dqn_agent.TestDQNAgent method), 299
`failUnless()` (tensorforce.tests.test_vpg_optimizers.TestVPGOptimizers method), 356
`failUnlessEqual()` (tensorforce.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 305

failUnlessEqual() force.tests.test_naf_agent.TestNAFAgent method), 310	(tensor-	failUnlessRaises() force.tests.test_reward_estimation.TestRewardEstimation method), 330	(tensor-
failUnlessEqual() force.tests.test_ppo_agent.TestPPOAgent method), 316	(tensor-	failUnlessRaises() force.tests.test_trpo_agent.TestTRPOAgent method), 335	(tensor-
failUnlessEqual() force.tests.test_quickstart_example.TestQuickstartExample method), 320	(tensor-	failUnlessRaises() force.tests.test_tutorial_code.TestTutorialCode method), 340	(tensor-
failUnlessEqual() force.tests.test_random_agent.TestRandomAgent method), 325	(tensor-	failUnlessRaises() force.tests.test_vpg_agent.TestVPGAgent method), 345	(tensor-
failUnlessEqual() force.tests.test_reward_estimation.TestRewardEstimation method), 330	(tensor-	failUnlessRaises() force.tests.test_vpg_baselines.TestVPGBaselines method), 351	(tensor-
failUnlessEqual() force.tests.test_trpo_agent.TestTRPOAgent method), 335	(tensor-	failUnlessRaises() force.tests.test_vpg_optimizers.TestVPGOptimizers method), 356	(tensor-
failUnlessEqual() force.tests.test_tutorial_code.TestTutorialCode method), 340	(tensor-	failureException force.tests.test_constant_agent.TestConstantAgent attribute), 288	(tensor-
failUnlessEqual() force.tests.test_vpg_agent.TestVPGAgent method), 345	(tensor-	failureException force.tests.test_dqfd_agent.TestDQFDAgent attribute), 294	(tensor-
failUnlessEqual() force.tests.test_vpg_baselines.TestVPGBaselines method), 351	(tensor-	failureException force.tests.test_dqn_agent.TestDQNAgent attribute), 299	(tensor-
failUnlessEqual() force.tests.test_vpg_optimizers.TestVPGOptimizers method), 356	(tensor-	failureException force.tests.test_dqn_nstep_agent.TestDQNNstepAgent attribute), 305	(tensor-
failUnlessRaises() force.tests.test_constant_agent.TestConstantAgent method), 288	(tensor-	failureException force.tests.test_naf_agent.TestNAFAgent attribute), 310	(tensor-
failUnlessRaises() force.tests.test_dqfd_agent.TestDQFDAgent method), 294	(tensor-	failureException force.tests.test_ppo_agent.TestPPOAgent attribute), 316	(tensor-
failUnlessRaises() force.tests.test_dqn_agent.TestDQNAgent method), 299	(tensor-	failureException force.tests.test_quickstart_example.TestQuickstartExample attribute), 320	(tensor-
failUnlessRaises() force.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 305	(tensor-	failureException force.tests.test_random_agent.TestRandomAgent attribute), 325	(tensor-
failUnlessRaises() force.tests.test_naf_agent.TestNAFAgent method), 310	(tensor-	failureException force.tests.test_reward_estimation.TestRewardEstimation attribute), 330	(tensor-
failUnlessRaises() force.tests.test_ppo_agent.TestPPOAgent method), 316	(tensor-	failureException force.tests.test_trpo_agent.TestTRPOAgent attribute), 335	(tensor-
failUnlessRaises() force.tests.test_quickstart_example.TestQuickstartExample method), 320	(tensor-	failureException force.tests.test_tutorial_code.TestTutorialCode attribute), 340	(tensor-
failUnlessRaises() force.tests.test_random_agent.TestRandomAgent method), 325	(tensor-	failureException force.tests.test_vpg_agent.TestVPGAgent attribute), 345	(tensor-

failureException (tensorforce.tests.test_vpg_baselines.TestVPGBaselines attribute), 351
 failureException (tensorforce.tests.test_vpg_optimizers.TestVPGOptimizers attribute), 356
 Flatten (class in tensorforce.core.networks), 132
 Flatten (class in tensorforce.core.networks.layer), 121
 fps (tensorforce.contrib.deepmind_lab.DeepMindLab attribute), 76
 from_config() (tensorforce.core.optimizers.solvers.conjugate_gradient.ConjugateGradient method), 143
 from_config() (tensorforce.core.optimizers.solvers.ConjugateGradient method), 149
 from_config() (tensorforce.core.optimizers.solvers.Iterative method), 147
 from_config() (tensorforce.core.optimizers.solvers.iterative.Iterative method), 144
 from_config() (tensorforce.core.optimizers.solvers.line_search.LineSearch method), 145
 from_config() (tensorforce.core.optimizers.solvers.LineSearch method), 150
 from_config() (tensorforce.core.optimizers.solvers.Solver static method), 147
 from_config() (tensorforce.core.optimizers.solvers.solver.Solver static method), 147
 from_json() (tensorforce.core.networks.complex_network.ComplexLayeredNetwork static method), 114
 from_json() (tensorforce.core.networks.LayeredNetwork static method), 141
 from_json() (tensorforce.core.networks.network.LayeredNetwork static method), 128
 from_spec() (tensorforce.agents.Agent static method), 51
 from_spec() (tensorforce.agents.agent.Agent static method), 26
 from_spec() (tensorforce.agents.constant_agent.ConstantAgent method), 28
 from_spec() (tensorforce.agents.ConstantAgent method), 53
 from_spec() (tensorforce.agents.DDPGAgent method), 73
 from_spec() (tensorforce.agents.dqfd_agent.DQFDAGent method), 31
 from_spec() (tensorforce.agents.DQFDAGent method), 59
 from_spec() (tensorforce.agents.dqn_agent.DQNAGent method), 33
 from_spec() (tensorforce.agents.dqn_nstep_agent.DQNNstepAgent method), 35
 from_spec() (tensorforce.agents.DQNAGent method), 61
 from_spec() (tensorforce.agents.DQNNstepAgent method), 63
 from_spec() (tensorforce.agents.learning_agent.LearningAgent method), 38
 from_spec() (tensorforce.agents.LearningAgent method), 56
 from_spec() (tensorforce.agents.naf_agent.NAFAGent method), 40
 from_spec() (tensorforce.agents.NAFAGent method), 65
 from_spec() (tensorforce.agents.ppo_agent.PPOAGent method), 43
 from_spec() (tensorforce.agents.PPOAGent method), 67
 from_spec() (tensorforce.agents.random_agent.RandomAgent method), 45
 from_spec() (tensorforce.agents.RandomAgent method), 54
 from_spec() (tensorforce.agents.trpo_agent.TRPOAGent method), 47
 from_spec() (tensorforce.agents.TRPOAGent method), 69
 from_spec() (tensorforce.agents.vpg_agent.VPGAGent method), 49
 from_spec() (tensorforce.agents.VPGAGent method), 71
 from_spec() (tensorforce.contrib.ale.ALE method), 75
 from_spec() (tensorforce.contrib.deepmind_lab.DeepMindLab method), 76
 from_spec() (tensorforce.contrib.maze_explorer.MazeExplorer method), 77
 from_spec() (tensorforce.contrib.openai_gym.OpenAIGym method), 78
 from_spec() (tensorforce.contrib.openai_universe.OpenAIUniverse method), 78
 from_spec() (tensorforce.contrib.remote_environment.RemoteEnvironment method), 80
 from_spec() (tensorforce.contrib.state_settable_environment.StateSettableEnvironment method), 81
 from_spec() (tensorforce.contrib.unreal_engine.UE4Environment method), 82
 from_spec() (tensorforce.core.baselines.aggregated_baseline.AggregatedBaseline method), 83
 from_spec() (tensorforce.core.baselines.AggregatedBaseline method), 89
 from_spec() (tensorforce.core.baselines.Baseline static method), 88
 from_spec() (tensorforce.core.baselines.baseline.Baseline static method), 84
 from_spec() (tensorforce.core.baselines.cnn_baseline.CNNBaseline method), 85
 from_spec() (tensorforce.core.baselines.CNNBaseline method), 91
 from_spec() (tensorforce.core.baselines.mlp_baseline.MLPBaseline method), 86
 from_spec() (tensorforce.core.baselines.MLPBaseline method), 90
 from_spec() (tensorforce.core.baselines.network_baseline.NetworkBaseline method), 87
 from_spec() (tensorforce.core.baselines.NetworkBaseline method), 90
 from_spec() (tensorforce.core.distributions.Bernoulli

method), 98

from_spec() (tensorflow.core.distributions.bernoulli.Bernoulli method), 92

from_spec() (tensorflow.core.distributions.Beta method), 99

from_spec() (tensorflow.core.distributions.beta.Beta method), 93

from_spec() (tensorflow.core.distributions.Categorical method), 98

from_spec() (tensorflow.core.distributions.categorical.Categorical method), 94

from_spec() (tensorflow.core.distributions.Distribution static method), 96

from_spec() (tensorflow.core.distributions.distribution.Distribution static method), 94

from_spec() (tensorflow.core.distributions.Gaussian method), 99

from_spec() (tensorflow.core.distributions.gaussian.Gaussian method), 96

from_spec() (tensorflow.core.explorations.Constant method), 102

from_spec() (tensorflow.core.explorations.constant.Constant method), 100

from_spec() (tensorflow.core.explorations.epsilon_anneal.EpsilonAnneal method), 100

from_spec() (tensorflow.core.explorations.epsilon_decay.EpsilonDecay method), 101

from_spec() (tensorflow.core.explorations.EpsilonAnneal method), 103

from_spec() (tensorflow.core.explorations.EpsilonDecay method), 103

from_spec() (tensorflow.core.explorations.Exploration static method), 102

from_spec() (tensorflow.core.explorations.exploration.Exploration static method), 101

from_spec() (tensorflow.core.explorations.GaussianNoise method), 103

from_spec() (tensorflow.core.explorations.ornstein_uhlenbeck.OrnsteinUhlenbeck method), 102

from_spec() (tensorflow.core.explorations.OrnsteinUhlenbeck method), 104

from_spec() (tensorflow.core.memories.Latest method), 111

from_spec() (tensorflow.core.memories.Memory static method), 108

from_spec() (tensorflow.core.memories.memory.Memory static method), 104

from_spec() (tensorflow.core.memories.prioritized_replay.PrioritizedReplay method), 106

from_spec() (tensorflow.core.memories.PrioritizedReplay method), 112

from_spec() (tensorflow.core.memories.Queue method), 109

from_spec() (tensorflow.core.memories.Replay method), 111

from_spec() (tensorflow.core.memories.replay.Replay method), 107

from_spec() (tensorflow.core.networks.complex_network.ComplexLayered method), 114

from_spec() (tensorflow.core.networks.complex_network.Input method), 115

from_spec() (tensorflow.core.networks.complex_network.Output method), 115

from_spec() (tensorflow.core.networks.Conv1d method), 137

from_spec() (tensorflow.core.networks.Conv2d method), 138

from_spec() (tensorflow.core.networks.Dense method), 135

from_spec() (tensorflow.core.networks.Dropout method), 132

from_spec() (tensorflow.core.networks.Dueling method), 136

from_spec() (tensorflow.core.networks.Embedding method), 134

from_spec() (tensorflow.core.networks.Flatten method), 132

from_spec() (tensorflow.core.networks.InternalLstm method), 138

from_spec() (tensorflow.core.networks.Layer static method), 129

from_spec() (tensorflow.core.networks.layer.Conv1d method), 116

from_spec() (tensorflow.core.networks.layer.Conv2d method), 117

from_spec() (tensorflow.core.networks.layer.Dense method), 118

from_spec() (tensorflow.core.networks.layer.Dropout method), 118

from_spec() (tensorflow.core.networks.layer.Dueling method), 119

from_spec() (tensorflow.core.networks.layer.Embedding method), 120

from_spec() (tensorflow.core.networks.layer.Flatten method), 121

from_spec() (tensorflow.core.networks.layer.InternalLstm method), 121

from_spec() (tensorflow.core.networks.layer.Layer static method), 122

from_spec() (tensorflow.core.networks.layer.Linear method), 123

from_spec() (tensorflow.core.networks.layer.Lstm method), 124

from_spec() (tensorflow.core.networks.layer.Nonlinearity method), 125

from_spec() (tensorflow.core.networks.layer.Pool2d method), 125

from_spec() (tensorflow.core.networks.layer.TFLayer

method), 126

from_spec() (tensorflow.core.networks.LayerBasedNetwork method), 141

from_spec() (tensorflow.core.networks.LayeredNetwork method), 142

from_spec() (tensorflow.core.networks.Linear method), 135

from_spec() (tensorflow.core.networks.Lstm method), 139

from_spec() (tensorflow.core.networks.Network static method), 140

from_spec() (tensorflow.core.networks.network.LayerBasedNetwork method), 127

from_spec() (tensorflow.core.networks.network.LayeredNetwork method), 128

from_spec() (tensorflow.core.networks.network.Network static method), 128

from_spec() (tensorflow.core.networks.Nonlinearity method), 131

from_spec() (tensorflow.core.networks.Pool2d method), 133

from_spec() (tensorflow.core.networks.TFLayer method), 130

from_spec() (tensorflow.core.optimizers.clipped_step.ClippedStep method), 152

from_spec() (tensorflow.core.optimizers.ClippedStep method), 174

from_spec() (tensorflow.core.optimizers.Evolutionary method), 171

from_spec() (tensorflow.core.optimizers.evolutionary.Evolutionary method), 153

from_spec() (tensorflow.core.optimizers.global_optimizer.GlobalOptimizer method), 155

from_spec() (tensorflow.core.optimizers.GlobalOptimizer method), 168

from_spec() (tensorflow.core.optimizers.meta_optimizer.MetaOptimizer method), 156

from_spec() (tensorflow.core.optimizers.MetaOptimizer method), 167

from_spec() (tensorflow.core.optimizers.multi_step.MultiStep method), 157

from_spec() (tensorflow.core.optimizers.MultiStep method), 175

from_spec() (tensorflow.core.optimizers.natural_gradient.NaturalGradient method), 159

from_spec() (tensorflow.core.optimizers.NaturalGradient method), 173

from_spec() (tensorflow.core.optimizers.optimized_step.OptimizedStep method), 160

from_spec() (tensorflow.core.optimizers.OptimizedStep method), 177

from_spec() (tensorflow.core.optimizers.Optimizer static method), 166

from_spec() (tensorflow.core.optimizers.optimizer.Optimizer static method), 162

from_spec() (tensorflow.core.optimizers.SubsamplingStep method), 178

from_spec() (tensorflow.core.optimizers.Synchronization method), 180

from_spec() (tensorflow.core.optimizers.synchronization.Synchronization method), 163

from_spec() (tensorflow.core.optimizers.tf_optimizer.TFOptimizer method), 165

from_spec() (tensorflow.core.optimizers.TFOptimizer method), 170

from_spec() (tensorflow.environments.Environment static method), 182

from_spec() (tensorflow.environments.environment.Environment static method), 182

from_spec() (tensorflow.environments.MinimalTest method), 183

G

Gaussian (class in tensorflow.core.distributions), 98

Gaussian (class in tensorflow.core.distributions.gaussian), 96

GaussianNoise (class in tensorflow.core.explorations), 103

get_component() (tensorflow.models.constant_model.ConstantModel method), 189

get_component() (tensorflow.models.distribution_model.DistributionModel method), 192

get_component() (tensorflow.models.DistributionModel method), 244

get_component() (tensorflow.models.DPGTargetModel method), 258

get_component() (tensorflow.models.MemoryModel method), 239

get_component() (tensorflow.models.Model method), 235

get_component() (tensorflow.models.model.Model method), 198

get_component() (tensorflow.models.pg_log_prob_model.PGLogProbModel method), 201

get_component() (tensorflow.models.pg_model.PGModel method), 206

get_component() (tensorflow.models.pg_prob_ratio_model.PGProbRatioModel method), 211

get_component() (tensorflow.models.PGLogProbModel method), 262

get_component() (tensorflow.models.PGModel method), 249

<code>get_component()</code> (tensorforce.models.PGProbRatioModel method), 254	<code>get_components()</code> (tensorforce.models.PGModel method), 249
<code>get_component()</code> (tensorforce.models.q_demo_model.QDemoModel method), 215	<code>get_components()</code> (tensorforce.models.PGProbRatioModel method), 254
<code>get_component()</code> (tensorforce.models.q_model.QModel method), 220	<code>get_components()</code> (tensorforce.models.q_demo_model.QDemoModel method), 215
<code>get_component()</code> (tensorforce.models.q_naf_model.QNAFModel method), 224	<code>get_components()</code> (tensorforce.models.q_model.QModel method), 220
<code>get_component()</code> (tensorforce.models.q_nstep_model.QNstepModel method), 228	<code>get_components()</code> (tensorforce.models.q_naf_model.QNAFModel method), 224
<code>get_component()</code> (tensorforce.models.QDemoModel method), 278	<code>get_components()</code> (tensorforce.models.q_nstep_model.QNstepModel method), 228
<code>get_component()</code> (tensorforce.models.QModel method), 266	<code>get_components()</code> (tensorforce.models.QDemoModel method), 278
<code>get_component()</code> (tensorforce.models.QNAFModel method), 274	<code>get_components()</code> (tensorforce.models.QModel method), 266
<code>get_component()</code> (tensorforce.models.QNstepModel method), 270	<code>get_components()</code> (tensorforce.models.QNAFModel method), 274
<code>get_component()</code> (tensorforce.models.random_model.RandomModel method), 232	<code>get_components()</code> (tensorforce.models.QNstepModel method), 270
<code>get_components()</code> (tensorforce.models.constant_model.ConstantModel method), 189	<code>get_components()</code> (tensorforce.models.random_model.RandomModel method), 232
<code>get_components()</code> (tensorforce.models.distribution_model.DistributionModel method), 192	<code>get_feed_dict()</code> (tensorforce.models.constant_model.ConstantModel method), 189
<code>get_components()</code> (tensorforce.models.DistributionModel method), 244	<code>get_feed_dict()</code> (tensorforce.models.distribution_model.DistributionModel method), 192
<code>get_components()</code> (tensorforce.models.DPGTargetModel method), 258	<code>get_feed_dict()</code> (tensorforce.models.DistributionModel method), 244
<code>get_components()</code> (tensorforce.models.MemoryModel method), 239	<code>get_feed_dict()</code> (tensorforce.models.DPGTargetModel method), 258
<code>get_components()</code> (tensorforce.models.Model method), 235	<code>get_feed_dict()</code> (tensorforce.models.MemoryModel method), 239
<code>get_components()</code> (tensorforce.models.model.Model method), 198	<code>get_feed_dict()</code> (tensorforce.models.Model method), 235
<code>get_components()</code> (tensorforce.models.pg_log_prob_model.PGLogProbModel method), 201	<code>get_feed_dict()</code> (tensorforce.models.model.Model method), 198
<code>get_components()</code> (tensorforce.models.pg_model.PGModel method), 206	<code>get_feed_dict()</code> (tensorforce.models.pg_log_prob_model.PGLogProbModel method), 201
<code>get_components()</code> (tensorforce.models.pg_prob_ratio_model.PGProbRatioModel method), 211	<code>get_feed_dict()</code> (tensorforce.models.pg_model.PGModel method), 206
<code>get_components()</code> (tensorforce.models.PGLogProbModel method), 262	<code>get_feed_dict()</code> (tensorforce.models.pg_prob_ratio_model.PGProbRatioModel method), 211
	<code>get_feed_dict()</code> (tensorforce.models.PGLogProbModel method), 262
	<code>get_feed_dict()</code> (tensorforce.models.PGModel method), 249

`get_feed_dict()` (tensorflow.models.PGProbRatioModel method), 254

`get_feed_dict()` (tensorflow.models.q_demo_model.QDemoModel method), 215

`get_feed_dict()` (tensorflow.models.q_model.QModel method), 220

`get_feed_dict()` (tensorflow.models.q_naf_model.QNAFModel method), 224

`get_feed_dict()` (tensorflow.models.q_nstep_model.QNStepModel method), 228

`get_feed_dict()` (tensorflow.models.QDemoModel method), 279

`get_feed_dict()` (tensorflow.models.QModel method), 266

`get_feed_dict()` (tensorflow.models.QNAFModel method), 274

`get_feed_dict()` (tensorflow.models.QNStepModel method), 270

`get_feed_dict()` (tensorflow.models.random_model.RandomModel method), 232

`get_list_of_named_tensor()` (tensorflow.core.networks.complex_network.ComplexLayeredNetwork method), 114

`get_list_of_named_tensor()` (tensorflow.core.networks.LayerBasedNetwork method), 141

`get_list_of_named_tensor()` (tensorflow.core.networks.LayeredNetwork method), 142

`get_list_of_named_tensor()` (tensorflow.core.networks.Network method), 140

`get_list_of_named_tensor()` (tensorflow.core.networks.network.LayerBasedNetwork method), 127

`get_list_of_named_tensor()` (tensorflow.core.networks.network.LayeredNetwork method), 128

`get_list_of_named_tensor()` (tensorflow.core.networks.network.Network method), 128

`get_named_tensor()` (tensorflow.core.networks.complex_network.ComplexLayeredNetwork method), 114

`get_named_tensor()` (tensorflow.core.networks.LayerBasedNetwork method), 141

`get_named_tensor()` (tensorflow.core.networks.LayeredNetwork method), 142

`get_named_tensor()` (tensorflow.core.networks.Network method), 140

`get_named_tensor()` (tensorflow.core.networks.network.LayerBasedNetwork method), 127

`get_named_tensor()` (tensorflow.core.networks.network.LayeredNetwork method), 128

`get_named_tensor()` (tensorflow.core.networks.network.Network method), 128

`get_object()` (in module tensorflow.util), 358

`get_savable_components()` (tensorflow.models.constant_model.ConstantModel method), 189

`get_savable_components()` (tensorflow.models.distribution_model.DistributionModel method), 192

`get_savable_components()` (tensorflow.models.DistributionModel method), 244

`get_savable_components()` (tensorflow.models.DPGTargetModel method), 258

`get_savable_components()` (tensorflow.models.MemoryModel method), 239

`get_savable_components()` (tensorflow.models.Model method), 235

`get_savable_components()` (tensorflow.models.model.Model method), 198

`get_savable_components()` (tensorflow.models.pg_log_prob_model.PGLogProbModel method), 202

`get_savable_components()` (tensorflow.models.pg_model.PGModel method), 206

`get_savable_components()` (tensorflow.models.pg_prob_ratio_model.PGProbRatioModel method), 211

`get_savable_components()` (tensorflow.models.PGLogProbModel method), 262

`get_savable_components()` (tensorflow.models.PGModel method), 249

`get_savable_components()` (tensorflow.models.PGProbRatioModel method), 254

`get_savable_components()` (tensorflow.models.q_demo_model.QDemoModel method), 215

`get_savable_components()` (tensorflow.models.q_model.QModel method), 220

`get_savable_components()` (tensorflow.models.q_naf_model.QNAFModel method), 224

- method), 224
- get_savable_components() (tensorflow.models.q_nstep_model.QNstepModel method), 228
- get_savable_components() (tensorflow.models.QDemoModel method), 279
- get_savable_components() (tensorflow.models.QModel method), 266
- get_savable_components() (tensorflow.models.QNAFModel method), 274
- get_savable_components() (tensorflow.models.QNstepModel method), 270
- get_savable_components() (tensorflow.models.random_model.RandomModel method), 232
- get_savable_variables() (tensorflow.util.SavableComponent method), 358
- get_state() (tensorflow.tests.test_tutorial_code.TestTutorialCodeMyGymEnv method), 336
- get_summaries() (tensorflow.core.baselines.aggregated_baseline.AggregatedBaseline method), 83
- get_summaries() (tensorflow.core.baselines.AggregatedBaseline method), 89
- get_summaries() (tensorflow.core.baselines.Baseline method), 88
- get_summaries() (tensorflow.core.baselines.baseline.Baseline method), 84
- get_summaries() (tensorflow.core.baselines.cnn_baseline.CNNBaseline method), 85
- get_summaries() (tensorflow.core.baselines.CNNBaseline method), 91
- get_summaries() (tensorflow.core.baselines.mlp_baseline.MLPBaseline method), 86
- get_summaries() (tensorflow.core.baselines.MLPBaseline method), 91
- get_summaries() (tensorflow.core.baselines.network_baseline.NetworkBaseline method), 87
- get_summaries() (tensorflow.core.baselines.NetworkBaseline method), 90
- get_summaries() (tensorflow.core.distributions.Bernoulli method), 98
- get_summaries() (tensorflow.core.distributions.bernoulli.Bernoulli method), 92
- get_summaries() (tensorflow.core.distributions.Beta method), 99
- get_summaries() (tensorflow.core.distributions.beta.Beta method), 93
- get_summaries() (tensorflow.core.distributions.Categorical method), 98
- get_summaries() (tensorflow.core.distributions.categorical.Categorical method), 94
- get_summaries() (tensorflow.core.distributions.Distribution method), 96
- get_summaries() (tensorflow.core.distributions.distribution.Distribution method), 94
- get_summaries() (tensorflow.core.distributions.Gaussian method), 99
- get_summaries() (tensorflow.core.distributions.gaussian.Gaussian method), 96
- get_summaries() (tensorflow.core.memories.Latest method), 111
- get_summaries() (tensorflow.core.memories.Memory method), 108
- get_summaries() (tensorflow.core.memories.memory.Memory method), 104
- get_summaries() (tensorflow.core.memories.prioritized_replay.PrioritizedReplay method), 106
- get_summaries() (tensorflow.core.memories.PrioritizedReplay method), 112
- get_summaries() (tensorflow.core.memories.Queue method), 110
- get_summaries() (tensorflow.core.memories.Replay method), 111
- get_summaries() (tensorflow.core.memories.replay.Replay method), 107
- get_summaries() (tensorflow.core.networks.complex_network.ComplexLayeredNetwork method), 114
- get_summaries() (tensorflow.core.networks.complex_network.Input method), 115
- get_summaries() (tensorflow.core.networks.complex_network.Output method), 115
- get_summaries() (tensorflow.core.networks.Conv1d method), 137
- get_summaries() (tensorflow.core.networks.Conv2d method), 138
- get_summaries() (tensorflow.core.networks.Dense

method), 135

get_summaries() (tensorflow.core.networks.Dropout method), 132

get_summaries() (tensorflow.core.networks.Dueling method), 136

get_summaries() (tensorflow.core.networks.Embedding method), 134

get_summaries() (tensorflow.core.networks.Flatten method), 132

get_summaries() (tensorflow.core.networks.InternalLstm method), 138

get_summaries() (tensorflow.core.networks.Layer method), 129

get_summaries() (tensorflow.core.networks.layer.Conv1d method), 116

get_summaries() (tensorflow.core.networks.layer.Conv2d method), 117

get_summaries() (tensorflow.core.networks.layer.Dense method), 118

get_summaries() (tensorflow.core.networks.layer.Dropout method), 119

get_summaries() (tensorflow.core.networks.layer.Dueling method), 119

get_summaries() (tensorflow.core.networks.layer.Embedding method), 120

get_summaries() (tensorflow.core.networks.layer.Flatten method), 121

get_summaries() (tensorflow.core.networks.layer.InternalLstm method), 121

get_summaries() (tensorflow.core.networks.layer.Layer method), 122

get_summaries() (tensorflow.core.networks.layer.Linear method), 123

get_summaries() (tensorflow.core.networks.layer.Lstm method), 124

get_summaries() (tensorflow.core.networks.layer.Nonlinearity method), 125

get_summaries() (tensorflow.core.networks.layer.Pool2d method), 125

get_summaries() (tensorflow.core.networks.layer.TFLayer method), 126

get_summaries() (tensorflow.core.networks.LayerBasedNetwork method), 141

get_summaries() (tensorflow.core.networks.LayeredNetwork method), 142

get_summaries() (tensorflow.core.networks.Linear method), 135

get_summaries() (tensorflow.core.networks.Lstm method), 139

get_summaries() (tensorflow.core.networks.Network method), 140

get_summaries() (tensorflow.core.networks.network.LayerBasedNetwork method), 127

get_summaries() (tensorflow.core.networks.network.LayeredNetwork method), 128

get_summaries() (tensorflow.core.networks.network.Network method), 129

get_summaries() (tensorflow.core.networks.Nonlinearity method), 131

get_summaries() (tensorflow.core.networks.Pool2d method), 133

get_summaries() (tensorflow.core.networks.TFLayer method), 130

get_summaries() (tensorflow.core.optimizers.clipped_step.ClippedStep method), 152

get_summaries() (tensorflow.core.optimizers.ClippedStep method), 174

get_summaries() (tensorflow.core.optimizers.Evolutionary method), 171

get_summaries() (tensorflow.core.optimizers.evolutionary.Evolutionary method), 153

get_summaries() (tensorflow.core.optimizers.global_optimizer.GlobalOptimizer method), 155

get_summaries() (tensorflow.core.optimizers.GlobalOptimizer method), 169

get_summaries() (tensorflow.core.optimizers.meta_optimizer.MetaOptimizer method), 156

get_summaries() (tensorflow.core.optimizers.MetaOptimizer method), 167

get_summaries() (tensorflow.core.optimizers.multi_step.MultiStep method), 157

get_summaries() (tensorflow.core.optimizers.MultiStep method), 175

get_summaries() (tensorflow.core.optimizers.natural_gradient.NaturalGradient

method), 159

get_summaries() (tensorflow.core.optimizers.NaturalGradient method), 173

get_summaries() (tensorflow.core.optimizers.optimized_step.OptimizedStep method), 161

get_summaries() (tensorflow.core.optimizers.OptimizedStep method), 177

get_summaries() (tensorflow.core.optimizers.Optimizer method), 166

get_summaries() (tensorflow.core.optimizers.optimizer.Optimizer method), 162

get_summaries() (tensorflow.core.optimizers.SubsamplingStep method), 178

get_summaries() (tensorflow.core.optimizers.Synchronization method), 180

get_summaries() (tensorflow.core.optimizers.synchronization.Synchronization method), 163

get_summaries() (tensorflow.core.optimizers.tf_optimizer.TFOptimizer method), 165

get_summaries() (tensorflow.core.optimizers.TFOptimizer method), 170

get_summaries() (tensorflow.models.constant_model.ConstantModel method), 189

get_summaries() (tensorflow.models.distribution_model.DistributionModel method), 192

get_summaries() (tensorflow.models.DistributionModel method), 244

get_summaries() (tensorflow.models.DPGTargetModel method), 258

get_summaries() (tensorflow.models.MemoryModel method), 239

get_summaries() (tensorflow.models.Model method), 236

get_summaries() (tensorflow.models.model.Model method), 198

get_summaries() (tensorflow.models.pg_log_prob_model.PGLogProbModel method), 202

get_summaries() (tensorflow.models.pg_model.PGModel method), 206

get_summaries() (tensorflow.models.pg_prob_ratio_model.PGProbRatioModel method), 211

get_summaries() (tensorflow.models.PGLogProbModel method), 262

get_summaries() (tensorflow.models.PGModel method), 249

get_summaries() (tensorflow.models.PGProbRatioModel method), 254

get_summaries() (tensorflow.models.q_demo_model.QDemoModel method), 215

get_summaries() (tensorflow.models.q_model.QModel method), 220

get_summaries() (tensorflow.models.q_naf_model.QNAFModel method), 224

get_summaries() (tensorflow.models.q_nstep_model.QNstepModel method), 228

get_summaries() (tensorflow.models.QDemoModel method), 279

get_summaries() (tensorflow.models.QModel method), 266

get_summaries() (tensorflow.models.QNAFModel method), 274

get_summaries() (tensorflow.models.QNstepModel method), 270

get_summaries() (tensorflow.models.random_model.RandomModel method), 232

get_variables() (tensorflow.core.baselines.aggregated_baseline.AggregatedBaseline method), 83

get_variables() (tensorflow.core.baselines.AggregatedBaseline method), 89

get_variables() (tensorflow.core.baselines.Baseline method), 88

get_variables() (tensorflow.core.baselines.baseline.Baseline method), 84

get_variables() (tensorflow.core.baselines.cnn_baseline.CNNBaseline method), 85

get_variables() (tensorflow.core.baselines.CNNBaseline method), 91

get_variables() (tensorflow.core.baselines.mlp_baseline.MLPBaseline method), 86

get_variables() (tensorflow.core.baselines.MLPBaseline method), 91

get_variables() (tensorflow.core.baselines.network_baseline.NetworkBaseline method), 87

`get_variables()` (tensorflow.core.baselines.NetworkBaseline method), 90

`get_variables()` (tensorflow.core.distributions.Bernoulli method), 98

`get_variables()` (tensorflow.core.distributions.bernoulli.Bernoulli method), 92

`get_variables()` (tensorflow.core.distributions.Beta method), 99

`get_variables()` (tensorflow.core.distributions.beta.Beta method), 93

`get_variables()` (tensorflow.core.distributions.Categorical method), 98

`get_variables()` (tensorflow.core.distributions.categorical.Categorical method), 94

`get_variables()` (tensorflow.core.distributions.Distribution method), 96

`get_variables()` (tensorflow.core.distributions.distribution.Distribution method), 95

`get_variables()` (tensorflow.core.distributions.Gaussian method), 99

`get_variables()` (tensorflow.core.distributions.gaussian.Gaussian method), 96

`get_variables()` (tensorflow.core.explorations.Constant method), 102

`get_variables()` (tensorflow.core.explorations.constant.Constant method), 100

`get_variables()` (tensorflow.core.explorations.epsilon_anneal.EpsilonAnneal method), 100

`get_variables()` (tensorflow.core.explorations.epsilon_decay.EpsilonDecay method), 101

`get_variables()` (tensorflow.core.explorations.EpsilonAnneal method), 103

`get_variables()` (tensorflow.core.explorations.EpsilonDecay method), 103

`get_variables()` (tensorflow.core.explorations.Exploration method), 102

`get_variables()` (tensorflow.core.explorations.exploration.Exploration method), 101

`get_variables()` (tensorflow.core.explorations.GaussianNoise method), 103

`get_variables()` (tensorflow.core.explorations.ornstein_uhlenbeck_process.OrnsteinUhlenbeckProcess method), 102

`get_variables()` (tensorflow.core.explorations.OrnsteinUhlenbeckProcess method), 104

`get_variables()` (tensorflow.core.memories.Latest method), 111

`get_variables()` (tensorflow.core.memories.Memory method), 108

`get_variables()` (tensorflow.core.memories.memory.Memory method), 105

`get_variables()` (tensorflow.core.memories.prioritized_replay.PrioritizedReplay method), 106

`get_variables()` (tensorflow.core.memories.PrioritizedReplay method), 113

`get_variables()` (tensorflow.core.memories.Queue method), 110

`get_variables()` (tensorflow.core.memories.Replay method), 112

`get_variables()` (tensorflow.core.memories.replay.Replay method), 108

`get_variables()` (tensorflow.core.networks.complex_network.ComplexLayeredNetwork method), 114

`get_variables()` (tensorflow.core.networks.complex_network.Input method), 115

`get_variables()` (tensorflow.core.networks.complex_network.Output method), 115

`get_variables()` (tensorflow.core.networks.Conv1d method), 137

`get_variables()` (tensorflow.core.networks.Conv2d method), 138

`get_variables()` (tensorflow.core.networks.Dense method), 136

`get_variables()` (tensorflow.core.networks.Dropout method), 132

`get_variables()` (tensorflow.core.networks.Dueling method), 136

`get_variables()` (tensorflow.core.networks.Embedding method), 134

`get_variables()` (tensorflow.core.networks.Flatten method), 132

`get_variables()` (tensorflow.core.networks.InternalLstm method), 139

`get_variables()` (tensorflow.core.networks.Layer method), 129

`get_variables()` (tensorflow.core.networks.layer.Conv1d method), 137

method), 116

get_variables() (tensorflow.core.networks.layer.Conv2d method), 117

get_variables() (tensorflow.core.networks.layer.Dense method), 118

get_variables() (tensorflow.core.networks.layer.Dropout method), 119

get_variables() (tensorflow.core.networks.layer.Dueling method), 119

get_variables() (tensorflow.core.networks.layer.Embedding method), 120

get_variables() (tensorflow.core.networks.layer.Flatten method), 121

get_variables() (tensorflow.core.networks.layer.InternalLstm method), 122

get_variables() (tensorflow.core.networks.layer.Layer method), 122

get_variables() (tensorflow.core.networks.layer.Linear method), 123

get_variables() (tensorflow.core.networks.layer.Lstm method), 124

get_variables() (tensorflow.core.networks.layer.Nonlinearity method), 125

get_variables() (tensorflow.core.networks.layer.Pool2d method), 125

get_variables() (tensorflow.core.networks.layer.TFLayer method), 126

get_variables() (tensorflow.core.networks.LayerBasedNetwork method), 141

get_variables() (tensorflow.core.networks.LayeredNetwork method), 142

get_variables() (tensorflow.core.networks.Linear method), 135

get_variables() (tensorflow.core.networks.Lstm method), 139

get_variables() (tensorflow.core.networks.Network method), 140

get_variables() (tensorflow.core.networks.network.LayerBasedNetwork method), 127

get_variables() (tensorflow.core.networks.network.LayeredNetwork method), 128

get_variables() (tensorflow.core.networks.network.Network method), 129

get_variables() (tensorflow.core.networks.Nonlinearity method), 131

get_variables() (tensorflow.core.networks.Pool2d method), 133

get_variables() (tensorflow.core.networks.TFLayer method), 130

get_variables() (tensorflow.core.optimizers.clipped_step.ClippedStep method), 152

get_variables() (tensorflow.core.optimizers.ClippedStep method), 174

get_variables() (tensorflow.core.optimizers.Evolutionary method), 171

get_variables() (tensorflow.core.optimizers.evolutionary.Evolutionary method), 153

get_variables() (tensorflow.core.optimizers.global_optimizer.GlobalOptimizer method), 155

get_variables() (tensorflow.core.optimizers.GlobalOptimizer method), 169

get_variables() (tensorflow.core.optimizers.meta_optimizer.MetaOptimizer method), 156

get_variables() (tensorflow.core.optimizers.MetaOptimizer method), 167

get_variables() (tensorflow.core.optimizers.multi_step.MultiStep method), 157

get_variables() (tensorflow.core.optimizers.MultiStep method), 175

get_variables() (tensorflow.core.optimizers.natural_gradient.NaturalGradient method), 159

get_variables() (tensorflow.core.optimizers.NaturalGradient method), 173

get_variables() (tensorflow.core.optimizers.optimized_step.OptimizedStep method), 161

get_variables() (tensorflow.core.optimizers.OptimizedStep method), 177

get_variables() (tensorflow.core.optimizers.Optimizer method), 166

get_variables() (tensorflow.core.optimizers.optimizer.Optimizer method), 162

get_variables() (tensorflow.core.optimizers.SubsamplingStep method), 178

get_variables() (tensorflow.core.optimizers.Synchronization method), 180

get_variables() (tensorflow

[force.core.optimizers.synchronization.Synchronization method\), 163](#)
[get_variables\(\) \(tensorflow.core.optimizers.tf_optimizer.TFOptimizer method\), 165](#)
[get_variables\(\) \(tensorflow.core.optimizers.TFOptimizer method\), 170](#)
[get_variables\(\) \(tensorflow.models.constant_model.ConstantModel method\), 189](#)
[get_variables\(\) \(tensorflow.models.distribution_model.DistributionModel method\), 192](#)
[get_variables\(\) \(tensorflow.models.DistributionModel method\), 244](#)
[get_variables\(\) \(tensorflow.models.DPGTargetModel method\), 258](#)
[get_variables\(\) \(tensorflow.models.MemoryModel method\), 239](#)
[get_variables\(\) \(tensorflow.models.Model method\), 236](#)
[get_variables\(\) \(tensorflow.models.model.Model method\), 198](#)
[get_variables\(\) \(tensorflow.models.pg_log_prob_model.PGLogProbModel method\), 202](#)
[get_variables\(\) \(tensorflow.models.pg_model.PGModel method\), 206](#)
[get_variables\(\) \(tensorflow.models.pg_prob_ratio_model.PGProbRatioModel method\), 211](#)
[get_variables\(\) \(tensorflow.models.PGLogProbModel method\), 262](#)
[get_variables\(\) \(tensorflow.models.PGModel method\), 249](#)
[get_variables\(\) \(tensorflow.models.PGProbRatioModel method\), 254](#)
[get_variables\(\) \(tensorflow.models.q_demo_model.QDemoModel method\), 215](#)
[get_variables\(\) \(tensorflow.models.q_model.QModel method\), 220](#)
[get_variables\(\) \(tensorflow.models.q_naf_model.QNAFModel method\), 224](#)
[get_variables\(\) \(tensorflow.models.q_nstep_model.QNstepModel method\), 228](#)
[get_variables\(\) \(tensorflow.models.QDemoModel method\), 279](#)
[get_variables\(\) \(tensorflow.models.QModel method\), 266](#)
[get_variables\(\) \(tensorflow.models.QNAFModel method\), 274](#)
[get_variables\(\) \(tensorflow.models.QNstepModel method\), 270](#)
[get_variables\(\) \(tensorflow.models.random_model.RandomModel method\), 232](#)
[get_wrapper\(\) \(tensorflow.core.optimizers.tf_optimizer.TFOptimizer static method\), 165](#)
[get_wrapper\(\) \(tensorflow.core.optimizers.TFOptimizer static method\), 170](#)
[global_step \(tensorflow.execution.threaded_runner.ThreadedRunner attribute\), 185](#)
[global_step \(tensorflow.execution.ThreadedRunner attribute\), 188](#)
[GlobalOptimizer \(class in tensorflow.core.optimizers\), 168](#)
[GlobalOptimizer \(class in tensorflow.core.optimizers.global_optimizer\), 154](#)
[id\(\) \(tensorflow.tests.test_constant_agent.TestConstantAgent method\), 288](#)
[id\(\) \(tensorflow.tests.test_dqfd_agent.TestDQFDAGent method\), 294](#)
[id\(\) \(tensorflow.tests.test_dqn_agent.TestDQNAgent method\), 299](#)
[id\(\) \(tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent method\), 305](#)
[id\(\) \(tensorflow.tests.test_naf_agent.TestNAFAgent method\), 310](#)
[id\(\) \(tensorflow.tests.test_ppo_agent.TestPPOAgent method\), 316](#)
[id\(\) \(tensorflow.tests.test_quickstart_example.TestQuickstartExample method\), 321](#)
[id\(\) \(tensorflow.tests.test_random_agent.TestRandomAgent method\), 326](#)
[id\(\) \(tensorflow.tests.test_reward_estimation.TestRewardEstimation method\), 330](#)
[id\(\) \(tensorflow.tests.test_trpo_agent.TestTRPOAgent method\), 335](#)
[id\(\) \(tensorflow.tests.test_tutorial_code.TestTutorialCode method\), 340](#)
[id\(\) \(tensorflow.tests.test_vpg_agent.TestVPGAgent method\), 345](#)
[id\(\) \(tensorflow.tests.test_vpg_baselines.TestVPGBaselines method\), 351](#)
[id\(\) \(tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers method\), 356](#)
[import_demo_experience\(\) \(tensorflow.models.q_demo_model.QDemoModel method\), 216](#)
[import_demo_experience\(\) \(tensorflow.models.QDemoModel method\), 279](#)
[import_demonstrations\(\) \(tensorflow.agents.dqfd_agent.DQFDAGent method\),](#)

- 31
- `import_demonstrations()` (tensorforce.agents.DQFDDAgent method), 59
- `import_experience()` (tensorforce.agents.DDPGAgent method), 73
- `import_experience()` (tensorforce.agents.dqfd_agent.DQFDDAgent method), 31
- `import_experience()` (tensorforce.agents.DQFDDAgent method), 59
- `import_experience()` (tensorforce.agents.dqn_agent.DQNAgent method), 33
- `import_experience()` (tensorforce.agents.dqn_nstep_agent.DQNNstepAgent method), 35
- `import_experience()` (tensorforce.agents.DQNAgent method), 61
- `import_experience()` (tensorforce.agents.DQNNstepAgent method), 63
- `import_experience()` (tensorforce.agents.learning_agent.LearningAgent method), 38
- `import_experience()` (tensorforce.agents.LearningAgent method), 56
- `import_experience()` (tensorforce.agents.naf_agent.NAFAgent method), 40
- `import_experience()` (tensorforce.agents.NAFAgent method), 65
- `import_experience()` (tensorforce.agents.ppo_agent.PPOAgent method), 43
- `import_experience()` (tensorforce.agents.PPOAgent method), 67
- `import_experience()` (tensorforce.agents.trpo_agent.TRPOAgent method), 47
- `import_experience()` (tensorforce.agents.TRPOAgent method), 69
- `import_experience()` (tensorforce.agents.vpg_agent.VPGAgent method), 49
- `import_experience()` (tensorforce.agents.VPGAgent method), 71
- `import_experience()` (tensorforce.models.distribution_model.DistributionModel method), 192
- `import_experience()` (tensorforce.models.DistributionModel method), 245
- `import_experience()` (tensorforce.models.DPGTargetModel method), 258
- `import_experience()` (tensorforce.models.MemoryModel method), 239
- `import_experience()` (tensorforce.models.pg_log_prob_model.PGLogProbModel method), 202
- `import_experience()` (tensorforce.models.pg_model.PGModel method), 206
- `import_experience()` (tensorforce.models.pg_prob_ratio_model.PGProbRatioModel method), 211
- `import_experience()` (tensorforce.models.PGLogProbModel method), 262
- `import_experience()` (tensorforce.models.PGModel method), 249
- `import_experience()` (tensorforce.models.PGProbRatioModel method), 254
- `import_experience()` (tensorforce.models.q_demo_model.QDemoModel method), 216
- `import_experience()` (tensorforce.models.q_model.QModel method), 220
- `import_experience()` (tensorforce.models.q_naf_model.QNAFModel method), 224
- `import_experience()` (tensorforce.models.q_nstep_model.QNstepModel method), 228
- `import_experience()` (tensorforce.models.QDemoModel method), 279
- `import_experience()` (tensorforce.models.QModel method), 266
- `import_experience()` (tensorforce.models.QNAFModel method), 274
- `import_experience()` (tensorforce.models.QNstepModel method), 270
- `initialize()` (tensorforce.models.constant_model.ConstantModel method), 190
- `initialize()` (tensorforce.models.distribution_model.DistributionModel method), 193
- `initialize()` (tensorforce.models.DistributionModel method), 245
- `initialize()` (tensorforce.models.DPGTargetModel method), 258
- `initialize()` (tensorforce.models.MemoryModel method), 239
- `initialize()` (tensorforce.models.Model method), 236
- `initialize()` (tensorforce.models.model.Model method), 198
- `initialize()` (tensorforce.models.pg_log_prob_model.PGLogProbModel method), 202

method), 202

initialize() (tensorforce.models.pg_model.PGModel method), 206

initialize() (tensorforce.models.pg_prob_ratio_model.PGProbRatioModel method), 211

initialize() (tensorforce.models.PGLogProbModel method), 262

initialize() (tensorforce.models.PGModel method), 249

initialize() (tensorforce.models.PGProbRatioModel method), 254

initialize() (tensorforce.models.q_demo_model.QDemoModel method), 216

initialize() (tensorforce.models.q_model.QModel method), 220

initialize() (tensorforce.models.q_naf_model.QNAFModel method), 224

initialize() (tensorforce.models.q_nstep_model.QNstepModel method), 229

initialize() (tensorforce.models.QDemoModel method), 279

initialize() (tensorforce.models.QModel method), 266

initialize() (tensorforce.models.QNAFModel method), 274

initialize() (tensorforce.models.QNstepModel method), 270

initialize() (tensorforce.models.random_model.RandomModel method), 233

initialize_model() (tensorforce.agents.Agent method), 51

initialize_model() (tensorforce.agents.agent.Agent method), 26

initialize_model() (tensorforce.agents.constant_agent.ConstantAgent method), 28

initialize_model() (tensorforce.agents.ConstantAgent method), 53

initialize_model() (tensorforce.agents.DDPGAgent method), 73

initialize_model() (tensorforce.agents.dqfd_agent.DQFDAgent method), 31

initialize_model() (tensorforce.agents.DQFDAgent method), 59

initialize_model() (tensorforce.agents.dqn_agent.DQNAgent method), 33

initialize_model() (tensorforce.agents.dqn_nstep_agent.DQNNstepAgent method), 35

initialize_model() (tensorforce.agents.DQNAgent method), 61

initialize_model() (tensorforce.agents.DQNNstepAgent method), 63

initialize_model() (tensorforce.agents.learning_agent.LearningAgent method), 38

initialize_model() (tensorforce.agents.LearningAgent method), 57

initialize_model() (tensorforce.agents.naf_agent.NAFAgent method), 40

initialize_model() (tensorforce.agents.NAFAgent method), 65

initialize_model() (tensorforce.agents.ppo_agent.PPOAgent method), 43

initialize_model() (tensorforce.agents.PPOAgent method), 67

initialize_model() (tensorforce.agents.random_agent.RandomAgent method), 45

initialize_model() (tensorforce.agents.RandomAgent method), 54

initialize_model() (tensorforce.agents.trpo_agent.TRPOAgent method), 47

initialize_model() (tensorforce.agents.TRPOAgent method), 69

initialize_model() (tensorforce.agents.vpg_agent.VPGAgent method), 49

initialize_model() (tensorforce.agents.VPGAgent method), 71

Input (class in tensorforce.core.networks.complex_network), 114

InternalLstm (class in tensorforce.core.networks), 138

InternalLstm (class in tensorforce.core.networks.layer), 121

internals_spec() (tensorforce.core.networks.complex_network.ComplexLayeredNetwork method), 114

internals_spec() (tensorforce.core.networks.complex_network.Input method), 115

internals_spec() (tensorforce.core.networks.complex_network.Output method), 116

internals_spec() (tensorforce.core.networks.Conv1d method), 137

internals_spec() (tensorforce.core.networks.Conv2d method), 138

internals_spec() (tensorforce.core.networks.Dense method), 136

internals_spec() (tensorforce.core.networks.Dropout method), 132

internals_spec() (tensorforce.core.networks.Dueling method), 136

internals_spec() (tensorforce.core.networks.Embedding method), 134

- `internals_spec()` (tensorflow.core.networks.Flatten method), 132
- `internals_spec()` (tensorflow.core.networks.InternalLstm method), 139
- `internals_spec()` (tensorflow.core.networks.Layer method), 129
- `internals_spec()` (tensorflow.core.networks.layer.Conv1d method), 116
- `internals_spec()` (tensorflow.core.networks.layer.Conv2d method), 117
- `internals_spec()` (tensorflow.core.networks.layer.Dense method), 118
- `internals_spec()` (tensorflow.core.networks.layer.Dropout method), 119
- `internals_spec()` (tensorflow.core.networks.layer.Dueling method), 120
- `internals_spec()` (tensorflow.core.networks.layer.Embedding method), 120
- `internals_spec()` (tensorflow.core.networks.layer.Flatten method), 121
- `internals_spec()` (tensorflow.core.networks.layer.InternalLstm method), 122
- `internals_spec()` (tensorflow.core.networks.layer.Layer method), 122
- `internals_spec()` (tensorflow.core.networks.layer.Linear method), 123
- `internals_spec()` (tensorflow.core.networks.layer.Lstm method), 124
- `internals_spec()` (tensorflow.core.networks.layer.Nonlinearity method), 125
- `internals_spec()` (tensorflow.core.networks.layer.Pool2d method), 126
- `internals_spec()` (tensorflow.core.networks.layer.TFLayer method), 126
- `internals_spec()` (tensorflow.core.networks.LayerBasedNetwork method), 141
- `internals_spec()` (tensorflow.core.networks.LayeredNetwork method), 142
- `internals_spec()` (tensorflow.core.networks.Linear method), 135
- `internals_spec()` (tensorflow.core.networks.Lstm method), 139
- `internals_spec()` (tensorflow.core.networks.Network method), 140
- `internals_spec()` (tensorflow.core.networks.network.LayerBasedNetwork method), 127
- `internals_spec()` (tensorflow.core.networks.network.LayeredNetwork method), 128
- `internals_spec()` (tensorflow.core.networks.network.Network method), 129
- `internals_spec()` (tensorflow.core.networks.Nonlinearity method), 131
- `internals_spec()` (tensorflow.core.networks.Pool2d method), 133
- `internals_spec()` (tensorflow.core.networks.TFLayer method), 130
- `is_terminal` (tensorflow.contrib.ale.ALE attribute), 75
- `Iterative` (class in tensorflow.core.optimizers.solvers), 147
- `Iterative` (class in tensorflow.core.optimizers.solvers.iterative), 144
- ## L
- `last_observation()` (tensorflow.agents.Agent method), 51
- `last_observation()` (tensorflow.agents.agent.Agent method), 26
- `last_observation()` (tensorflow.agents.constant_agent.ConstantAgent method), 28
- `last_observation()` (tensorflow.agents.ConstantAgent method), 53
- `last_observation()` (tensorflow.agents.DDPGAgent method), 73
- `last_observation()` (tensorflow.agents.dqfd_agent.DQFDAgent method), 31
- `last_observation()` (tensorflow.agents.DQFDAgent method), 59
- `last_observation()` (tensorflow.agents.dqn_agent.DQNAgent method), 33
- `last_observation()` (tensorflow.agents.dqn_nstep_agent.DQNNstepAgent method), 35
- `last_observation()` (tensorflow.agents.DQNAgent method), 61
- `last_observation()` (tensorflow.agents.DQNNstepAgent method), 63
- `last_observation()` (tensorflow.agents.learning_agent.LearningAgent method), 38
- `last_observation()` (tensorflow.agents.LearningAgent method), 57
- `last_observation()` (tensorflow.agents.naf_agent.NAFAgent method), 40
- `last_observation()` (tensorflow.agents.NAFAgent method), 65

`last_observation()` (tensorforce.agents.ppo_agent.PPOAgent method), 43

`last_observation()` (tensorforce.agents.PPOAgent method), 67

`last_observation()` (tensorforce.agents.random_agent.RandomAgent method), 45

`last_observation()` (tensorforce.agents.RandomAgent method), 54

`last_observation()` (tensorforce.agents.trpo_agent.TRPOAgent method), 47

`last_observation()` (tensorforce.agents.TRPOAgent method), 69

`last_observation()` (tensorforce.agents.vpg_agent.VPGAgent method), 49

`last_observation()` (tensorforce.agents.VPGAgent method), 71

`Latest` (class in tensorforce.core.memories), 110

`Layer` (class in tensorforce.core.networks), 129

`Layer` (class in tensorforce.core.networks.layer), 122

`LayerBasedNetwork` (class in tensorforce.core.networks), 140

`LayerBasedNetwork` (class in tensorforce.core.networks.network), 127

`LayeredNetwork` (class in tensorforce.core.networks), 141

`LayeredNetwork` (class in tensorforce.core.networks.network), 127

`LearningAgent` (class in tensorforce.agents), 55

`LearningAgent` (class in tensorforce.agents.learning_agent), 37

`Linear` (class in tensorforce.core.networks), 134

`Linear` (class in tensorforce.core.networks.layer), 123

`LineSearch` (class in tensorforce.core.optimizers.solvers), 150

`LineSearch` (class in tensorforce.core.optimizers.solvers.line_search), 145

`longMessage` (tensorforce.tests.test_constant_agent.TestConstantAgent attribute), 288

`longMessage` (tensorforce.tests.test_dqfd_agent.TestDQFDAGent attribute), 294

`longMessage` (tensorforce.tests.test_dqn_agent.TestDQNAgent attribute), 299

`longMessage` (tensorforce.tests.test_dqn_nstep_agent.TestDQNNstepAgent attribute), 305

`longMessage` (tensorforce.tests.test_naf_agent.TestNAFAGent attribute), 310

`longMessage` (tensorforce.tests.test_ppo_agent.TestPPOAgent attribute), 316

`longMessage` (tensorforce.tests.test_quickstart_example.TestQuickstartExample attribute), 321

`longMessage` (tensorforce.tests.test_random_agent.TestRandomAgent attribute), 326

`longMessage` (tensorforce.tests.test_reward_estimation.TestRewardEstimation attribute), 330

`longMessage` (tensorforce.tests.test_trpo_agent.TestTRPOAgent attribute), 335

`longMessage` (tensorforce.tests.test_tutorial_code.TestTutorialCode attribute), 340

`longMessage` (tensorforce.tests.test_vpg_agent.TestVPGAgent attribute), 346

`longMessage` (tensorforce.tests.test_vpg_baselines.TestVPGBaselines attribute), 351

`longMessage` (tensorforce.tests.test_vpg_optimizers.TestVPGOptimizers attribute), 356

`Lstm` (class in tensorforce.core.networks), 139

`Lstm` (class in tensorforce.core.networks.layer), 123

M

`map_tensors()` (in module tensorforce.util), 358

`maxDiff` (tensorforce.tests.test_constant_agent.TestConstantAgent attribute), 288

`maxDiff` (tensorforce.tests.test_dqfd_agent.TestDQFDAGent attribute), 294

`maxDiff` (tensorforce.tests.test_dqn_agent.TestDQNAgent attribute), 299

`maxDiff` (tensorforce.tests.test_dqn_nstep_agent.TestDQNNstepAgent attribute), 305

`maxDiff` (tensorforce.tests.test_naf_agent.TestNAFAGent attribute), 310

`maxDiff` (tensorforce.tests.test_ppo_agent.TestPPOAgent attribute), 316

`maxDiff` (tensorforce.tests.test_quickstart_example.TestQuickstartExample attribute), 321

`maxDiff` (tensorforce.tests.test_random_agent.TestRandomAgent attribute), 326

`maxDiff` (tensorforce.tests.test_reward_estimation.TestRewardEstimation attribute), 330

`maxDiff` (tensorforce.tests.test_trpo_agent.TestTRPOAgent attribute), 335

`maxDiff` (tensorforce.tests.test_tutorial_code.TestTutorialCode attribute), 340

`maxDiff` (tensorforce.tests.test_vpg_agent.TestVPGAgent attribute), 346

`maxDiff` (tensorforce.tests.test_vpg_baselines.TestVPGBaselines attribute), 351

`maxDiff` (tensorforce.tests.test_vpg_optimizers.TestVPGOptimizers attribute), 356

`MazeExplorer` (class in tensorforce.contrib.maze_explorer), 76

`Memory` (class in tensorforce.core.memories), 108

`Memory` (class in tensorforce.core.memories.memory), 104

`MemoryModel` (class in tensorforce.models), 238

- merge_custom() (tensorforce.meta_parameter_recorder.MetaParameterRecorder method), 357
- message (tensorforce.exception.TensorForceError attribute), 357
- message (tensorforce.TensorForceError attribute), 359
- MetaOptimizer (class in tensorforce.core.optimizers), 167
- MetaOptimizer (class in tensorforce.core.optimizers.meta_optimizer), 156
- MetaParameterRecorder (class in tensorforce.meta_parameter_recorder), 357
- MinimalTest (class in tensorforce.environments), 183
- minimize() (tensorforce.core.optimizers.clipped_step.ClippedStep method), 152
- minimize() (tensorforce.core.optimizers.ClippedStep method), 174
- minimize() (tensorforce.core.optimizers.Evolutionary method), 171
- minimize() (tensorforce.core.optimizers.evolutionary.Evolutionary method), 153
- minimize() (tensorforce.core.optimizers.global_optimizer.GlobalOptimizer method), 155
- minimize() (tensorforce.core.optimizers.GlobalOptimizer method), 169
- minimize() (tensorforce.core.optimizers.meta_optimizer.MetaOptimizer method), 156
- minimize() (tensorforce.core.optimizers.MetaOptimizer method), 167
- minimize() (tensorforce.core.optimizers.multi_step.MultiStep method), 157
- minimize() (tensorforce.core.optimizers.MultiStep method), 175
- minimize() (tensorforce.core.optimizers.natural_gradient.NaturalGradient method), 159
- minimize() (tensorforce.core.optimizers.NaturalGradient method), 173
- minimize() (tensorforce.core.optimizers.optimized_step.OptimizedStep method), 161
- minimize() (tensorforce.core.optimizers.OptimizedStep method), 177
- minimize() (tensorforce.core.optimizers.Optimizer method), 166
- minimize() (tensorforce.core.optimizers.optimizer.Optimizer method), 162
- minimize() (tensorforce.core.optimizers.SubsamplingStep method), 178
- minimize() (tensorforce.core.optimizers.Synchronization method), 180
- minimize() (tensorforce.core.optimizers.synchronization.Synchronization method), 163
- minimize() (tensorforce.core.optimizers.tf_optimizer.TFOptimizer method), 165
- minimize() (tensorforce.core.optimizers.TFOptimizer method), 170
- MLPBaseline (class in tensorforce.core.baselines), 90
- MLPBaseline (class in tensorforce.core.baselines.mlp_baseline), 86
- Model (class in tensorforce.models), 234
- Model (class in tensorforce.models.model), 196
- MsgPackNumpyProtocol (class in tensorforce.contrib.remote_environment), 79
- multi_config (tensorforce.tests.base_agent_test.BaseAgentTest attribute), 283
- multi_config (tensorforce.tests.test_constant_agent.TestConstantAgent attribute), 288
- multi_config (tensorforce.tests.test_dqfd_agent.TestDQFDAgent attribute), 294
- multi_config (tensorforce.tests.test_dqn_agent.TestDQNAgent attribute), 299
- multi_config (tensorforce.tests.test_dqn_nstep_agent.TestDQNNstepAgent attribute), 305
- multi_config (tensorforce.tests.test_naf_agent.TestNAFAgent attribute), 310
- multi_config (tensorforce.tests.test_ppo_agent.TestPPOAgent attribute), 316
- multi_config (tensorforce.tests.test_random_agent.TestRandomAgent attribute), 326
- multi_config (tensorforce.tests.test_trpo_agent.TestTRPOAgent attribute), 335
- multi_config (tensorforce.tests.test_vpg_agent.TestVPGAgent attribute), 346
- MultiStep (class in tensorforce.core.optimizers), 175
- MultiStep (class in tensorforce.core.optimizers.multi_step), 157
- ## N
- NaturalGradient (class in tensorforce.agents), 64
- NAFAgent (class in tensorforce.agents.naf_agent), 39
- NaturalGradient (class in tensorforce.core.optimizers), 172
- NaturalGradient (class in tensorforce.core.optimizers.natural_gradient), 158
- Network (class in tensorforce.core.networks), 140
- Network (class in tensorforce.core.networks.network), 128
- NetworkBaseline (class in tensorforce.core.baselines), 90
- NetworkBaseline (class in tensorforce.core.baselines.network_baseline), 87
- Nonlinearity (class in tensorforce.core.networks), 131
- Nonlinearity (class in tensorforce.core.networks.layer), 124
- np_dtype() (in module tensorforce.util), 358
- num_steps (tensorforce.contrib.deepmind_lab.DeepMindLab attribute), 76
- ## O
- observe() (tensorforce.agents.Agent method), 51
- observe() (tensorforce.agents.agent.Agent method), 26

[observe\(\)](#) ([tensorflow.agents.constant_agent.ConstantAgent](#)[observe\(\)](#) method), [28](#)
[observe\(\)](#) ([tensorflow.agents.ConstantAgent](#) method), [53](#)
[observe\(\)](#) ([tensorflow.agents.DDPGAgent](#) method), [73](#)
[observe\(\)](#) ([tensorflow.agents.dqfd_agent.DQFDAGent](#) method), [31](#)
[observe\(\)](#) ([tensorflow.agents.DQFDAGent](#) method), [59](#)
[observe\(\)](#) ([tensorflow.agents.dqn_agent.DQNAGent](#) method), [33](#)
[observe\(\)](#) ([tensorflow.agents.dqn_nstep_agent.DQNNstepAgent](#) method), [35](#)
[observe\(\)](#) ([tensorflow.agents.DQNAGent](#) method), [61](#)
[observe\(\)](#) ([tensorflow.agents.DQNNstepAgent](#) method), [63](#)
[observe\(\)](#) ([tensorflow.agents.learning_agent.LearningAgent](#)[observe\(\)](#) method), [38](#)
[observe\(\)](#) ([tensorflow.agents.LearningAgent](#) method), [57](#)
[observe\(\)](#) ([tensorflow.agents.naf_agent.NAFAGent](#) method), [40](#)
[observe\(\)](#) ([tensorflow.agents.NAFAGent](#) method), [65](#)
[observe\(\)](#) ([tensorflow.agents.ppo_agent.PPOAGent](#) method), [43](#)
[observe\(\)](#) ([tensorflow.agents.PPOAGent](#) method), [67](#)
[observe\(\)](#) ([tensorflow.agents.random_agent.RandomAgent](#) method), [45](#)
[observe\(\)](#) ([tensorflow.agents.RandomAgent](#) method), [54](#)
[observe\(\)](#) ([tensorflow.agents.trpo_agent.TRPOAGent](#) method), [47](#)
[observe\(\)](#) ([tensorflow.agents.TRPOAGent](#) method), [69](#)
[observe\(\)](#) ([tensorflow.agents.vpg_agent.VPGAGent](#) method), [49](#)
[observe\(\)](#) ([tensorflow.agents.VPGAGent](#) method), [71](#)
[observe\(\)](#) ([tensorflow.models.constant_model.ConstantModel](#) method), [190](#)
[observe\(\)](#) ([tensorflow.models.distribution_model.DistributionModel](#) method), [193](#)
[observe\(\)](#) ([tensorflow.models.DistributionModel](#) method), [245](#)
[observe\(\)](#) ([tensorflow.models.DPGTargetModel](#) method), [258](#)
[observe\(\)](#) ([tensorflow.models.MemoryModel](#) method), [239](#)
[observe\(\)](#) ([tensorflow.models.Model](#) method), [236](#)
[observe\(\)](#) ([tensorflow.models.model.Model](#) method), [198](#)
[observe\(\)](#) ([tensorflow.models.pg_log_prob_model.PGLogProbModel](#) method), [202](#)
[observe\(\)](#) ([tensorflow.models.pg_model.PGModel](#) method), [206](#)
[observe\(\)](#) ([tensorflow.models.pg_prob_ratio_model.PGProbRatioModel](#) method), [211](#)
[observe\(\)](#) ([tensorflow.models.PGLogProbModel](#) method), [262](#)
[observe\(\)](#) ([tensorflow.models.PGModel](#) method), [249](#)
[observe\(\)](#) ([tensorflow.models.PGProbRatioModel](#) method), [254](#)
[observe\(\)](#) ([tensorflow.models.q_demo_model.QDemoModel](#) method), [216](#)
[observe\(\)](#) ([tensorflow.models.q_model.QModel](#) method), [220](#)
[observe\(\)](#) ([tensorflow.models.q_naf_model.QNAFModel](#) method), [224](#)
[observe\(\)](#) ([tensorflow.models.q_nstep_model.QNstepModel](#) method), [229](#)
[observe\(\)](#) ([tensorflow.models.QDemoModel](#) method), [279](#)
[observe\(\)](#) ([tensorflow.models.QModel](#) method), [266](#)
[observe\(\)](#) ([tensorflow.models.QNAFModel](#) method), [274](#)
[observe\(\)](#) ([tensorflow.models.QNstepModel](#) method), [270](#)
[observe\(\)](#) ([tensorflow.models.random_model.RandomModel](#) method), [233](#)
[OpenAIGym](#) (class in [tensorflow.contrib.openai_gym](#)), [77](#)
[OpenAIUniverse](#) (class in [tensorflow.contrib.openai_universe](#)), [78](#)
[OptimizedStep](#) (class in [tensorflow.core.optimizers](#)), [176](#)
[OptimizedStep](#) (class in [tensorflow.core.optimizers.optimized_step](#)), [160](#)
[Optimizer](#) (class in [tensorflow.core.optimizers](#)), [166](#)
[Optimizer](#) (class in [tensorflow.core.optimizers.optimizer](#)), [162](#)
[optimizer_arguments\(\)](#) ([tensorflow.models.distribution_model.DistributionModel](#) method), [193](#)
[optimizer_arguments\(\)](#) ([tensorflow.models.DistributionModel](#) method), [245](#)
[optimizer_arguments\(\)](#) ([tensorflow.models.DPGTargetModel](#) method), [258](#)
[optimizer_arguments\(\)](#) ([tensorflow.models.MemoryModel](#) method), [240](#)
[optimizer_arguments\(\)](#) ([tensorflow.models.pg_log_prob_model.PGLogProbModel](#) method), [202](#)
[optimizer_arguments\(\)](#) ([tensorflow.models.pg_model.PGModel](#) method), [206](#)
[optimizer_arguments\(\)](#) ([tensorflow.models.pg_prob_ratio_model.PGProbRatioModel](#) method), [211](#)
[optimizer_arguments\(\)](#) ([tensorflow.models.PGLogProbModel](#) method), [262](#)
[optimizer_arguments\(\)](#) ([tensorflow.models.PGModel](#) method), [249](#)
[optimizer_arguments\(\)](#) ([tensorflow.models.PGProbRatioModel](#) method), [254](#)
[optimizer_arguments\(\)](#) ([tensorflow.models.q_demo_model.QDemoModel](#) method), [216](#)
[optimizer_arguments\(\)](#) ([tensorflow.models.q_model.QModel](#) method), [220](#)
[optimizer_arguments\(\)](#) ([tensorflow.models.q_naf_model.QNAFModel](#) method), [224](#)
[optimizer_arguments\(\)](#) ([tensorflow.models.q_nstep_model.QNstepModel](#) method), [229](#)
[optimizer_arguments\(\)](#) ([tensorflow.models.QDemoModel](#) method), [279](#)
[optimizer_arguments\(\)](#) ([tensorflow.models.QModel](#) method), [266](#)
[optimizer_arguments\(\)](#) ([tensorflow.models.QNAFModel](#) method), [274](#)
[optimizer_arguments\(\)](#) ([tensorflow.models.QNstepModel](#) method), [270](#)
[optimizer_arguments\(\)](#) ([tensorflow.models.random_model.RandomModel](#) method), [233](#)

[force.models.PGProbRatioModel](#) (method), [254](#)
[optimizer_arguments\(\)](#) ([tensorflow.models.q_demo_model.QDemoModel](#) method), [216](#)
[optimizer_arguments\(\)](#) ([tensorflow.models.q_model.QModel](#) method), [220](#)
[optimizer_arguments\(\)](#) ([tensorflow.models.q_naf_model.QNAFModel](#) method), [224](#)
[optimizer_arguments\(\)](#) ([tensorflow.models.q_nstep_model.QNstepModel](#) method), [229](#)
[optimizer_arguments\(\)](#) ([tensorflow.models.QDemoModel](#) method), [279](#)
[optimizer_arguments\(\)](#) ([tensorflow.models.QModel](#) method), [266](#)
[optimizer_arguments\(\)](#) ([tensorflow.models.QNAFModel](#) method), [275](#)
[optimizer_arguments\(\)](#) ([tensorflow.models.QNstepModel](#) method), [271](#)
[OrnsteinUhlenbeckProcess](#) (class in [tensorflow.core.explorations](#)), [104](#)
[OrnsteinUhlenbeckProcess](#) (class in [tensorflow.core.explorations.ornstein_uhlenbeck_process](#)), [102](#)
[Output](#) (class in [tensorflow.core.networks.complex_network](#)), [115](#)

P

[pass_threshold](#) ([tensorflow.tests.base_agent_test.BaseAgentTest](#) attribute), [283](#)
[pass_threshold](#) ([tensorflow.tests.base_test.BaseTest](#) attribute), [284](#)
[pass_threshold](#) ([tensorflow.tests.test_constant_agent.TestConstantAgent](#) attribute), [289](#)
[pass_threshold](#) ([tensorflow.tests.test_dqfd_agent.TestDQFDAGent](#) attribute), [294](#)
[pass_threshold](#) ([tensorflow.tests.test_dqn_agent.TestDQNAgent](#) attribute), [299](#)
[pass_threshold](#) ([tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent](#) attribute), [305](#)
[pass_threshold](#) ([tensorflow.tests.test_naf_agent.TestNAFAgent](#) attribute), [310](#)
[pass_threshold](#) ([tensorflow.tests.test_ppo_agent.TestPPOAgent](#) attribute), [316](#)
[pass_threshold](#) ([tensorflow.tests.test_random_agent.TestRandomAgent](#) attribute), [326](#)
[pass_threshold](#) ([tensorflow.tests.test_trpo_agent.TestTRPOAgent](#) attribute), [336](#)
[pass_threshold](#) ([tensorflow.tests.test_vpg_agent.TestVPAGent](#) attribute), [346](#)
[pass_threshold](#) ([tensorflow.tests.test_vpg_baselines.TestVPGBaselines](#) attribute), [351](#)
[pass_threshold](#) ([tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers](#) attribute), [356](#)
[PGLogProbModel](#) (class in [tensorflow.models](#)), [261](#)
[PGLogProbModel](#) (class in [tensorflow.models.pg_log_prob_model](#)), [200](#)
[PGModel](#) (class in [tensorflow.models](#)), [248](#)
[PGModel](#) (class in [tensorflow.models.pg_model](#)), [205](#)
[PGProbRatioModel](#) (class in [tensorflow.models](#)), [253](#)
[PGProbRatioModel](#) (class in [tensorflow.models.pg_prob_ratio_model](#)), [210](#)
[Pool2d](#) (class in [tensorflow.core.networks](#)), [133](#)
[Pool2d](#) (class in [tensorflow.core.networks.layer](#)), [125](#)
[PPOAgent](#) (class in [tensorflow.agents](#)), [66](#)
[PPOAgent](#) (class in [tensorflow.agents.ppo_agent](#)), [42](#)
[pre_run\(\)](#) ([tensorflow.tests.base_agent_test.BaseAgentTest](#) method), [283](#)
[pre_run\(\)](#) ([tensorflow.tests.base_test.BaseTest](#) method), [284](#)
[pre_run\(\)](#) ([tensorflow.tests.test_constant_agent.TestConstantAgent](#) method), [289](#)
[pre_run\(\)](#) ([tensorflow.tests.test_dqfd_agent.TestDQFDAGent](#) method), [294](#)
[pre_run\(\)](#) ([tensorflow.tests.test_dqn_agent.TestDQNAgent](#) method), [299](#)
[pre_run\(\)](#) ([tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent](#) method), [305](#)
[pre_run\(\)](#) ([tensorflow.tests.test_naf_agent.TestNAFAgent](#) method), [310](#)
[pre_run\(\)](#) ([tensorflow.tests.test_ppo_agent.TestPPOAgent](#) method), [316](#)
[pre_run\(\)](#) ([tensorflow.tests.test_random_agent.TestRandomAgent](#) method), [326](#)
[pre_run\(\)](#) ([tensorflow.tests.test_trpo_agent.TestTRPOAgent](#) method), [336](#)
[pre_run\(\)](#) ([tensorflow.tests.test_vpg_agent.TestVPAGent](#) method), [346](#)
[pre_run\(\)](#) ([tensorflow.tests.test_vpg_baselines.TestVPGBaselines](#) method), [351](#)
[pre_run\(\)](#) ([tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers](#) method), [356](#)
[prepare_kwargs\(\)](#) (in module [tensorflow.util](#)), [359](#)
[pretrain\(\)](#) ([tensorflow.agents.dqfd_agent.DQFDAGent](#) method), [31](#)
[pretrain\(\)](#) ([tensorflow.agents.DQFDAGent](#) method), [59](#)
[PrioritizedReplay](#) (class in [tensorflow.core.memories](#)), [112](#)
[PrioritizedReplay](#) (class in [tensorflow.core.memories.prioritized_replay](#)), [106](#)
[prod\(\)](#) (in module [tensorflow.util](#)), [359](#)

Q

QDemoModel (class in tensorforce.models), 277

QDemoModel (class in tensorforce.models.q_demo_model), 214

QModel (class in tensorforce.models), 265

QModel (class in tensorforce.models.q_model), 219

QNAFModel (class in tensorforce.models), 273

QNAFModel (class in tensorforce.models.q_naf_model), 223

QNstepModel (class in tensorforce.models), 269

QNstepModel (class in tensorforce.models.q_nstep_model), 227

Queue (class in tensorforce.core.memories), 109

R

RandomAgent (class in tensorforce.agents), 53

RandomAgent (class in tensorforce.agents.random_agent), 44

RandomModel (class in tensorforce.models.random_model), 231

rank() (in module tensorforce.util), 359

recv() (tensorforce.contrib.remote_environment.MsgPackNumpyProtocol method), 79

register_saver_ops() (tensorforce.util.SavableComponent method), 358

RemoteEnvironment (class in tensorforce.contrib.remote_environment), 79

render() (tensorforce.contrib.openai_universe.OpenAIGym method), 78

Replay (class in tensorforce.core.memories), 111

Replay (class in tensorforce.core.memories.replay), 107

requires_network (tensorforce.tests.base_agent_test.BaseAgentTest attribute), 283

requires_network (tensorforce.tests.base_test.BaseTest attribute), 284

requires_network (tensorforce.tests.test_constant_agent.TestConstantAgent attribute), 289

requires_network (tensorforce.tests.test_dqfd_agent.TestDQFDDAgent attribute), 294

requires_network (tensorforce.tests.test_dqn_agent.TestDQNAgent attribute), 300

requires_network (tensorforce.tests.test_dqn_nstep_agent.TestDQNNstepAgent attribute), 305

requires_network (tensorforce.tests.test_naf_agent.TestNAFAGent attribute), 310

requires_network (tensorforce.tests.test_ppo_agent.TestPPOAgent attribute), 316

requires_network (tensorforce.tests.test_random_agent.TestRandomAgent attribute), 326

requires_network (tensorforce.tests.test_trpo_agent.TestTRPOAgent attribute), 336

requires_network (tensorforce.tests.test_vpg_agent.TestVPGAgent attribute), 346

requires_network (tensorforce.tests.test_vpg_baselines.TestVPGBaselines attribute), 351

requires_network (tensorforce.tests.test_vpg_optimizers.TestVPGOptimizers attribute), 356

reset() (tensorforce.agents.Agent method), 51

reset() (tensorforce.agents.agent.Agent method), 27

reset() (tensorforce.agents.constant_agent.ConstantAgent method), 28

reset() (tensorforce.agents.ConstantAgent method), 53

reset() (tensorforce.agents.DDPGAgent method), 73

reset() (tensorforce.agents.dqfd_agent.DQFDDAgent method), 31

reset() (tensorforce.agents.DQFDDAgent method), 59

reset() (tensorforce.agents.dqn_agent.DQNAgent method), 33

reset() (tensorforce.agents.dqn_nstep_agent.DQNNstepAgent method), 36

reset() (tensorforce.agents.DQNAgent method), 61

reset() (tensorforce.agents.DQNNstepAgent method), 63

reset() (tensorforce.agents.learning_agent.LearningAgent method), 38

reset() (tensorforce.agents.LearningAgent method), 57

reset() (tensorforce.agents.naf_agent.NAFAGent method), 41

reset() (tensorforce.agents.NAFAGent method), 65

reset() (tensorforce.agents.ppo_agent.PPOAgent method), 43

reset() (tensorforce.agents.PPOAgent method), 67

reset() (tensorforce.agents.random_agent.RandomAgent method), 45

reset() (tensorforce.agents.RandomAgent method), 54

reset() (tensorforce.agents.trpo_agent.TRPOAgent method), 47

reset() (tensorforce.agents.TRPOAgent method), 69

reset() (tensorforce.agents.vpg_agent.VPGAgent method), 49

reset() (tensorforce.agents.VPGAgent method), 71

reset() (tensorforce.contrib.ale.ALE method), 75

reset() (tensorforce.contrib.deepmind_lab.DeepMindLab method), 76

reset() (tensorforce.contrib.maze_explorer.MazeExplorer method), 77

reset() (tensorforce.contrib.openai_gym.OpenAIGym

method), 78

reset() (tensorforce.contrib.openai_universe.OpenAIUniverse method), 78

reset() (tensorforce.contrib.remote_environment.RemoteEnvironment method), 80

reset() (tensorforce.contrib.state_settable_environment.StateSettableEnvironment method), 81

reset() (tensorforce.contrib.unreal_engine.UE4Environment method), 82

reset() (tensorforce.environments.Environment method), 182

reset() (tensorforce.environments.environment.Environment method), 182

reset() (tensorforce.environments.MinimalTest method), 183

reset() (tensorforce.execution.BaseRunner method), 186

reset() (tensorforce.execution.Runner method), 187

reset() (tensorforce.execution.runner.Runner method), 184

reset() (tensorforce.execution.threaded_runner.ThreadedRunner method), 185

reset() (tensorforce.execution.ThreadedRunner method), 188

reset() (tensorforce.models.constant_model.ConstantModel method), 190

reset() (tensorforce.models.distribution_model.DistributionModel method), 193

reset() (tensorforce.models.DistributionModel method), 245

reset() (tensorforce.models.DPGTargetModel method), 258

reset() (tensorforce.models.MemoryModel method), 240

reset() (tensorforce.models.Model method), 236

reset() (tensorforce.models.model.Model method), 199

reset() (tensorforce.models.pg_log_prob_model.PGLogProbModel method), 202

reset() (tensorforce.models.pg_model.PGModel method), 206

reset() (tensorforce.models.pg_prob_ratio_model.PGProbRatioModel method), 212

reset() (tensorforce.models.PGLogProbModel method), 262

reset() (tensorforce.models.PGModel method), 250

reset() (tensorforce.models.PGProbRatioModel method), 254

reset() (tensorforce.models.q_demo_model.QDemoModel method), 216

reset() (tensorforce.models.q_model.QModel method), 220

reset() (tensorforce.models.q_naf_model.QNAFModel method), 224

reset() (tensorforce.models.q_nstep_model.QNstepModel method), 229

reset() (tensorforce.models.QDemoModel method), 279

reset() (tensorforce.models.QModel method), 266

reset() (tensorforce.models.QNAFModel method), 275

reset() (tensorforce.models.QNstepModel method), 271

reset() (tensorforce.models.random_model.RandomModel method), 233

reset() (tensorforce.models.constant_model.ConstantModel method), 190

restore() (tensorforce.models.distribution_model.DistributionModel method), 193

restore() (tensorforce.models.DistributionModel method), 245

restore() (tensorforce.models.DPGTargetModel method), 258

restore() (tensorforce.models.MemoryModel method), 240

restore() (tensorforce.models.Model method), 236

restore() (tensorforce.models.model.Model method), 199

restore() (tensorforce.models.pg_log_prob_model.PGLogProbModel method), 202

restore() (tensorforce.models.pg_model.PGModel method), 207

restore() (tensorforce.models.pg_prob_ratio_model.PGProbRatioModel method), 212

restore() (tensorforce.models.PGLogProbModel method), 262

restore() (tensorforce.models.PGModel method), 250

restore() (tensorforce.models.PGProbRatioModel method), 254

restore() (tensorforce.models.q_demo_model.QDemoModel method), 216

restore() (tensorforce.models.q_model.QModel method), 220

restore() (tensorforce.models.q_naf_model.QNAFModel method), 225

restore() (tensorforce.models.q_nstep_model.QNstepModel method), 229

restore() (tensorforce.models.QDemoModel method), 279

restore() (tensorforce.models.QModel method), 267

restore() (tensorforce.models.QNAFModel method), 275

restore() (tensorforce.models.QNstepModel method), 271

restore() (tensorforce.models.random_model.RandomModel method), 233

restore() (tensorforce.util.SavableComponent method), 358

restore_component() (tensorforce.models.constant_model.ConstantModel method), 190

restore_component() (tensorforce.models.distribution_model.DistributionModel method), 193

restore_component() (tensorforce.models.DistributionModel method), 245

[restore_component\(\)](#) (tensorflow.models.DPGTargetModel method), [258](#)
[restore_component\(\)](#) (tensorflow.models.MemoryModel method), [240](#)
[restore_component\(\)](#) (tensorflow.models.Model method), [236](#)
[restore_component\(\)](#) (tensorflow.models.model.Model method), [199](#)
[restore_component\(\)](#) (tensorflow.models.pg_log_prob_model.PGLogProbModel method), [202](#)
[restore_component\(\)](#) (tensorflow.models.pg_model.PGModel method), [207](#)
[restore_component\(\)](#) (tensorflow.models.pg_prob_ratio_model.PGProbRatioModel method), [212](#)
[restore_component\(\)](#) (tensorflow.models.PGLogProbModel method), [263](#)
[restore_component\(\)](#) (tensorflow.models.PGModel method), [250](#)
[restore_component\(\)](#) (tensorflow.models.PGProbRatioModel method), [254](#)
[restore_component\(\)](#) (tensorflow.models.q_demo_model.QDemoModel method), [216](#)
[restore_component\(\)](#) (tensorflow.models.q_model.QModel method), [221](#)
[restore_component\(\)](#) (tensorflow.models.q_naf_model.QNAFModel method), [225](#)
[restore_component\(\)](#) (tensorflow.models.q_nstep_model.QNstepModel method), [229](#)
[restore_component\(\)](#) (tensorflow.models.QDemoModel method), [279](#)
[restore_component\(\)](#) (tensorflow.models.QModel method), [267](#)
[restore_component\(\)](#) (tensorflow.models.QNAFModel method), [275](#)
[restore_component\(\)](#) (tensorflow.models.QNstepModel method), [271](#)
[restore_component\(\)](#) (tensorflow.models.random_model.RandomModel method), [233](#)
[restore_model\(\)](#) (tensorflow.agents.Agent method), [51](#)
[restore_model\(\)](#) (tensorflow.agents.agent.Agent method), [27](#)
[restore_model\(\)](#) (tensorflow.agents.constant_agent.ConstantAgent method), [29](#)
[restore_model\(\)](#) (tensorflow.agents.ConstantAgent method), [53](#)
[restore_model\(\)](#) (tensorflow.agents.DDPGAgent method), [73](#)
[restore_model\(\)](#) (tensorflow.agents.dqfd_agent.DQFDAgent method), [31](#)
[restore_model\(\)](#) (tensorflow.agents.DQFDAgent method), [59](#)
[restore_model\(\)](#) (tensorflow.agents.dqn_agent.DQNAgent method), [33](#)
[restore_model\(\)](#) (tensorflow.agents.dqn_nstep_agent.DQNNstepAgent method), [36](#)
[restore_model\(\)](#) (tensorflow.agents.DQNAgent method), [61](#)
[restore_model\(\)](#) (tensorflow.agents.DQNNstepAgent method), [63](#)
[restore_model\(\)](#) (tensorflow.agents.learning_agent.LearningAgent method), [39](#)
[restore_model\(\)](#) (tensorflow.agents.LearningAgent method), [57](#)
[restore_model\(\)](#) (tensorflow.agents.naf_agent.NAFAgent method), [41](#)
[restore_model\(\)](#) (tensorflow.agents.NAFAgent method), [65](#)
[restore_model\(\)](#) (tensorflow.agents.ppo_agent.PPOAgent method), [43](#)
[restore_model\(\)](#) (tensorflow.agents.PPOAgent method), [67](#)
[restore_model\(\)](#) (tensorflow.agents.random_agent.RandomAgent method), [45](#)
[restore_model\(\)](#) (tensorflow.agents.RandomAgent method), [54](#)
[restore_model\(\)](#) (tensorflow.agents.trpo_agent.TRPOAgent method), [47](#)
[restore_model\(\)](#) (tensorflow.agents.TRPOAgent method), [69](#)
[restore_model\(\)](#) (tensorflow.agents.vpg_agent.VPGAgent method), [50](#)
[restore_model\(\)](#) (tensorflow.agents.VPGAgent method), [71](#)
[run\(\)](#) (tensorflow.execution.BaseRunner method), [186](#)
[run\(\)](#) (tensorflow.execution.Runner method), [187](#)
[run\(\)](#) (tensorflow.execution.runner.Runner method), [184](#)
[run\(\)](#) (tensorflow.execution.threaded_runner.ThreadedRunner method), [184](#)

method), 185

run() (tensorflow.execution.ThreadedRunner method), 188

run() (tensorflow.tests.test_constant_agent.TestConstantAgent method), 289

run() (tensorflow.tests.test_dqfd_agent.TestDQFDAGent method), 294

run() (tensorflow.tests.test_dqn_agent.TestDQNAGent method), 300

run() (tensorflow.tests.test_dqn_nstep_agent.TestDQNNStepAgent method), 305

run() (tensorflow.tests.test_naf_agent.TestNAFAGent method), 310

run() (tensorflow.tests.test_ppo_agent.TestPPOAgent method), 316

run() (tensorflow.tests.test_quickstart_example.TestQuickstartExample method), 321

run() (tensorflow.tests.test_random_agent.TestRandomAgent method), 326

run() (tensorflow.tests.test_reward_estimation.TestRewardEstimation method), 330

run() (tensorflow.tests.test_trpo_agent.TestTRPOAgent method), 336

run() (tensorflow.tests.test_tutorial_code.TestTutorialCode method), 340

run() (tensorflow.tests.test_vpg_agent.TestVPGAgent method), 346

run() (tensorflow.tests.test_vpg_baselines.TestVPGBaselines method), 351

run() (tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers method), 356

Runner (class in tensorflow.execution), 187

Runner (class in tensorflow.execution.runner), 183

S

SavableComponent (class in tensorflow.util), 358

save() (tensorflow.models.constant_model.ConstantModel method), 190

save() (tensorflow.models.distribution_model.DistributionModel method), 193

save() (tensorflow.models.DistributionModel method), 245

save() (tensorflow.models.DPGTargetModel method), 258

save() (tensorflow.models.MemoryModel method), 240

save() (tensorflow.models.Model method), 236

save() (tensorflow.models.model.Model method), 199

save() (tensorflow.models.pg_log_prob_model.PGLogProbModel method), 202

save() (tensorflow.models.pg_model.PGModel method), 207

save() (tensorflow.models.pg_prob_ratio_model.PGProbRatioModel method), 212

save() (tensorflow.models.PGLogProbModel method), 263

save() (tensorflow.models.PGModel method), 250

save() (tensorflow.models.PGProbRatioModel method), 255

save() (tensorflow.models.PGLogProbModel method), 263

save() (tensorflow.models.PGModel method), 250

save() (tensorflow.models.PGProbRatioModel method), 255

save() (tensorflow.models.q_demo_model.QDemoModel method), 216

save() (tensorflow.models.q_model.QModel method), 221

save() (tensorflow.models.q_naf_model.QNAFModel method), 225

save() (tensorflow.models.q_nstep_model.QNstepModel method), 229

save() (tensorflow.models.QDemoModel method), 279

save() (tensorflow.models.QModel method), 267

save() (tensorflow.models.QNAFModel method), 275

save() (tensorflow.models.QNstepModel method), 271

save() (tensorflow.models.random_model.RandomModel method), 233

save_component() (tensorflow.util.SavableComponent method), 358

save_component() (tensorflow.models.constant_model.ConstantModel method), 191

save_component() (tensorflow.models.distribution_model.DistributionModel method), 193

save_component() (tensorflow.models.DistributionModel method), 245

save_component() (tensorflow.models.DPGTargetModel method), 259

save_component() (tensorflow.models.MemoryModel method), 240

save_component() (tensorflow.models.Model method), 237

save_component() (tensorflow.models.model.Model method), 199

save_component() (tensorflow.models.pg_log_prob_model.PGLogProbModel method), 203

save_component() (tensorflow.models.pg_model.PGModel method), 207

save_component() (tensorflow.models.pg_prob_ratio_model.PGProbRatioModel method), 212

save_component() (tensorflow.models.PGLogProbModel method), 263

save_component() (tensorflow.models.PGModel method), 250

save_component() (tensorflow.models.PGProbRatioModel method), 255

`save_component()` (tensorforce.models.q_demo_model.QDemoModel method), 216
`save_component()` (tensorforce.models.q_model.QModel method), 221
`save_component()` (tensorforce.models.q_naf_model.QNAFModel method), 225
`save_component()` (tensorforce.models.q_nstep_model.QNStepModel method), 229
`save_component()` (tensorforce.models.QDemoModel method), 280
`save_component()` (tensorforce.models.QModel method), 267
`save_component()` (tensorforce.models.QNAFModel method), 275
`save_component()` (tensorforce.models.QNStepModel method), 271
`save_component()` (tensorforce.models.random_model.RandomModel method), 233
`save_model()` (tensorforce.agents.Agent method), 51
`save_model()` (tensorforce.agents.agent.Agent method), 27
`save_model()` (tensorforce.agents.constant_agent.ConstantAgent method), 29
`save_model()` (tensorforce.agents.ConstantAgent method), 53
`save_model()` (tensorforce.agents.DDPGAgent method), 73
`save_model()` (tensorforce.agents.dqfd_agent.DQFDAgent method), 32
`save_model()` (tensorforce.agents.DQFDAgent method), 59
`save_model()` (tensorforce.agents.dqn_agent.DQNAgent method), 34
`save_model()` (tensorforce.agents.dqn_nstep_agent.DQNNStepAgent method), 36
`save_model()` (tensorforce.agents.DQNAgent method), 61
`save_model()` (tensorforce.agents.DQNNStepAgent method), 63
`save_model()` (tensorforce.agents.learning_agent.LearningAgent method), 39
`save_model()` (tensorforce.agents.LearningAgent method), 57
`save_model()` (tensorforce.agents.naf_agent.NAFAgent method), 41
`save_model()` (tensorforce.agents.NAFAgent method), 65
`save_model()` (tensorforce.agents.ppo_agent.PPOAgent method), 43
`save_model()` (tensorforce.agents.PPOAgent method), 67
`save_model()` (tensorforce.agents.random_agent.RandomAgent method), 45
`save_model()` (tensorforce.agents.RandomAgent method), 55
`save_model()` (tensorforce.agents.trpo_agent.TRPOAgent method), 48
`save_model()` (tensorforce.agents.TRPOAgent method), 70
`save_model()` (tensorforce.agents.vpg_agent.VPGAgent method), 50
`save_model()` (tensorforce.agents.VPGAgent method), 72
`seed()` (tensorforce.contrib.ale.ALE method), 75
`seed()` (tensorforce.contrib.deepmind_lab.DeepMindLab method), 76
`seed()` (tensorforce.contrib.maze_explorer.MazeExplorer method), 77
`seed()` (tensorforce.contrib.openai_gym.OpenAIGym method), 78
`seed()` (tensorforce.contrib.openai_universe.OpenAIUniverse method), 78
`seed()` (tensorforce.contrib.remote_environment.RemoteEnvironment method), 80
`seed()` (tensorforce.contrib.state_settable_environment.StateSettableEnvironment method), 81
`seed()` (tensorforce.contrib.unreal_engine.UE4Environment method), 83
`seed()` (tensorforce.environments.Environment method), 182
`seed()` (tensorforce.environments.environment.Environment method), 182
`seed()` (tensorforce.environments.MinimalTest method), 183
`send()` (tensorforce.contrib.remote_environment.MsgPackNumpyProtocol method), 79
`set_named_tensor()` (tensorforce.core.networks.complex_network.ComplexLayeredNetwork method), 114
`set_named_tensor()` (tensorforce.core.networks.LayerBasedNetwork method), 141
`set_named_tensor()` (tensorforce.core.networks.LayeredNetwork method), 142
`set_named_tensor()` (tensorforce.core.networks.Network method), 140
`set_named_tensor()` (tensorforce.core.networks.network.LayerBasedNetwork method), 127
`set_named_tensor()` (tensorforce.core.networks.network.LayeredNetwork method), 128
`set_named_tensor()` (tensorforce.core.networks.network.Network method), 129
`set_normalized_actions()` (tensorforce.agents.Agent method), 43

method), 52

set_normalized_actions() (tensorflow.agents.agent.Agent method), 27

set_normalized_actions() (tensorflow.agents.constant_agent.ConstantAgent method), 29

set_normalized_actions() (tensorflow.agents.ConstantAgent method), 53

set_normalized_actions() (tensorflow.agents.DDPGAgent method), 74

set_normalized_actions() (tensorflow.agents.dqfd_agent.DQFDDAgent method), 32

set_normalized_actions() (tensorflow.agents.DQFDDAgent method), 60

set_normalized_actions() (tensorflow.agents.dqn_agent.DQNAgent method), 34

set_normalized_actions() (tensorflow.agents.dqn_nstep_agent.DQNNstepAgent method), 36

set_normalized_actions() (tensorflow.agents.DQNAgent method), 62

set_normalized_actions() (tensorflow.agents.DQNNstepAgent method), 64

set_normalized_actions() (tensorflow.agents.learning_agent.LearningAgent method), 39

set_normalized_actions() (tensorflow.agents.LearningAgent method), 57

set_normalized_actions() (tensorflow.agents.naf_agent.NAFAgent method), 41

set_normalized_actions() (tensorflow.agents.NAFAgent method), 65

set_normalized_actions() (tensorflow.agents.ppo_agent.PPOAgent method), 44

set_normalized_actions() (tensorflow.agents.PPOAgent method), 68

set_normalized_actions() (tensorflow.agents.random_agent.RandomAgent method), 45

set_normalized_actions() (tensorflow.agents.RandomAgent method), 55

set_normalized_actions() (tensorflow.agents.trpo_agent.TRPOAgent method), 48

set_normalized_actions() (tensorflow.agents.TRPOAgent method), 70

set_normalized_actions() (tensorflow.agents.vpg_agent.VPGAgent method), 50

set_normalized_actions() (tensorflow.agents.VPGAgent method), 72

set_normalized_states() (tensorflow.agents.Agent method), 52

set_normalized_states() (tensorflow.agents.agent.Agent method), 27

set_normalized_states() (tensorflow.agents.constant_agent.ConstantAgent method), 29

set_normalized_states() (tensorflow.agents.ConstantAgent method), 53

set_normalized_states() (tensorflow.agents.DDPGAgent method), 74

set_normalized_states() (tensorflow.agents.dqfd_agent.DQFDDAgent method), 32

set_normalized_states() (tensorflow.agents.DQFDDAgent method), 60

set_normalized_states() (tensorflow.agents.dqn_agent.DQNAgent method), 34

set_normalized_states() (tensorflow.agents.dqn_nstep_agent.DQNNstepAgent method), 36

set_normalized_states() (tensorflow.agents.DQNAgent method), 62

set_normalized_states() (tensorflow.agents.DQNNstepAgent method), 64

set_normalized_states() (tensorflow.agents.learning_agent.LearningAgent method), 39

set_normalized_states() (tensorflow.agents.LearningAgent method), 57

set_normalized_states() (tensorflow.agents.naf_agent.NAFAgent method), 41

set_normalized_states() (tensorflow.agents.NAFAgent method), 65

set_normalized_states() (tensorflow.agents.ppo_agent.PPOAgent method), 44

set_normalized_states() (tensorflow.agents.PPOAgent method), 68

set_normalized_states() (tensorflow.agents.random_agent.RandomAgent method), 45

set_normalized_states() (tensorflow.agents.RandomAgent method), 55

set_normalized_states() (tensorflow.agents.trpo_agent.TRPOAgent method), 48

set_normalized_states() (tensorflow.agents.TRPOAgent method), 70

`set_normalized_states()` (tensorflow.agents.vpg_agent.VPGAgent method), 50
`set_normalized_states()` (tensorflow.agents.VPGAgent method), 72
`set_state()` (tensorflow.contrib.state.settable_environment.StateSettableEnvironment method), 81
`set_state()` (tensorflow.contrib.unreal_engine.UEnv4Environment method), 83
`setup()` (tensorflow.models.constant_model.ConstantModel method), 191
`setup()` (tensorflow.models.distribution_model.DistributionModel method), 193
`setup()` (tensorflow.models.DistributionModel method), 246
`setup()` (tensorflow.models.DPGTargetModel method), 259
`setup()` (tensorflow.models.MemoryModel method), 241
`setup()` (tensorflow.models.Model method), 237
`setup()` (tensorflow.models.model.Model method), 199
`setup()` (tensorflow.models.pg_log_prob_model.PGLogProbModel method), 203
`setup()` (tensorflow.models.pg_model.PGModel method), 207
`setup()` (tensorflow.models.pg_prob_ratio_model.PGProbRatioModel method), 212
`setup()` (tensorflow.models.PGLogProbModel method), 263
`setup()` (tensorflow.models.PGModel method), 250
`setup()` (tensorflow.models.PGProbRatioModel method), 255
`setup()` (tensorflow.models.q_demo_model.QDemoModel method), 217
`setup()` (tensorflow.models.q_model.QModel method), 221
`setup()` (tensorflow.models.q_naf_model.QNAFModel method), 225
`setup()` (tensorflow.models.q_nstep_model.QNstepModel method), 229
`setup()` (tensorflow.models.QDemoModel method), 280
`setup()` (tensorflow.models.QModel method), 267
`setup()` (tensorflow.models.QNAFModel method), 275
`setup()` (tensorflow.models.QNstepModel method), 271
`setup()` (tensorflow.models.random_model.RandomModel method), 234
`setUp()` (tensorflow.tests.test_constant_agent.TestConstantAgent method), 289
`setUp()` (tensorflow.tests.test_dqfd_agent.TestDQFDAgent method), 294
`setUp()` (tensorflow.tests.test_dqn_agent.TestDQNAgent method), 300
`setUp()` (tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 305
`setUp()` (tensorflow.tests.test_naf_agent.TestNAFAgent method), 310
`setUp()` (tensorflow.tests.test_ppo_agent.TestPPOAgent method), 316
`setUp()` (tensorflow.tests.test_quickstart_example.TestQuickstartExample method), 321
`setUp()` (tensorflow.tests.test_random_agent.TestRandomAgent method), 326
`setUp()` (tensorflow.tests.test_reward_estimation.TestRewardEstimation method), 330
`setUp()` (tensorflow.tests.test_trpo_agent.TestTRPOAgent method), 336
`setUp()` (tensorflow.tests.test_tutorial_code.TestTutorialCode method), 340
`setUp()` (tensorflow.tests.test_vpg_agent.TestVPGAgent method), 346
`setUp()` (tensorflow.tests.test_vpg_baselines.TestVPGBaselines method), 351
`setUp()` (tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers method), 356
`setUpClass()` (tensorflow.tests.test_constant_agent.TestConstantAgent method), 289
`setUpClass()` (tensorflow.tests.test_dqfd_agent.TestDQFDAgent method), 294
`setUpClass()` (tensorflow.tests.test_dqn_agent.TestDQNAgent method), 300
`setUpClass()` (tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 305
`setUpClass()` (tensorflow.tests.test_naf_agent.TestNAFAgent method), 311
`setUpClass()` (tensorflow.tests.test_ppo_agent.TestPPOAgent method), 316
`setUpClass()` (tensorflow.tests.test_quickstart_example.TestQuickstartExample method), 321
`setUpClass()` (tensorflow.tests.test_random_agent.TestRandomAgent method), 326
`setUpClass()` (tensorflow.tests.test_reward_estimation.TestRewardEstimation method), 330
`setUpClass()` (tensorflow.tests.test_trpo_agent.TestTRPOAgent method), 336
`setUpClass()` (tensorflow.tests.test_tutorial_code.TestTutorialCode method), 340
`setUpClass()` (tensorflow.tests.test_vpg_agent.TestVPGAgent method), 346
`setUpClass()` (tensorflow.tests.test_vpg_baselines.TestVPGBaselines method), 351
`setUpClass()` (tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers method), 356
`shape()` (in module tensorflow.util), 359
`shortDescription()` (tensorflow.tests.test_constant_agent.TestConstantAgent method), 289
`shortDescription()` (tensorflow.tests.test_dqfd_agent.TestDQFDAgent method), 294

[shortDescription\(\)](#) (tensorflow.tests.test_dqn_agent.TestDQNAgent method), 300
[shortDescription\(\)](#) (tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 305
[shortDescription\(\)](#) (tensorflow.tests.test_naf_agent.TestNAFAgent method), 311
[shortDescription\(\)](#) (tensorflow.tests.test_ppo_agent.TestPPOAgent method), 316
[shortDescription\(\)](#) (tensorflow.tests.test_quickstart_example.TestQuickstartExample method), 321
[shortDescription\(\)](#) (tensorflow.tests.test_random_agent.TestRandomAgent method), 326
[shortDescription\(\)](#) (tensorflow.tests.test_reward_estimation.TestRewardEstimation method), 331
[shortDescription\(\)](#) (tensorflow.tests.test_trpo_agent.TestTRPOAgent method), 336
[shortDescription\(\)](#) (tensorflow.tests.test_tutorial_code.TestTutorialCode method), 341
[shortDescription\(\)](#) (tensorflow.tests.test_vpg_agent.TestVPGAgent method), 346
[shortDescription\(\)](#) (tensorflow.tests.test_vpg_baselines.TestVPGBaselines method), 351
[shortDescription\(\)](#) (tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers method), 356
[should_stop\(\)](#) (tensorflow.agents.Agent method), 52
[should_stop\(\)](#) (tensorflow.agents.agent.Agent method), 27
[should_stop\(\)](#) (tensorflow.agents.constant_agent.ConstantAgent method), 29
[should_stop\(\)](#) (tensorflow.agents.ConstantAgent method), 53
[should_stop\(\)](#) (tensorflow.agents.DDPGAgent method), 74
[should_stop\(\)](#) (tensorflow.agents.dqfd_agent.DQFDAgent method), 32
[should_stop\(\)](#) (tensorflow.agents.DQFDAgent method), 60
[should_stop\(\)](#) (tensorflow.agents.dqn_agent.DQNAgent method), 34
[should_stop\(\)](#) (tensorflow.agents.dqn_nstep_agent.DQNNstepAgent method), 36
[should_stop\(\)](#) (tensorflow.agents.DQNAgent method), 62
[should_stop\(\)](#) (tensorflow.agents.DQNNstepAgent method), 64
[should_stop\(\)](#) (tensorflow.agents.learning_agent.LearningAgent method), 39
[should_stop\(\)](#) (tensorflow.agents.LearningAgent method), 57
[should_stop\(\)](#) (tensorflow.agents.naf_agent.NAFAgent method), 41
[should_stop\(\)](#) (tensorflow.agents.NAFAgent method), 66
[should_stop\(\)](#) (tensorflow.agents.ppo_agent.PPOAgent method), 44
[should_stop\(\)](#) (tensorflow.agents.PPOAgent method), 68
[should_stop\(\)](#) (tensorflow.agents.random_agent.RandomAgent method), 45
[should_stop\(\)](#) (tensorflow.agents.RandomAgent method), 55
[should_stop\(\)](#) (tensorflow.agents.trpo_agent.TRPOAgent method), 48
[should_stop\(\)](#) (tensorflow.agents.TRPOAgent method), 70
[should_stop\(\)](#) (tensorflow.agents.vpg_agent.VPGAgent method), 50
[should_stop\(\)](#) (tensorflow.agents.VPGAgent method), 72
[SingleRunner](#) (in module tensorflow.execution), 187
[SingleRunner](#) (in module tensorflow.execution.runner), 184
[skipTest\(\)](#) (tensorflow.tests.test_constant_agent.TestConstantAgent method), 289
[skipTest\(\)](#) (tensorflow.tests.test_dqfd_agent.TestDQFDAgent method), 294
[skipTest\(\)](#) (tensorflow.tests.test_dqn_agent.TestDQNAgent method), 300
[skipTest\(\)](#) (tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 305
[skipTest\(\)](#) (tensorflow.tests.test_naf_agent.TestNAFAgent method), 311
[skipTest\(\)](#) (tensorflow.tests.test_ppo_agent.TestPPOAgent method), 316
[skipTest\(\)](#) (tensorflow.tests.test_quickstart_example.TestQuickstartExample method), 321
[skipTest\(\)](#) (tensorflow.tests.test_random_agent.TestRandomAgent method), 326
[skipTest\(\)](#) (tensorflow.tests.test_reward_estimation.TestRewardEstimation method), 331
[skipTest\(\)](#) (tensorflow.tests.test_trpo_agent.TestTRPOAgent method), 336
[skipTest\(\)](#) (tensorflow.tests.test_tutorial_code.TestTutorialCode method), 341
[skipTest\(\)](#) (tensorflow.tests.test_vpg_agent.TestVPGAgent method), 346
[skipTest\(\)](#) (tensorflow.tests.test_vpg_baselines.TestVPGBaselines method), 351

`skipTest()` (tensorforce.tests.test_vpg_optimizers.TestVPGOptimizers attribute), 182
method), 356

`Solver` (class in tensorforce.core.optimizers.solvers), 147

`Solver` (class in tensorforce.core.optimizers.solvers.solver), 147

`state_action_value()` (tensorforce.core.distributions.Bernoulli method), 98

`state_action_value()` (tensorforce.core.distributions.bernoulli.Bernoulli method), 92

`state_action_value()` (tensorforce.core.distributions.Categorical method), 98

`state_action_value()` (tensorforce.core.distributions.categorical.Categorical method), 94

`state_action_value()` (tensorforce.core.distributions.Gaussian method), 99

`state_action_value()` (tensorforce.core.distributions.gaussian.Gaussian method), 96

`state_from_space()` (tensorforce.contrib.openai_gym.OpenAIGym static method), 78

`state_value()` (tensorforce.core.distributions.Bernoulli method), 98

`state_value()` (tensorforce.core.distributions.bernoulli.Bernoulli method), 92

`state_value()` (tensorforce.core.distributions.Categorical method), 98

`state_value()` (tensorforce.core.distributions.categorical.Categorical method), 94

`state_value()` (tensorforce.core.distributions.Gaussian method), 99

`state_value()` (tensorforce.core.distributions.gaussian.Gaussian method), 96

`states` (tensorforce.contrib.ale.ALE attribute), 75

`states` (tensorforce.contrib.deepmind_lab.DeepMindLab attribute), 76

`states` (tensorforce.contrib.maze_explorer.MazeExplorer attribute), 77

`states` (tensorforce.contrib.openai_gym.OpenAIGym attribute), 78

`states` (tensorforce.contrib.openai_universe.OpenAIUniverse attribute), 78

`states` (tensorforce.contrib.remote_environment.RemoteEnvironment attribute), 80

`states` (tensorforce.contrib.state_settable_environment.StateSettableEnvironment attribute), 81

`states` (tensorforce.environments.Environment attribute), 183

`states` (tensorforce.environments.environment.Environment attribute), 183

`states()` (tensorforce.contrib.unreal_engine.UE4Environment method), 83

`StateSettableEnvironment` (class in tensorforce.contrib.state_settable_environment), 80

`strip_name_scope()` (in module tensorforce.util), 359

`SubsamplingStep` (class in tensorforce.core.optimizers), 178

`Synchronization` (class in tensorforce.core.optimizers), 179

`Synchronization` (class in tensorforce.core.optimizers.synchronization), 163

T

`target_optimizer_arguments()` (tensorforce.models.q_demo_model.QDemoModel method), 217

`target_optimizer_arguments()` (tensorforce.models.q_model.QModel method), 221

`target_optimizer_arguments()` (tensorforce.models.q_naf_model.QNAFModel method), 225

`target_optimizer_arguments()` (tensorforce.models.q_nstep_model.QNstepModel method), 230

`target_optimizer_arguments()` (tensorforce.models.QDemoModel method), 280

`target_optimizer_arguments()` (tensorforce.models.QModel method), 267

`target_optimizer_arguments()` (tensorforce.models.QNAFModel method), 275

`target_optimizer_arguments()` (tensorforce.models.QNstepModel method), 271

`tearDown()` (tensorforce.tests.test_constant_agent.TestConstantAgent method), 289

`tearDown()` (tensorforce.tests.test_dqfd_agent.TestDQFDAgent method), 294

`tearDown()` (tensorforce.tests.test_dqn_agent.TestDQNAgent method), 300

`tearDown()` (tensorforce.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 305

`tearDown()` (tensorforce.tests.test_naf_agent.TestNAFAgent method), 311

`tearDown()` (tensorforce.tests.test_ppo_agent.TestPPOAgent method), 316

`tearDown()` (tensorforce.tests.test_quickstart_example.TestQuickstartExample method), 321

`tearDown()` (tensorforce.tests.test_random_agent.TestRandomAgent method), 326

[tearDown\(\) \(tensorflow.tests.test_reward_estimation.TestRewardEstimation method\), 331](#)
[tearDown\(\) \(tensorflow.tests.test_trpo_agent.TestTRPOAgent method\), 336](#)
[tearDown\(\) \(tensorflow.tests.test_tutorial_code.TestTutorialCode method\), 341](#)
[tearDown\(\) \(tensorflow.tests.test_vpg_agent.TestVPGAgent method\), 346](#)
[tearDown\(\) \(tensorflow.tests.test_vpg_baselines.TestVPGBaselines method\), 351](#)
[tearDown\(\) \(tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers method\), 356](#)
[tearDownClass\(\) \(tensorflow.tests.test_constant_agent.TestConstantAgent method\), 289](#)
[tearDownClass\(\) \(tensorflow.tests.test_dqfd_agent.TestDQFDAgent method\), 294](#)
[tearDownClass\(\) \(tensorflow.tests.test_dqn_agent.TestDQNAgent method\), 300](#)
[tearDownClass\(\) \(tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent method\), 305](#)
[tearDownClass\(\) \(tensorflow.tests.test_naf_agent.TestNAFAgent method\), 311](#)
[tearDownClass\(\) \(tensorflow.tests.test_ppo_agent.TestPPOAgent method\), 316](#)
[tearDownClass\(\) \(tensorflow.tests.test_quickstart_example.TestQuickstartExample method\), 321](#)
[tearDownClass\(\) \(tensorflow.tests.test_random_agent.TestRandomAgent method\), 326](#)
[tearDownClass\(\) \(tensorflow.tests.test_reward_estimation.TestRewardEstimation method\), 331](#)
[tearDownClass\(\) \(tensorflow.tests.test_trpo_agent.TestTRPOAgent method\), 336](#)
[tearDownClass\(\) \(tensorflow.tests.test_tutorial_code.TestTutorialCode method\), 341](#)
[tearDownClass\(\) \(tensorflow.tests.test_vpg_agent.TestVPGAgent method\), 346](#)
[tearDownClass\(\) \(tensorflow.tests.test_vpg_baselines.TestVPGBaselines method\), 351](#)
[tearDownClass\(\) \(tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers method\), 356](#)
[tensorflow \(module\), 50](#)
[tensorflow.agents \(module\), 26](#)
[tensorflow.agents.constant_agent \(module\), 27](#)
[tensorflow.agents.dqfd_agent \(module\), 30](#)
[tensorflow.agents.dqn_agent \(module\), 32](#)
[tensorflow.agents.dqn_nstep_agent \(module\), 34](#)
[tensorflow.agents.learning_agent \(module\), 37](#)
[tensorflow.agents.naf_agent \(module\), 39](#)
[tensorflow.agents.ppo_agent \(module\), 42](#)
[tensorflow.agents.random_agent \(module\), 44](#)
[tensorflow.agents.trpo_agent \(module\), 46](#)
[tensorflow.agents.vpg_agent \(module\), 48](#)
[tensorflow.contrib \(module\), 83](#)
[tensorflow.contrib.ale \(module\), 74](#)
[tensorflow.contrib.deepmind_lab \(module\), 75](#)
[tensorflow.contrib.maze_explorer \(module\), 76](#)
[tensorflow.contrib.openai_gym \(module\), 77](#)
[tensorflow.contrib.openai_universe \(module\), 78](#)
[tensorflow.contrib.remote_environment \(module\), 79](#)
[tensorflow.contrib.state_settable_environment \(module\), 80](#)
[tensorflow.contrib.unreal_engine \(module\), 81](#)
[tensorflow.core \(module\), 181](#)
[tensorflow.core.baselines \(module\), 88](#)
[tensorflow.core.baselines.aggregated_baseline \(module\), 83](#)
[tensorflow.core.baselines.baseline \(module\), 84](#)
[tensorflow.core.baselines.cnn_baseline \(module\), 85](#)
[tensorflow.core.baselines.mlp_baseline \(module\), 86](#)
[tensorflow.core.baselines.network_baseline \(module\), 87](#)
[tensorflow.core.distributions \(module\), 96](#)
[tensorflow.core.distributions.bernoulli \(module\), 92](#)
[tensorflow.core.distributions.beta \(module\), 93](#)
[tensorflow.core.distributions.categorical \(module\), 94](#)
[tensorflow.core.distributions.distribution \(module\), 94](#)
[tensorflow.core.distributions.gaussian \(module\), 96](#)
[tensorflow.core.explorations \(module\), 102](#)
[tensorflow.core.explorations.constant \(module\), 100](#)
[tensorflow.core.explorations.epsilon_anneal \(module\), 100](#)
[tensorflow.core.explorations.epsilon_decay \(module\), 101](#)
[tensorflow.core.explorations.exploration \(module\), 101](#)
[tensorflow.core.explorations.ornstein_uhlenbeck_process \(module\), 102](#)
[tensorflow.core.memories \(module\), 108](#)
[tensorflow.core.memories.memory \(module\), 104](#)
[tensorflow.core.memories.prioritized_replay \(module\), 106](#)
[tensorflow.core.memories.replay \(module\), 107](#)
[tensorflow.core.networks \(module\), 129](#)
[tensorflow.core.networks.complex_network \(module\), 114](#)

tensorforce.core.networks.layer (module), 116
tensorforce.core.networks.network (module), 127
tensorforce.core.optimizers (module), 166
tensorforce.core.optimizers.clipped_step (module), 151
tensorforce.core.optimizers.evolutionary (module), 153
tensorforce.core.optimizers.global_optimizer (module), 154
tensorforce.core.optimizers.meta_optimizer (module), 156
tensorforce.core.optimizers.multi_step (module), 157
tensorforce.core.optimizers.natural_gradient (module), 158
tensorforce.core.optimizers.optimized_step (module), 160
tensorforce.core.optimizers.optimizer (module), 162
tensorforce.core.optimizers.solvers (module), 147
tensorforce.core.optimizers.solvers.conjugate_gradient (module), 142
tensorforce.core.optimizers.solvers.iterative (module), 144
tensorforce.core.optimizers.solvers.line_search (module), 145
tensorforce.core.optimizers.solvers.solver (module), 147
tensorforce.core.optimizers.synchronization (module), 163
tensorforce.core.optimizers.tf_optimizer (module), 164
tensorforce.environments (module), 182
tensorforce.environments.environment (module), 181
tensorforce.exception (module), 357
tensorforce.execution (module), 186
tensorforce.execution.runner (module), 183
tensorforce.execution.threaded_runner (module), 184
tensorforce.meta_parameter_recorder (module), 357
tensorforce.models (module), 234
tensorforce.models.constant_model (module), 188
tensorforce.models.distribution_model (module), 191
tensorforce.models.model (module), 196
tensorforce.models.pg_log_prob_model (module), 200
tensorforce.models.pg_model (module), 205
tensorforce.models.pg_prob_ratio_model (module), 210
tensorforce.models.q_demo_model (module), 214
tensorforce.models.q_model (module), 219
tensorforce.models.q_naf_model (module), 223
tensorforce.models.q_nstep_model (module), 227
tensorforce.models.random_model (module), 231
tensorforce.tests (module), 357
tensorforce.tests.base_agent_test (module), 282
tensorforce.tests.base_test (module), 283
tensorforce.tests.test_constant_agent (module), 284
tensorforce.tests.test_dqfd_agent (module), 289
tensorforce.tests.test_dqn_agent (module), 295
tensorforce.tests.test_dqn_nstep_agent (module), 300
tensorforce.tests.test_naf_agent (module), 306
tensorforce.tests.test_ppo_agent (module), 311
tensorforce.tests.test_quickstart_example (module), 317
tensorforce.tests.test_random_agent (module), 321
tensorforce.tests.test_reward_estimation (module), 326
tensorforce.tests.test_trpo_agent (module), 331
tensorforce.tests.test_tutorial_code (module), 336
tensorforce.tests.test_vpg_agent (module), 341
tensorforce.tests.test_vpg_baselines (module), 346
tensorforce.tests.test_vpg_optimizers (module), 351
tensorforce.util (module), 358
TensorForceError, 357, 359
test_adam() (tensorforce.tests.test_vpg_optimizers.TestVPGOptimizers method), 356
test_baseline() (tensorforce.tests.test_reward_estimation.TestRewardEstimation method), 331
test_baseline_no_optimizer() (tensorforce.tests.test_vpg_baselines.TestVPGBaselines method), 351
test_basic() (tensorforce.tests.test_reward_estimation.TestRewardEstimation method), 331
test_blogpost_introduction() (tensorforce.tests.test_tutorial_code.TestTutorialCode method), 341
test_blogpost_introduction_runner() (tensorforce.tests.test_tutorial_code.TestTutorialCode method), 341
test_bool() (tensorforce.tests.base_agent_test.BaseAgentTest method), 283
test_bool() (tensorforce.tests.test_constant_agent.TestConstantAgent method), 289
test_bool() (tensorforce.tests.test_dqfd_agent.TestDQFDAGent method), 294
test_bool() (tensorforce.tests.test_dqn_agent.TestDQNAgent method), 300
test_bool() (tensorforce.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 305
test_bool() (tensorforce.tests.test_naf_agent.TestNAFAGent method), 311
test_bool() (tensorforce.tests.test_ppo_agent.TestPPOAgent method), 316
test_bool() (tensorforce.tests.test_random_agent.TestRandomAgent method), 326
test_bool() (tensorforce.tests.test_trpo_agent.TestTRPOAgent method), 336
test_bool() (tensorforce.tests.test_vpg_agent.TestVPAGent method), 346
test_bounded_float() (tensorforce.tests.base_agent_test.BaseAgentTest method), 283
test_bounded_float() (tensorforce.tests.test_constant_agent.TestConstantAgent method), 289
test_bounded_float() (tensorforce.tests.test_dqfd_agent.TestDQFDAGent method), 294

<code>test_bounded_float()</code>	(tensorflow.tests.test_dqn_agent.TestDQNAgent method), 300	<code>test_int()</code>	(tensorflow.tests.base_agent_test.BaseAgentTest method), 283
<code>test_bounded_float()</code>	(tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 305	<code>test_int()</code>	(tensorflow.tests.test_constant_agent.TestConstantAgent method), 289
<code>test_bounded_float()</code>	(tensorflow.tests.test_naf_agent.TestNAFAgent method), 311	<code>test_int()</code>	(tensorflow.tests.test_dqfd_agent.TestDQFDAgent method), 295
<code>test_bounded_float()</code>	(tensorflow.tests.test_ppo_agent.TestPPOAgent method), 316	<code>test_int()</code>	(tensorflow.tests.test_dqn_agent.TestDQNAgent method), 300
<code>test_bounded_float()</code>	(tensorflow.tests.test_random_agent.TestRandomAgent method), 326	<code>test_int()</code>	(tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 305
<code>test_bounded_float()</code>	(tensorflow.tests.test_trpo_agent.TestTRPOAgent method), 336	<code>test_int()</code>	(tensorflow.tests.test_naf_agent.TestNAFAgent method), 311
<code>test_bounded_float()</code>	(tensorflow.tests.test_vpg_agent.TestVPGAgent method), 346	<code>test_int()</code>	(tensorflow.tests.test_ppo_agent.TestPPOAgent method), 316
<code>test_clipped_step()</code>	(tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers method), 356	<code>test_int()</code>	(tensorflow.tests.test_random_agent.TestRandomAgent method), 326
<code>test_evolutionary()</code>	(tensorflow.tests.test_vpg_optimizers.TestVPGOptimizers method), 356	<code>test_int()</code>	(tensorflow.tests.test_trpo_agent.TestTRPOAgent method), 336
<code>test_example()</code>	(tensorflow.tests.test_quickstart_example.TestQuickstartExample method), 321	<code>test_int()</code>	(tensorflow.tests.test_vpg_agent.TestVPGAgent method), 346
<code>test_float()</code>	(tensorflow.tests.base_agent_test.BaseAgentTest method), 283	<code>test_lstm()</code>	(tensorflow.tests.base_agent_test.BaseAgentTest method), 283
<code>test_float()</code>	(tensorflow.tests.test_constant_agent.TestConstantAgent method), 289	<code>test_lstm()</code>	(tensorflow.tests.test_constant_agent.TestConstantAgent method), 289
<code>test_float()</code>	(tensorflow.tests.test_dqfd_agent.TestDQFDAgent method), 295	<code>test_lstm()</code>	(tensorflow.tests.test_dqfd_agent.TestDQFDAgent method), 295
<code>test_float()</code>	(tensorflow.tests.test_dqn_agent.TestDQNAgent method), 300	<code>test_lstm()</code>	(tensorflow.tests.test_dqn_agent.TestDQNAgent method), 300
<code>test_float()</code>	(tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 305	<code>test_lstm()</code>	(tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 306
<code>test_float()</code>	(tensorflow.tests.test_naf_agent.TestNAFAgent method), 311	<code>test_lstm()</code>	(tensorflow.tests.test_naf_agent.TestNAFAgent method), 311
<code>test_float()</code>	(tensorflow.tests.test_ppo_agent.TestPPOAgent method), 316	<code>test_lstm()</code>	(tensorflow.tests.test_ppo_agent.TestPPOAgent method), 316
<code>test_float()</code>	(tensorflow.tests.test_random_agent.TestRandomAgent method), 326	<code>test_lstm()</code>	(tensorflow.tests.test_random_agent.TestRandomAgent method), 326
<code>test_float()</code>	(tensorflow.tests.test_trpo_agent.TestTRPOAgent method), 336	<code>test_lstm()</code>	(tensorflow.tests.test_trpo_agent.TestTRPOAgent method), 336
<code>test_float()</code>	(tensorflow.tests.test_vpg_agent.TestVPGAgent method), 346	<code>test_lstm()</code>	(tensorflow.tests.test_vpg_agent.TestVPGAgent method), 346
<code>test_gae()</code>	(tensorflow.tests.test_reward_estimation.TestRewardEstimation method), 331	<code>test_multi()</code>	(tensorflow.tests.base_agent_test.BaseAgentTest method), 283
<code>test_gae_baseline()</code>	(tensorflow.tests.test_vpg_baselines.TestVPGBaselines method), 351	<code>test_multi()</code>	(tensorflow.tests.test_constant_agent.TestConstantAgent method), 289
		<code>test_multi()</code>	(tensorflow.tests.test_dqfd_agent.TestDQFDAgent method), 295
		<code>test_multi()</code>	(tensorflow.tests.test_dqn_agent.TestDQNAgent method), 300
		<code>test_multi()</code>	(tensorflow.tests.test_dqn_nstep_agent.TestDQNNstepAgent method), 306
		<code>test_multi()</code>	(tensorflow.tests.test_naf_agent.TestNAFAgent method), 311
		<code>test_multi()</code>	(tensorflow.tests.test_ppo_agent.TestPPOAgent method), 316

test_multi() (tensorforce.tests.test_random_agent.TestRandomAgent method), 326	TestVPGAgent (class in tensorforce.tests.test_vpg_agent), 341
test_multi() (tensorforce.tests.test_trpo_agent.TestTRPOAgent method), 336	TestVPGBaselines (class in tensorforce.tests.test_vpg_baselines), 346
test_multi() (tensorforce.tests.test_vpg_agent.TestVPGAgent method), 346	TestVPGOptimizers (class in tensorforce.tests.test_vpg_optimizers), 351
test_multi_baseline() (tensorforce.tests.test_vpg_baselines.TestVPGBaselines method), 351	text_output() (tensorforce.meta_parameter_recorder.MetaParameterRecorder method), 357
test_multi_step() (tensorforce.tests.test_vpg_optimizers.TestVPGOptimizers method), 356	tf_action_exploration() (tensorforce.models.constant_model.ConstantModel method), 191
test_natural_gradient() (tensorforce.tests.test_vpg_optimizers.TestVPGOptimizers method), 356	tf_action_exploration() (tensorforce.models.distribution_model.DistributionModel method), 194
test_network_baseline() (tensorforce.tests.test_vpg_baselines.TestVPGBaselines method), 351	tf_action_exploration() (tensorforce.models.DistributionModel method), 246
test_optimized_step() (tensorforce.tests.test_vpg_optimizers.TestVPGOptimizers method), 356	tf_action_exploration() (tensorforce.models.DPGTargetModel method), 259
test_reinforceio_homepage() (tensorforce.tests.test_tutorial_code.TestTutorialCode method), 341	tf_action_exploration() (tensorforce.models.MemoryModel method), 241
test_states_baseline() (tensorforce.tests.test_vpg_baselines.TestVPGBaselines method), 351	tf_action_exploration() (tensorforce.models.Model method), 237
test_subsampling_step() (tensorforce.tests.test_vpg_optimizers.TestVPGOptimizers method), 356	tf_action_exploration() (tensorforce.models.model.Model method), 199
TestConstantAgent (class in tensorforce.tests.test_constant_agent), 284	tf_action_exploration() (tensorforce.models.pg_log_prob_model.PGLogProbModel method), 203
TestDQFDAgent (class in tensorforce.tests.test_dqfd_agent), 289	tf_action_exploration() (tensorforce.models.pg_model.PGModel method), 207
TestDQNAgent (class in tensorforce.tests.test_dqn_agent), 295	tf_action_exploration() (tensorforce.models.pg_prob_ratio_model.PGProbRatioModel method), 212
TestDQNNstepAgent (class in tensorforce.tests.test_dqn_nstep_agent), 300	tf_action_exploration() (tensorforce.models.PGLogProbModel method), 263
TestNAFAgent (class in tensorforce.tests.test_naf_agent), 306	tf_action_exploration() (tensorforce.models.PGModel method), 250
TestPPOAgent (class in tensorforce.tests.test_ppo_agent), 311	tf_action_exploration() (tensorforce.models.PGProbRatioModel method), 255
TestQuickstartExample (class in tensorforce.tests.test_quickstart_example), 317	tf_action_exploration() (tensorforce.models.q_demo_model.QDemoModel method), 217
TestRandomAgent (class in tensorforce.tests.test_random_agent), 321	tf_action_exploration() (tensorforce.models.q_model.QModel method), 221
TestRewardEstimation (class in tensorforce.tests.test_reward_estimation), 326	tf_action_exploration() (tensorforce.models.q_naf_model.QNAFModel method), 225
TestTRPOAgent (class in tensorforce.tests.test_trpo_agent), 331	tf_action_exploration() (tensorforce.models.q_nstep_model.QNstepModel method), 225
TestTutorialCode (class in tensorforce.tests.test_tutorial_code), 336	
TestTutorialCode.MyClient (class in tensorforce.tests.test_tutorial_code), 336	

method), 230

`tf_action_exploration()` (tensorflow.models.QDemoModel method), 280

`tf_action_exploration()` (tensorflow.models.QModel method), 267

`tf_action_exploration()` (tensorflow.models.QNAFModel method), 276

`tf_action_exploration()` (tensorflow.models.QNstepModel method), 271

`tf_action_exploration()` (tensorflow.models.random_model.RandomModel method), 234

`tf_actions_and_internals()` (tensorflow.models.constant_model.ConstantModel method), 191

`tf_actions_and_internals()` (tensorflow.models.distribution_model.DistributionModel method), 194

`tf_actions_and_internals()` (tensorflow.models.DistributionModel method), 246

`tf_actions_and_internals()` (tensorflow.models.DPGTargetModel method), 259

`tf_actions_and_internals()` (tensorflow.models.MemoryModel method), 241

`tf_actions_and_internals()` (tensorflow.models.Model method), 237

`tf_actions_and_internals()` (tensorflow.models.model.Model method), 199

`tf_actions_and_internals()` (tensorflow.models.pg_log_prob_model.PGLogProbModel method), 203

`tf_actions_and_internals()` (tensorflow.models.pg_model.PGModel method), 207

`tf_actions_and_internals()` (tensorflow.models.pg_prob_ratio_model.PGProbRatioModel method), 212

`tf_actions_and_internals()` (tensorflow.models.PGLogProbModel method), 263

`tf_actions_and_internals()` (tensorflow.models.PGModel method), 250

`tf_actions_and_internals()` (tensorflow.models.PGProbRatioModel method), 255

`tf_actions_and_internals()` (tensorflow.models.q_demo_model.QDemoModel method), 217

`tf_actions_and_internals()` (tensorflow.models.q_model.QModel method), 221

`tf_actions_and_internals()` (tensorflow.models.q_naf_model.QNAFModel method), 226

`tf_actions_and_internals()` (tensorflow.models.q_nstep_model.QNstepModel method), 230

`tf_actions_and_internals()` (tensorflow.models.QDemoModel method), 280

`tf_actions_and_internals()` (tensorflow.models.QModel method), 268

`tf_actions_and_internals()` (tensorflow.models.QNAFModel method), 276

`tf_actions_and_internals()` (tensorflow.models.QNstepModel method), 272

`tf_actions_and_internals()` (tensorflow.models.random_model.RandomModel method), 234

`tf_apply()` (tensorflow.core.networks.complex_network.ComplexLayeredNetwork method), 114

`tf_apply()` (tensorflow.core.networks.complex_network.Input method), 115

`tf_apply()` (tensorflow.core.networks.complex_network.Output method), 116

`tf_apply()` (tensorflow.core.networks.Conv1d method), 137

`tf_apply()` (tensorflow.core.networks.Conv2d method), 138

`tf_apply()` (tensorflow.core.networks.Dense method), 136

`tf_apply()` (tensorflow.core.networks.Dropout method), 132

`tf_apply()` (tensorflow.core.networks.Dueling method), 136

`tf_apply()` (tensorflow.core.networks.Embedding method), 134

`tf_apply()` (tensorflow.core.networks.Flatten method), 132

`tf_apply()` (tensorflow.core.networks.InternalLstm method), 139

`tf_apply()` (tensorflow.core.networks.Layer method), 130

`tf_apply()` (tensorflow.core.networks.layer.Conv1d method), 117

`tf_apply()` (tensorflow.core.networks.layer.Conv2d method), 117

`tf_apply()` (tensorflow.core.networks.layer.Dense method), 118

`tf_apply()` (tensorflow.core.networks.layer.Dropout method), 119

`tf_apply()` (tensorflow.core.networks.layer.Dueling method), 120

`tf_apply()` (tensorflow.core.networks.layer.Embedding method), 120

`tf_apply()` (tensorflow.core.networks.layer.Flatten method), 121

`tf_apply()` (tensorflow.core.networks.layer.InternalLstm

method), 122

tf_apply() (tensorflow.core.networks.layer.Layer method), 122

tf_apply() (tensorflow.core.networks.layer.Linear method), 123

tf_apply() (tensorflow.core.networks.layer.Lstm method), 124

tf_apply() (tensorflow.core.networks.layer.Nonlinearity method), 125

tf_apply() (tensorflow.core.networks.layer.Pool2d method), 126

tf_apply() (tensorflow.core.networks.layer.TFLayer method), 126

tf_apply() (tensorflow.core.networks.LayerBasedNetwork method), 141

tf_apply() (tensorflow.core.networks.LayeredNetwork method), 142

tf_apply() (tensorflow.core.networks.Linear method), 135

tf_apply() (tensorflow.core.networks.Lstm method), 139

tf_apply() (tensorflow.core.networks.Network method), 140

tf_apply() (tensorflow.core.networks.network.LayerBasedNetwork method), 127

tf_apply() (tensorflow.core.networks.network.LayeredNetwork method), 128

tf_apply() (tensorflow.core.networks.network.Network method), 129

tf_apply() (tensorflow.core.networks.Nonlinearity method), 131

tf_apply() (tensorflow.core.networks.Pool2d method), 133

tf_apply() (tensorflow.core.networks.TFLayer method), 130

tf_baseline_loss() (tensorflow.models.pg_log_prob_model.PGLogProbModel method), 203

tf_baseline_loss() (tensorflow.models.pg_model.PGModel method), 207

tf_baseline_loss() (tensorflow.models.pg_prob_ratio_model.PGProbRatioModel method), 213

tf_baseline_loss() (tensorflow.models.PGLogProbModel method), 263

tf_baseline_loss() (tensorflow.models.PGModel method), 251

tf_baseline_loss() (tensorflow.models.PGProbRatioModel method), 255

tf_combined_loss() (tensorflow.models.q_demo_model.QDemoModel method), 217

tf_combined_loss() (tensorflow.models.QDemoModel method), 280

tf_demo_loss() (tensorflow.models.q_demo_model.QDemoModel method), 217

tf_demo_loss() (tensorflow.models.QDemoModel method), 280

tf_demo_optimization() (tensorflow.models.q_demo_model.QDemoModel method), 217

tf_demo_optimization() (tensorflow.models.QDemoModel method), 280

tf_discounted_cumulative_reward() (tensorflow.models.distribution_model.DistributionModel method), 194

tf_discounted_cumulative_reward() (tensorflow.models.DistributionModel method), 246

tf_discounted_cumulative_reward() (tensorflow.models.DPGTargetModel method), 259

tf_discounted_cumulative_reward() (tensorflow.models.MemoryModel method), 241

tf_discounted_cumulative_reward() (tensorflow.models.pg_log_prob_model.PGLogProbModel method), 203

tf_discounted_cumulative_reward() (tensorflow.models.pg_model.PGModel method), 208

tf_discounted_cumulative_reward() (tensorflow.models.pg_prob_ratio_model.PGProbRatioModel method), 213

tf_discounted_cumulative_reward() (tensorflow.models.PGLogProbModel method), 263

tf_discounted_cumulative_reward() (tensorflow.models.PGModel method), 251

tf_discounted_cumulative_reward() (tensorflow.models.PGProbRatioModel method), 255

tf_discounted_cumulative_reward() (tensorflow.models.q_demo_model.QDemoModel method), 217

tf_discounted_cumulative_reward() (tensorflow.models.q_model.QModel method), 221

tf_discounted_cumulative_reward() (tensorflow.models.q_naf_model.QNAFModel method), 226

tf_discounted_cumulative_reward() (tensorflow.models.q_nstep_model.QNstepModel method), 230

tf_discounted_cumulative_reward() (tensorflow.models.QDemoModel method), 280

tf_discounted_cumulative_reward() (tensorflow

force.models.QModel method), 268	force.models.distribution_model.DistributionModel method), 194
tf_discounted_cumulative_reward() (tensorforce.models.QNAFModel method), 276	tf_import_experience() (tensorforce.models.DistributionModel method), 246
tf_discounted_cumulative_reward() (tensorforce.models.QNstepModel method), 272	tf_import_experience() (tensorforce.models.DPGTargetModel method), 259
tf_dtype() (in module tensorforce.util), 359	tf_import_experience() (tensorforce.models.MemoryModel method), 241
tf_entropy() (tensorforce.core.distributions.Bernoulli method), 98	tf_import_experience() (tensorforce.models.pg_log_prob_model.PGLogProbModel method), 203
tf_entropy() (tensorforce.core.distributions.bernoulli.Bernoulli method), 92	tf_import_experience() (tensorforce.models.pg_model.PGModel method), 208
tf_entropy() (tensorforce.core.distributions.Beta method), 99	tf_import_experience() (tensorforce.models.pg_prob_ratio_model.PGProbRatioModel method), 213
tf_entropy() (tensorforce.core.distributions.beta.Beta method), 93	tf_import_experience() (tensorforce.models.PGLogProbModel method), 264
tf_entropy() (tensorforce.core.distributions.Categorical method), 98	tf_import_experience() (tensorforce.models.PGProbRatioModel method), 256
tf_entropy() (tensorforce.core.distributions.categorical.Categorical method), 94	tf_import_experience() (tensorforce.models.q_demo_model.QDemoModel method), 217
tf_entropy() (tensorforce.core.distributions.Distribution method), 97	tf_import_experience() (tensorforce.models.q_model.QModel method), 222
tf_entropy() (tensorforce.core.distributions.distribution.Distribution method), 95	tf_import_experience() (tensorforce.models.q_naf_model.QNAFModel method), 226
tf_entropy() (tensorforce.core.distributions.Gaussian method), 99	tf_import_experience() (tensorforce.models.q_nstep_model.QNstepModel method), 230
tf_entropy() (tensorforce.core.distributions.gaussian.Gaussian method), 96	tf_import_experience() (tensorforce.models.QDemoModel method), 281
tf_explore() (tensorforce.core.explorations.Constant method), 103	tf_import_experience() (tensorforce.models.QModel method), 268
tf_explore() (tensorforce.core.explorations.constant.Constant method), 100	tf_import_experience() (tensorforce.models.QNAFModel method), 272
tf_explore() (tensorforce.core.explorations.epsilon_anneal.EpsilonAnneal method), 100	tf_initialize() (tensorforce.core.memories.Latest method), 111
tf_explore() (tensorforce.core.explorations.epsilon_decay.EpsilonDecay method), 101	tf_initialize() (tensorforce.core.memories.Memory method), 108
tf_explore() (tensorforce.core.explorations.EpsilonAnneal method), 103	tf_initialize() (tensorforce.core.memories.memory.Memory method), 105
tf_explore() (tensorforce.core.explorations.EpsilonDecay method), 103	tf_initialize() (tensorforce.core.memories.prioritized_replay.PrioritizedReplay method), 105
tf_explore() (tensorforce.core.explorations.Exploration method), 102	
tf_explore() (tensorforce.core.explorations.exploration.Exploration method), 101	
tf_explore() (tensorforce.core.explorations.GaussianNoise method), 104	
tf_explore() (tensorforce.core.explorations.ornstein_uhlenbeck_process.OrnsteinUhlenbeckProcess method), 102	
tf_explore() (tensorforce.core.explorations.OrnsteinUhlenbeckProcess method), 104	
tf_import_demo_experience() (tensorforce.models.q_demo_model.QDemoModel method), 217	
tf_import_demo_experience() (tensorforce.models.QDemoModel method), 281	
tf_import_experience() (tensorforce.models.QModel method), 268	

method), 106

`tf_initialize()` (tensorflow.core.memories.PrioritizedReplay method), 113

`tf_initialize()` (tensorflow.core.memories.Queue method), 110

`tf_initialize()` (tensorflow.core.memories.Replay method), 112

`tf_initialize()` (tensorflow.core.memories.replay.Replay method), 108

`tf_initialize()` (tensorflow.core.optimizers.solvers.conjugate_gradient.ConjugateGradient method), 143

`tf_initialize()` (tensorflow.core.optimizers.solvers.ConjugateGradient method), 149

`tf_initialize()` (tensorflow.core.optimizers.solvers.Iterative method), 148

`tf_initialize()` (tensorflow.core.optimizers.solvers.iterative.Iterative method), 144

`tf_initialize()` (tensorflow.core.optimizers.solvers.line_search.LineSearch method), 145

`tf_initialize()` (tensorflow.core.optimizers.solvers.LineSearch method), 150

`tf_initialize()` (tensorflow.models.constant_model.ConstantModel method), 191

`tf_initialize()` (tensorflow.models.distribution_model.DistributionModel method), 194

`tf_initialize()` (tensorflow.models.DistributionModel method), 246

`tf_initialize()` (tensorflow.models.DPGTargetModel method), 260

`tf_initialize()` (tensorflow.models.MemoryModel method), 242

`tf_initialize()` (tensorflow.models.Model method), 237

`tf_initialize()` (tensorflow.models.model.Model method), 200

`tf_initialize()` (tensorflow.models.pg_log_prob_model.PGLogProbModel method), 204

`tf_initialize()` (tensorflow.models.pg_model.PGModel method), 208

`tf_initialize()` (tensorflow.models.pg_prob_ratio_model.PGProbRatioModel method), 213

`tf_initialize()` (tensorflow.models.PGLogProbModel method), 264

`tf_initialize()` (tensorflow.models.PGModel method), 251

`tf_initialize()` (tensorflow.models.PGProbRatioModel method), 256

`tf_initialize()` (tensorflow.models.q_demo_model.QDemoModel method), 218

`tf_initialize()` (tensorflow.models.q_model.QModel method), 222

`tf_initialize()` (tensorflow.models.q_naf_model.QNAFModel method), 226

`tf_initialize()` (tensorflow.models.q_nstep_model.QNstepModel method), 230

`tf_initialize()` (tensorflow.models.QDemoModel method), 281

`tf_initialize()` (tensorflow.models.QModel method), 268

`tf_initialize()` (tensorflow.models.QNAFModel method), 276

`tf_initialize()` (tensorflow.models.QNstepModel method), 272

`tf_initialize()` (tensorflow.models.random_model.RandomModel method), 234

`tf_kl_divergence()` (tensorflow.core.distributions.Bernoulli method), 98

`tf_kl_divergence()` (tensorflow.core.distributions.bernoulli.Bernoulli method), 93

`tf_kl_divergence()` (tensorflow.core.distributions.Beta method), 99

`tf_kl_divergence()` (tensorflow.core.distributions.beta.Beta method), 93

`tf_kl_divergence()` (tensorflow.core.distributions.Categorical method), 98

`tf_kl_divergence()` (tensorflow.core.distributions.categorical.Categorical method), 94

`tf_kl_divergence()` (tensorflow.core.distributions.Distribution method), 97

`tf_kl_divergence()` (tensorflow.core.distributions.distribution.Distribution method), 95

`tf_kl_divergence()` (tensorflow.core.distributions.Gaussian method), 96

`tf_kl_divergence()` (tensorflow.models.distribution_model.DistributionModel method), 194

`tf_kl_divergence()` (tensorflow.models.DistributionModel method), 246

`tf_kl_divergence()` (tensorflow.models.DPGTargetModel method), 260

`tf_kl_divergence()` (tensorflow.models.pg_log_prob_model.PGLogProbModel method), 204

`tf_kl_divergence()` (tensorflow.models.pg_model.PGModel method), 208

`tf_kl_divergence()` (tensorflow.models.pg_prob_ratio_model.PGProbRatioModel method), 213

`tf_kl_divergence()` (tensorflow.models.PGLogProbModel method), 264

`tf_kl_divergence()` (tensorflow.models.PGModel method), 251

`tf_kl_divergence()` (tensorflow.models.PGProbRatioModel method), 256

`tf_kl_divergence()` (tensorflow.models.q_demo_model.QDemoModel method), 218

`tf_kl_divergence()` (tensorflow.models.q_model.QModel method), 222

`tf_kl_divergence()` (tensorflow.models.q_naf_model.QNAFModel method), 226

`tf_kl_divergence()` (tensorflow.models.q_nstep_model.QNstepModel method), 230

`tf_kl_divergence()` (tensorflow.models.QDemoModel method), 281

`tf_kl_divergence()` (tensorflow.models.QModel method), 268

`tf_kl_divergence()` (tensorflow.models.QNAFModel method), 276

`tf_kl_divergence()` (tensorflow.models.QNstepModel method), 272

`tf_kl_divergence()` (tensorflow.models.random_model.RandomModel method), 234

`tf_kl_divergence()` (tensorflow.core.distributions.Bernoulli method), 98

`tf_kl_divergence()` (tensorflow.core.distributions.bernoulli.Bernoulli method), 93

`tf_kl_divergence()` (tensorflow.core.distributions.Beta method), 99

`tf_kl_divergence()` (tensorflow.core.distributions.beta.Beta method), 93

`tf_kl_divergence()` (tensorflow.core.distributions.Categorical method), 98

`tf_kl_divergence()` (tensorflow.core.distributions.categorical.Categorical method), 94

`tf_kl_divergence()` (tensorflow.core.distributions.Distribution method), 97

`tf_kl_divergence()` (tensorflow.core.distributions.distribution.Distribution method), 95

`tf_kl_divergence()` (tensorflow.core.distributions.Gaussian method), 96

`tf_kl_divergence()` (tensorflow.models.distribution_model.DistributionModel method), 194

`tf_kl_divergence()` (tensorflow.models.DistributionModel method), 246

`tf_kl_divergence()` (tensorflow.models.DPGTargetModel method), 260

`tf_kl_divergence()` (tensorflow.models.pg_log_prob_model.PGLogProbModel method), 204

`tf_kl_divergence()` (tensorflow.models.pg_model.PGModel method), 208

`tf_kl_divergence()` (tensorflow.models.pg_prob_ratio_model.PGProbRatioModel method), 213

`tf_kl_divergence()` (tensorflow.models.PGLogProbModel method), 264

`tf_kl_divergence()` (tensorflow.models.PGModel method), 251

`tf_kl_divergence()` (tensorflow.models.PGProbRatioModel method), 256

`tf_kl_divergence()` (tensorflow.models.q_demo_model.QDemoModel method), 218

`tf_kl_divergence()` (tensorflow.models.q_model.QModel method), 222

`tf_kl_divergence()` (tensorflow.models.q_naf_model.QNAFModel method), 226

`tf_kl_divergence()` (tensorflow.models.q_nstep_model.QNstepModel method), 230

`tf_kl_divergence()` (tensorflow.models.QDemoModel method), 281

`tf_kl_divergence()` (tensorflow.models.QModel method), 268

`tf_kl_divergence()` (tensorflow.models.QNAFModel method), 276

`tf_kl_divergence()` (tensorflow.models.QNstepModel method), 272

`tf_kl_divergence()` (tensorflow.models.random_model.RandomModel method), 234

method), 213

`tf_kl_divergence()` (tensorflow.models.PGLogProbModel method), 264

`tf_kl_divergence()` (tensorflow.models.PGModel method), 251

`tf_kl_divergence()` (tensorflow.models.PGProbRatioModel method), 256

`tf_kl_divergence()` (tensorflow.models.q_demo_model.QDemoModel method), 218

`tf_kl_divergence()` (tensorflow.models.q_model.QModel method), 222

`tf_kl_divergence()` (tensorflow.models.q_naf_model.QNAFModel method), 226

`tf_kl_divergence()` (tensorflow.models.q_nstep_model.QNstepModel method), 230

`tf_kl_divergence()` (tensorflow.models.QDemoModel method), 281

`tf_kl_divergence()` (tensorflow.models.QModel method), 268

`tf_kl_divergence()` (tensorflow.models.QNAFModel method), 276

`tf_kl_divergence()` (tensorflow.models.QNstepModel method), 272

`tf_layers` (tensorflow.core.networks.layer.TFLayer attribute), 126

`tf_layers` (tensorflow.core.networks.TFLayer attribute), 130

`tf_log_probability()` (tensorflow.core.distributions.Bernoulli method), 98

`tf_log_probability()` (tensorflow.core.distributions.bernoulli.Bernoulli method), 93

`tf_log_probability()` (tensorflow.core.distributions.Beta method), 100

`tf_log_probability()` (tensorflow.core.distributions.beta.Beta method), 93

`tf_log_probability()` (tensorflow.core.distributions.Categorical method), 98

`tf_log_probability()` (tensorflow.core.distributions.categorical.Categorical method), 94

`tf_log_probability()` (tensorflow.core.distributions.Distribution method), 97

`tf_log_probability()` (tensorflow.core.distributions.distribution.Distribution method), 95

`tf_log_probability()` (tensorflow.core.distributions.Gaussian method), 99

`tf_log_probability()` (tensorflow.core.distributions.gaussian.Gaussian method), 96

`tf_loss()` (tensorflow.core.baselines.aggregated_baseline.AggregatedBaseline method), 83

`tf_loss()` (tensorflow.core.baselines.AggregatedBaseline method), 89

`tf_loss()` (tensorflow.core.baselines.Baseline method), 88

`tf_loss()` (tensorflow.core.baselines.baseline.Baseline method), 84

`tf_loss()` (tensorflow.core.baselines.cnn_baseline.CNNBaseline method), 85

`tf_loss()` (tensorflow.core.baselines.CNNBaseline method), 91

`tf_loss()` (tensorflow.core.baselines.mlp_baseline.MLPBaseline method), 86

`tf_loss()` (tensorflow.core.baselines.MLPBaseline method), 91

`tf_loss()` (tensorflow.core.baselines.network_baseline.NetworkBaseline method), 87

`tf_loss()` (tensorflow.core.baselines.NetworkBaseline method), 90

`tf_loss()` (tensorflow.models.distribution_model.DistributionModel method), 194

`tf_loss()` (tensorflow.models.DistributionModel method), 246

`tf_loss()` (tensorflow.models.DPGTargetModel method), 260

`tf_loss()` (tensorflow.models.MemoryModel method), 242

`tf_loss()` (tensorflow.models.pg_log_prob_model.PGLogProbModel method), 204

`tf_loss()` (tensorflow.models.pg_model.PGModel method), 208

`tf_loss()` (tensorflow.models.pg_prob_ratio_model.PGProbRatioModel method), 213

`tf_loss()` (tensorflow.models.PGLogProbModel method), 264

`tf_loss()` (tensorflow.models.PGModel method), 251

`tf_loss()` (tensorflow.models.PGProbRatioModel method), 256

`tf_loss()` (tensorflow.models.q_demo_model.QDemoModel method), 218

`tf_loss()` (tensorflow.models.q_model.QModel method), 222

`tf_loss()` (tensorflow.models.q_naf_model.QNAFModel method), 226

`tf_loss()` (tensorflow.models.q_nstep_model.QNstepModel method), 230

`tf_loss()` (tensorflow.models.QDemoModel method),

- 281
- `tf_loss()` (tensorforce.models.QModel method), 268
- `tf_loss()` (tensorforce.models.QNAFModel method), 276
- `tf_loss()` (tensorforce.models.QNstepModel method), 272
- `tf_loss_per_instance()` (tensorforce.models.distribution_model.DistributionModel method), 195
- `tf_loss_per_instance()` (tensorforce.models.DistributionModel method), 247
- `tf_loss_per_instance()` (tensorforce.models.DPGTargetModel method), 260
- `tf_loss_per_instance()` (tensorforce.models.MemoryModel method), 242
- `tf_loss_per_instance()` (tensorforce.models.pg_log_prob_model.PGLogProbModel method), 204
- `tf_loss_per_instance()` (tensorforce.models.pg_model.PGModel method), 209
- `tf_loss_per_instance()` (tensorforce.models.pg_prob_ratio_model.PGProbRatioModel method), 214
- `tf_loss_per_instance()` (tensorforce.models.PGLogProbModel method), 264
- `tf_loss_per_instance()` (tensorforce.models.PGModel method), 252
- `tf_loss_per_instance()` (tensorforce.models.PGProbRatioModel method), 256
- `tf_loss_per_instance()` (tensorforce.models.q_demo_model.QDemoModel method), 218
- `tf_loss_per_instance()` (tensorforce.models.q_model.QModel method), 222
- `tf_loss_per_instance()` (tensorforce.models.q_naf_model.QNAFModel method), 227
- `tf_loss_per_instance()` (tensorforce.models.q_nstep_model.QNstepModel method), 231
- `tf_loss_per_instance()` (tensorforce.models.QDemoModel method), 281
- `tf_loss_per_instance()` (tensorforce.models.QModel method), 268
- `tf_loss_per_instance()` (tensorforce.models.QNAFModel method), 277
- `tf_loss_per_instance()` (tensorforce.models.QNstepModel method), 273
- `tf_next_step()` (tensorforce.core.optimizers.solvers.conjugate_gradient.ConjugateGradient method), 143
- `tf_next_step()` (tensorforce.core.optimizers.solvers.ConjugateGradient method), 149
- `tf_next_step()` (tensorforce.core.optimizers.solvers.Iterative method), 148
- `tf_next_step()` (tensorforce.core.optimizers.solvers.iterative.Iterative method), 144
- `tf_next_step()` (tensorforce.core.optimizers.solvers.line_search.LineSearch method), 146
- `tf_next_step()` (tensorforce.core.optimizers.solvers.LineSearch method), 150
- `tf_observe_timestep()` (tensorforce.models.constant_model.ConstantModel method), 191
- `tf_observe_timestep()` (tensorforce.models.distribution_model.DistributionModel method), 195
- `tf_observe_timestep()` (tensorforce.models.DistributionModel method), 247
- `tf_observe_timestep()` (tensorforce.models.DPGTargetModel method), 260
- `tf_observe_timestep()` (tensorforce.models.MemoryModel method), 242
- `tf_observe_timestep()` (tensorforce.models.Model method), 237
- `tf_observe_timestep()` (tensorforce.models.model.Model method), 200
- `tf_observe_timestep()` (tensorforce.models.pg_log_prob_model.PGLogProbModel method), 204
- `tf_observe_timestep()` (tensorforce.models.pg_model.PGModel method), 209
- `tf_observe_timestep()` (tensorforce.models.pg_prob_ratio_model.PGProbRatioModel method), 214
- `tf_observe_timestep()` (tensorforce.models.PGLogProbModel method), 264
- `tf_observe_timestep()` (tensorforce.models.PGModel method), 252
- `tf_observe_timestep()` (tensorforce.models.PGProbRatioModel method), 256
- `tf_observe_timestep()` (tensorforce.models.q_demo_model.QDemoModel method), 218
- `tf_observe_timestep()` (tensorforce.models.q_model.QModel method), 222
- `tf_observe_timestep()` (tensorforce.models.q_naf_model.QNAFModel method), 227

<code>tf_observe_timestep()</code> (tensorflow.models.q_nstep_model.QNstepModel method), 231	<code>tf_optimization()</code> (tensorflow.models.QNAFModel method), 277
<code>tf_observe_timestep()</code> (tensorflow.models.QDemoModel method), 281	<code>tf_optimization()</code> (tensorflow.models.QNstepModel method), 273
<code>tf_observe_timestep()</code> (tensorflow.models.QModel method), 269	<code>tf_optimizers</code> (tensorflow.core.optimizers.tf_optimizer.TFOptimizer attribute), 165
<code>tf_observe_timestep()</code> (tensorflow.models.QNAFModel method), 277	<code>tf_optimizers</code> (tensorflow.core.optimizers.TFOptimizer attribute), 171
<code>tf_observe_timestep()</code> (tensorflow.models.QNstepModel method), 273	<code>tf_parameterize()</code> (tensorflow.core.distributions.Bernoulli method), 98
<code>tf_observe_timestep()</code> (tensorflow.models.random_model.RandomModel method), 234	<code>tf_parameterize()</code> (tensorflow.core.distributions.bernoulli.Bernoulli method), 93
<code>tf_optimization()</code> (tensorflow.models.distribution_model.DistributionModel method), 195	<code>tf_parameterize()</code> (tensorflow.core.distributions.Beta method), 100
<code>tf_optimization()</code> (tensorflow.models.DistributionModel method), 247	<code>tf_parameterize()</code> (tensorflow.core.distributions.beta.Beta method), 93
<code>tf_optimization()</code> (tensorflow.models.DPGTargetModel method), 260	<code>tf_parameterize()</code> (tensorflow.core.distributions.Categorical method), 98
<code>tf_optimization()</code> (tensorflow.models.MemoryModel method), 242	<code>tf_parameterize()</code> (tensorflow.core.distributions.categorical.Categorical method), 94
<code>tf_optimization()</code> (tensorflow.models.pg_log_prob_model.PGLogProbModel method), 204	<code>tf_parameterize()</code> (tensorflow.core.distributions.Distribution method), 97
<code>tf_optimization()</code> (tensorflow.models.pg_model.PGModel method), 209	<code>tf_parameterize()</code> (tensorflow.core.distributions.distribution.Distribution method), 95
<code>tf_optimization()</code> (tensorflow.models.pg_prob_ratio_model.PGProbRatioModel method), 214	<code>tf_parameterize()</code> (tensorflow.core.distributions.Gaussian method), 99
<code>tf_optimization()</code> (tensorflow.models.PGLogProbModel method), 264	<code>tf_parameterize()</code> (tensorflow.core.distributions.gaussian.Gaussian method), 96
<code>tf_optimization()</code> (tensorflow.models.PGModel method), 252	<code>tf_predict()</code> (tensorflow.core.baselines.aggregated_baseline.AggregatedBaseline method), 84
<code>tf_optimization()</code> (tensorflow.models.PGProbRatioModel method), 256	<code>tf_predict()</code> (tensorflow.core.baselines.AggregatedBaseline method), 89
<code>tf_optimization()</code> (tensorflow.models.q_demo_model.QDemoModel method), 218	<code>tf_predict()</code> (tensorflow.core.baselines.Baseline method), 88
<code>tf_optimization()</code> (tensorflow.models.q_model.QModel method), 222	<code>tf_predict()</code> (tensorflow.core.baselines.baseline.Baseline method), 85
<code>tf_optimization()</code> (tensorflow.models.q_naf_model.QNAFModel method), 227	<code>tf_predict()</code> (tensorflow.core.baselines.cnn_baseline.CNNBaseline method), 86
<code>tf_optimization()</code> (tensorflow.models.q_nstep_model.QNstepModel method), 231	<code>tf_predict()</code> (tensorflow.core.baselines.CNNBaseline method), 92
<code>tf_optimization()</code> (tensorflow.models.QDemoModel method), 281	<code>tf_predict()</code> (tensorflow.core.baselines.mlp_baseline.MLPBaseline method), 86
<code>tf_optimization()</code> (tensorflow.models.QModel method), 269	<code>tf_predict()</code> (tensorflow.core.baselines.MLPBaseline method), 91
	<code>tf_predict()</code> (tensorflow.core.baselines.network_baseline.NetworkBaseline method), 87

- method), 87
- `tf_predict()` (tensorflow.core.baselines.NetworkBaseline method), 90
- `tf_predict_target_q()` (tensorflow.models.DPGTargetModel method), 260
- `tf_preprocess()` (tensorflow.models.constant_model.ConstantModel method), 191
- `tf_preprocess()` (tensorflow.models.distribution_model.DistributionModel method), 195
- `tf_preprocess()` (tensorflow.models.DistributionModel method), 247
- `tf_preprocess()` (tensorflow.models.DPGTargetModel method), 260
- `tf_preprocess()` (tensorflow.models.MemoryModel method), 243
- `tf_preprocess()` (tensorflow.models.Model method), 238
- `tf_preprocess()` (tensorflow.models.model.Model method), 200
- `tf_preprocess()` (tensorflow.models.pg_log_prob_model.PGLogProbModel method), 204
- `tf_preprocess()` (tensorflow.models.pg_model.PGModel method), 209
- `tf_preprocess()` (tensorflow.models.pg_prob_ratio_model.PGProbRatioModel method), 214
- `tf_preprocess()` (tensorflow.models.PGLogProbModel method), 265
- `tf_preprocess()` (tensorflow.models.PGModel method), 252
- `tf_preprocess()` (tensorflow.models.PGProbRatioModel method), 256
- `tf_preprocess()` (tensorflow.models.q_demo_model.QDemoModel method), 218
- `tf_preprocess()` (tensorflow.models.q_model.QModel method), 222
- `tf_preprocess()` (tensorflow.models.q_naf_model.QNAFModel method), 227
- `tf_preprocess()` (tensorflow.models.q_nstep_model.QNstepModel method), 231
- `tf_preprocess()` (tensorflow.models.QDemoModel method), 281
- `tf_preprocess()` (tensorflow.models.QModel method), 269
- `tf_preprocess()` (tensorflow.models.QNAFModel method), 277
- `tf_preprocess()` (tensorflow.models.QNstepModel method), 273
- `tf_preprocess()` (tensorflow.models.random_model.RandomModel method), 234
- `tf_q_delta()` (tensorflow.models.q_demo_model.QDemoModel method), 218
- `tf_q_delta()` (tensorflow.models.q_model.QModel method), 222
- `tf_q_delta()` (tensorflow.models.q_naf_model.QNAFModel method), 227
- `tf_q_delta()` (tensorflow.models.q_nstep_model.QNstepModel method), 231
- `tf_q_delta()` (tensorflow.models.QDemoModel method), 281
- `tf_q_delta()` (tensorflow.models.QModel method), 269
- `tf_q_delta()` (tensorflow.models.QNAFModel method), 277
- `tf_q_delta()` (tensorflow.models.QNstepModel method), 273
- `tf_q_value()` (tensorflow.models.q_demo_model.QDemoModel method), 218
- `tf_q_value()` (tensorflow.models.q_model.QModel method), 223
- `tf_q_value()` (tensorflow.models.q_naf_model.QNAFModel method), 227
- `tf_q_value()` (tensorflow.models.q_nstep_model.QNstepModel method), 231
- `tf_q_value()` (tensorflow.models.QDemoModel method), 282
- `tf_q_value()` (tensorflow.models.QModel method), 269
- `tf_q_value()` (tensorflow.models.QNAFModel method), 277
- `tf_q_value()` (tensorflow.models.QNstepModel method), 273
- `tf_reference()` (tensorflow.core.baselines.aggregated_baseline.AggregatedBaseline method), 84
- `tf_reference()` (tensorflow.core.baselines.AggregatedBaseline method), 89
- `tf_reference()` (tensorflow.core.baselines.Baseline method), 88
- `tf_reference()` (tensorflow.core.baselines.baseline.Baseline method), 85
- `tf_reference()` (tensorflow.core.baselines.cnn_baseline.CNNBaseline method), 86
- `tf_reference()` (tensorflow.core.baselines.CNNBaseline method), 92
- `tf_reference()` (tensorflow.core.baselines.mlp_baseline.MLPBaseline method), 87
- `tf_reference()` (tensorflow.core.baselines.MLPBaseline method), 91
- `tf_reference()` (tensorflow.core.baselines.network_baseline.NetworkBaseline method), 87
- `tf_reference()` (tensorflow.core.baselines.NetworkBaseline method), 90
- `tf_reference()` (tensorflow.models.distribution_model.DistributionModel

method), 195

tf_reference() (tensorflow.models.DistributionModel method), 247

tf_reference() (tensorflow.models.DPGTargetModel method), 260

tf_reference() (tensorflow.models.MemoryModel method), 243

tf_reference() (tensorflow.models.pg_log_prob_model.PGLogProbModel method), 204

tf_reference() (tensorflow.models.pg_model.PGModel method), 209

tf_reference() (tensorflow.models.pg_prob_ratio_model.PGProbRatioModel method), 214

tf_reference() (tensorflow.models.PGLogProbModel method), 265

tf_reference() (tensorflow.models.PGModel method), 252

tf_reference() (tensorflow.models.PGProbRatioModel method), 256

tf_reference() (tensorflow.models.q_demo_model.QDemoModel method), 218

tf_reference() (tensorflow.models.q_model.QModel method), 223

tf_reference() (tensorflow.models.q_naf_model.QNAFModel method), 227

tf_reference() (tensorflow.models.q_nstep_model.QNstepModel method), 231

tf_reference() (tensorflow.models.QDemoModel method), 282

tf_reference() (tensorflow.models.QModel method), 269

tf_reference() (tensorflow.models.QNAFModel method), 277

tf_reference() (tensorflow.models.QNstepModel method), 273

tf_regularization_loss() (tensorflow.core.baselines.aggregated_baseline.AggregatedBaseline method), 84

tf_regularization_loss() (tensorflow.core.baselines.AggregatedBaseline method), 89

tf_regularization_loss() (tensorflow.core.baselines.Baseline method), 89

tf_regularization_loss() (tensorflow.core.baselines.baseline.Baseline method), 85

tf_regularization_loss() (tensorflow.core.baselines.cnn_baseline.CNNBaseline method), 86

tf_regularization_loss() (tensorflow.core.baselines.CNNBaseline method), 92

tf_regularization_loss() (tensorflow.core.baselines.mlp_baseline.MLPBaseline method), 87

tf_regularization_loss() (tensorflow.core.baselines.MLPBaseline method), 91

tf_regularization_loss() (tensorflow.core.baselines.network_baseline.NetworkBaseline method), 88

tf_regularization_loss() (tensorflow.core.baselines.NetworkBaseline method), 90

tf_regularization_loss() (tensorflow.core.distributions.Bernoulli method), 93

tf_regularization_loss() (tensorflow.core.distributions.bernoulli.Bernoulli method), 93

tf_regularization_loss() (tensorflow.core.distributions.Beta method), 100

tf_regularization_loss() (tensorflow.core.distributions.beta.Beta method), 93

tf_regularization_loss() (tensorflow.core.distributions.Categorical method), 98

tf_regularization_loss() (tensorflow.core.distributions.categorical.Categorical method), 94

tf_regularization_loss() (tensorflow.core.distributions.Distribution method), 97

tf_regularization_loss() (tensorflow.core.distributions.distribution.Distribution method), 95

tf_regularization_loss() (tensorflow.core.distributions.Gaussian method), 99

tf_regularization_loss() (tensorflow.core.distributions.gaussian.Gaussian method), 96

tf_regularization_loss() (tensorflow.core.networks.complex_network.ComplexLayeredNetwork method), 114

tf_regularization_loss() (tensorflow.core.networks.complex_network.Input method), 115

tf_regularization_loss() (tensorflow.core.networks.complex_network.Output method), 116

tf_regularization_loss() (tensorflow.core.networks.Conv1d method), 137

tf_regularization_loss() (tensorflow.core.networks.Conv2d method), 138

tf_regularization_loss() (tensorflow.core.networks.Dense method), 136

tf_regularization_loss() (tensorflow

force.core.networks.Dropout method), 132	126
tf_regularization_loss() (tensorflow.core.networks.Dueling method), 136	tf_regularization_loss() (tensorflow.core.networks.LayerBasedNetwork method), 141
tf_regularization_loss() (tensorflow.core.networks.Embedding method), 134	tf_regularization_loss() (tensorflow.core.networks.LayeredNetwork method), 142
tf_regularization_loss() (tensorflow.core.networks.Flatten method), 132	tf_regularization_loss() (tensorflow.core.networks.Linear method), 135
tf_regularization_loss() (tensorflow.core.networks.InternalLstm method), 139	tf_regularization_loss() (tensorflow.core.networks.Lstm method), 139
tf_regularization_loss() (tensorflow.core.networks.Layer method), 130	tf_regularization_loss() (tensorflow.core.networks.Network method), 140
tf_regularization_loss() (tensorflow.core.networks.layer.Conv1d method), 117	tf_regularization_loss() (tensorflow.core.networks.network.LayerBasedNetwork method), 127
tf_regularization_loss() (tensorflow.core.networks.layer.Conv2d method), 117	tf_regularization_loss() (tensorflow.core.networks.network.LayeredNetwork method), 128
tf_regularization_loss() (tensorflow.core.networks.layer.Dense method), 118	tf_regularization_loss() (tensorflow.core.networks.network.Network method), 129
tf_regularization_loss() (tensorflow.core.networks.layer.Dropout method), 119	tf_regularization_loss() (tensorflow.core.networks.Nonlinearity method), 131
tf_regularization_loss() (tensorflow.core.networks.layer.Dueling method), 120	tf_regularization_loss() (tensorflow.core.networks.Pool2d method), 133
tf_regularization_loss() (tensorflow.core.networks.layer.Embedding method), 120	tf_regularization_loss() (tensorflow.core.networks.TFLayer method), 130
tf_regularization_loss() (tensorflow.core.networks.layer.Flatten method), 121	tf_regularization_losses() (tensorflow.models.distribution_model.DistributionModel method), 196
tf_regularization_loss() (tensorflow.core.networks.layer.InternalLstm method), 122	tf_regularization_losses() (tensorflow.models.DistributionModel method), 248
tf_regularization_loss() (tensorflow.core.networks.layer.Layer method), 122	tf_regularization_losses() (tensorflow.models.DPGTargetModel method), 261
tf_regularization_loss() (tensorflow.core.networks.layer.Linear method), 123	tf_regularization_losses() (tensorflow.models.MemoryModel method), 243
tf_regularization_loss() (tensorflow.core.networks.layer.Lstm method), 124	tf_regularization_losses() (tensorflow.models.pg_log_prob_model.PGLogProbModel method), 205
tf_regularization_loss() (tensorflow.core.networks.layer.Nonlinearity method), 125	tf_regularization_losses() (tensorflow.models.pg_model.PGModel method), 209
tf_regularization_loss() (tensorflow.core.networks.layer.Pool2d method), 126	tf_regularization_losses() (tensorflow.models.pg_prob_ratio_model.PGProbRatioModel method), 214
tf_regularization_loss() (tensorflow.core.networks.layer.TFLayer method), 126	tf_regularization_losses() (tensorflow.models.PGLogProbModel method), 265
	tf_regularization_losses() (tensorflow.models.PGModel method), 252

<code>tf_regularization_losses()</code>	(tensorflow.models.PGProbRatioModel method), 257	<code>tf_retrieve_indices()</code>	(tensorflow.core.memories.replay.Replay method), 108
<code>tf_regularization_losses()</code>	(tensorflow.models.q_demo_model.QDemoModel method), 219	<code>tf_retrieve_sequences()</code>	(tensorflow.core.memories.Latest method), 111
<code>tf_regularization_losses()</code>	(tensorflow.models.q_model.QModel method), 223	<code>tf_retrieve_sequences()</code>	(tensorflow.core.memories.Memory method), 109
<code>tf_regularization_losses()</code>	(tensorflow.models.q_naf_model.QNAFModel method), 227	<code>tf_retrieve_sequences()</code>	(tensorflow.core.memories.memory.Memory method), 105
<code>tf_regularization_losses()</code>	(tensorflow.models.q_nstep_model.QNstepModel method), 231	<code>tf_retrieve_sequences()</code>	(tensorflow.core.memories.prioritized_replay.PrioritizedReplay method), 107
<code>tf_regularization_losses()</code>	(tensorflow.models.QDemoModel method), 282	<code>tf_retrieve_sequences()</code>	(tensorflow.core.memories.PrioritizedReplay method), 113
<code>tf_regularization_losses()</code>	(tensorflow.models.QModel method), 269	<code>tf_retrieve_sequences()</code>	(tensorflow.core.memories.Queue method), 110
<code>tf_regularization_losses()</code>	(tensorflow.models.QNAFModel method), 277	<code>tf_retrieve_sequences()</code>	(tensorflow.core.memories.Replay method), 112
<code>tf_regularization_losses()</code>	(tensorflow.models.QNstepModel method), 273	<code>tf_retrieve_sequences()</code>	(tensorflow.core.memories.replay.Replay method), 108
<code>tf_retrieve_episodes()</code>	(tensorflow.core.memories.Latest method), 111	<code>tf_retrieve_timesteps()</code>	(tensorflow.core.memories.Latest method), 111
<code>tf_retrieve_episodes()</code>	(tensorflow.core.memories.Memory method), 109	<code>tf_retrieve_timesteps()</code>	(tensorflow.core.memories.Memory method), 109
<code>tf_retrieve_episodes()</code>	(tensorflow.core.memories.memory.Memory method), 105	<code>tf_retrieve_timesteps()</code>	(tensorflow.core.memories.memory.Memory method), 105
<code>tf_retrieve_episodes()</code>	(tensorflow.core.memories.prioritized_replay.PrioritizedReplay method), 106	<code>tf_retrieve_timesteps()</code>	(tensorflow.core.memories.prioritized_replay.PrioritizedReplay method), 107
<code>tf_retrieve_episodes()</code>	(tensorflow.core.memories.PrioritizedReplay method), 113	<code>tf_retrieve_timesteps()</code>	(tensorflow.core.memories.PrioritizedReplay method), 113
<code>tf_retrieve_episodes()</code>	(tensorflow.core.memories.Queue method), 110	<code>tf_retrieve_timesteps()</code>	(tensorflow.core.memories.Queue method), 110
<code>tf_retrieve_episodes()</code>	(tensorflow.core.memories.Replay method), 112	<code>tf_retrieve_timesteps()</code>	(tensorflow.core.memories.Replay method), 112
<code>tf_retrieve_episodes()</code>	(tensorflow.core.memories.replay.Replay method), 108	<code>tf_retrieve_timesteps()</code>	(tensorflow.core.memories.replay.Replay method), 108
<code>tf_retrieve_indices()</code>	(tensorflow.core.memories.Latest method), 111	<code>tf_reward_estimation()</code>	(tensorflow.models.pg_log_prob_model.PGLogProbModel method), 205
<code>tf_retrieve_indices()</code>	(tensorflow.core.memories.prioritized_replay.PrioritizedReplay method), 106	<code>tf_reward_estimation()</code>	(tensorflow.models.pg_model.PGModel method), 209
<code>tf_retrieve_indices()</code>	(tensorflow.core.memories.PrioritizedReplay method), 113	<code>tf_reward_estimation()</code>	(tensorflow.models.pg_prob_ratio_model.PGProbRatioModel method), 214
<code>tf_retrieve_indices()</code>	(tensorflow.core.memories.Queue method), 110	<code>tf_reward_estimation()</code>	(tensorflow.core.memories.Replay method), 108
<code>tf_retrieve_indices()</code>	(tensorflow.core.memories.Replay method), 108		

force.models.PGLogProbModel method), 265
 tf_reward_estimation() (tensorflow.models.PGModel method), 252
 tf_reward_estimation() (tensorflow.models.PGProbRatioModel method), 257
 tf_sample() (tensorflow.core.distributions.Bernoulli method), 98
 tf_sample() (tensorflow.core.distributions.bernoulli.Bernoulli method), 93
 tf_sample() (tensorflow.core.distributions.Beta method), 100
 tf_sample() (tensorflow.core.distributions.beta.Beta method), 93
 tf_sample() (tensorflow.core.distributions.Categorical method), 98
 tf_sample() (tensorflow.core.distributions.categorical.Categorical method), 94
 tf_sample() (tensorflow.core.distributions.Distribution method), 97
 tf_sample() (tensorflow.core.distributions.distribution.Distribution method), 95
 tf_sample() (tensorflow.core.distributions.Gaussian method), 99
 tf_sample() (tensorflow.core.distributions.gaussian.Gaussian method), 96
 tf_solve() (tensorflow.core.optimizers.solvers.conjugate_gradient.ConjugateGradient method), 143
 tf_solve() (tensorflow.core.optimizers.solvers.ConjugateGradient method), 149
 tf_solve() (tensorflow.core.optimizers.solvers.Iterative method), 148
 tf_solve() (tensorflow.core.optimizers.solvers.iterative.Iterative method), 145
 tf_solve() (tensorflow.core.optimizers.solvers.line_search.LineSearch method), 146
 tf_solve() (tensorflow.core.optimizers.solvers.LineSearch method), 151
 tf_solve() (tensorflow.core.optimizers.solvers.Solver method), 147
 tf_solve() (tensorflow.core.optimizers.solvers.solver.Solver method), 147
 tf_step() (tensorflow.core.optimizers.clipped_step.ClippedStep method), 152
 tf_step() (tensorflow.core.optimizers.ClippedStep method), 175
 tf_step() (tensorflow.core.optimizers.Evolutionary method), 172
 tf_step() (tensorflow.core.optimizers.evolutionary.Evolutionary method), 154
 tf_step() (tensorflow.core.optimizers.global_optimizer.GlobalOptimizer method), 155
 tf_step() (tensorflow.core.optimizers.GlobalOptimizer method), 169
 tf_step() (tensorflow.core.optimizers.meta_optimizer.MetaOptimizer method), 157
 tf_step() (tensorflow.core.optimizers.MetaOptimizer method), 168
 tf_step() (tensorflow.core.optimizers.multi_step.MultiStep method), 158
 tf_step() (tensorflow.core.optimizers.MultiStep method), 176
 tf_step() (tensorflow.core.optimizers.natural_gradient.NaturalGradient method), 159
 tf_step() (tensorflow.core.optimizers.NaturalGradient method), 173
 tf_step() (tensorflow.core.optimizers.optimized_step.OptimizedStep method), 161
 tf_step() (tensorflow.core.optimizers.OptimizedStep method), 177
 tf_step() (tensorflow.core.optimizers.Optimizer method), 167
 tf_step() (tensorflow.core.optimizers.optimizer.Optimizer method), 163
 tf_step() (tensorflow.core.optimizers.solvers.conjugate_gradient.ConjugateGradient method), 144
 tf_step() (tensorflow.core.optimizers.solvers.ConjugateGradient method), 150
 tf_step() (tensorflow.core.optimizers.solvers.Iterative method), 148
 tf_step() (tensorflow.core.optimizers.solvers.iterative.Iterative method), 145
 tf_step() (tensorflow.core.optimizers.solvers.line_search.LineSearch method), 146
 tf_step() (tensorflow.core.optimizers.solvers.LineSearch method), 151
 tf_step() (tensorflow.core.optimizers.SubsamplingStep method), 179
 tf_step() (tensorflow.core.optimizers.Synchronization method), 180
 tf_step() (tensorflow.core.optimizers.synchronization.Synchronization method), 164
 tf_step() (tensorflow.core.optimizers.tf_optimizer.TFOptimizer method), 165
 tf_step() (tensorflow.core.optimizers.TFOptimizer method), 171
 tf_store() (tensorflow.core.memories.Latest method), 111
 tf_store() (tensorflow.core.memories.Memory method), 109
 tf_store() (tensorflow.core.memories.memory.Memory method), 105
 tf_store() (tensorflow.core.memories.prioritized_replay.PrioritizedReplay method), 107
 tf_store() (tensorflow.core.memories.PrioritizedReplay method), 113
 tf_store() (tensorflow.core.memories.Queue method),

- 110
- `tf_store()` (`tensorforce.core.memories.Replay` method), 112
- `tf_store()` (`tensorforce.core.memories.replay.Replay` method), 108
- `tf_target_actions_and_internals()` (`tensorforce.models.DPGTargetModel` method), 261
- `tf_tensors()` (`tensorforce.core.networks.complex_network.Input` method), 115
- `tf_tensors()` (`tensorforce.core.networks.complex_network.Output` method), 116
- `tf_tensors()` (`tensorforce.core.networks.Conv1d` method), 137
- `tf_tensors()` (`tensorforce.core.networks.Conv2d` method), 138
- `tf_tensors()` (`tensorforce.core.networks.Dense` method), 136
- `tf_tensors()` (`tensorforce.core.networks.Dropout` method), 132
- `tf_tensors()` (`tensorforce.core.networks.Dueling` method), 136
- `tf_tensors()` (`tensorforce.core.networks.Embedding` method), 134
- `tf_tensors()` (`tensorforce.core.networks.Flatten` method), 133
- `tf_tensors()` (`tensorforce.core.networks.InternalLstm` method), 139
- `tf_tensors()` (`tensorforce.core.networks.Layer` method), 130
- `tf_tensors()` (`tensorforce.core.networks.layer.Conv1d` method), 117
- `tf_tensors()` (`tensorforce.core.networks.layer.Conv2d` method), 117
- `tf_tensors()` (`tensorforce.core.networks.layer.Dense` method), 118
- `tf_tensors()` (`tensorforce.core.networks.layer.Dropout` method), 119
- `tf_tensors()` (`tensorforce.core.networks.layer.Dueling` method), 120
- `tf_tensors()` (`tensorforce.core.networks.layer.Embedding` method), 120
- `tf_tensors()` (`tensorforce.core.networks.layer.Flatten` method), 121
- `tf_tensors()` (`tensorforce.core.networks.layer.InternalLstm` method), 122
- `tf_tensors()` (`tensorforce.core.networks.layer.Layer` method), 122
- `tf_tensors()` (`tensorforce.core.networks.layer.Linear` method), 123
- `tf_tensors()` (`tensorforce.core.networks.layer.Lstm` method), 124
- `tf_tensors()` (`tensorforce.core.networks.layer.Nonlinearity` method), 125
- `tf_tensors()` (`tensorforce.core.networks.layer.Pool2d` method), 126
- `tf_tensors()` (`tensorforce.core.networks.layer.TFLayer` method), 126
- `tf_tensors()` (`tensorforce.core.networks.Linear` method), 135
- `tf_tensors()` (`tensorforce.core.networks.Lstm` method), 139
- `tf_tensors()` (`tensorforce.core.networks.Nonlinearity` method), 131
- `tf_tensors()` (`tensorforce.core.networks.Pool2d` method), 133
- `tf_tensors()` (`tensorforce.core.networks.TFLayer` method), 130
- `tf_update_batch()` (`tensorforce.core.memories.Latest` method), 111
- `tf_update_batch()` (`tensorforce.core.memories.Memory` method), 109
- `tf_update_batch()` (`tensorforce.core.memories.memory.Memory` method), 105
- `tf_update_batch()` (`tensorforce.core.memories.prioritized_replay.PrioritizedReplay` method), 107
- `tf_update_batch()` (`tensorforce.core.memories.PrioritizedReplay` method), 113
- `tf_update_batch()` (`tensorforce.core.memories.Queue` method), 110
- `tf_update_batch()` (`tensorforce.core.memories.Replay` method), 112
- `tf_update_batch()` (`tensorforce.core.memories.replay.Replay` method), 108
- `TFLayer` (class in `tensorforce.core.networks`), 130
- `TFLayer` (class in `tensorforce.core.networks.layer`), 126
- `TFOptimizer` (class in `tensorforce.core.optimizers`), 169
- `TFOptimizer` (class in `tensorforce.core.optimizers.tf_optimizer`), 164
- `ThreadedRunner` (class in `tensorforce.execution`), 187
- `ThreadedRunner` (class in `tensorforce.execution.threaded_runner`), 184
- `timestep` (`tensorforce.execution.BaseRunner` attribute), 187
- `timestep` (`tensorforce.execution.Runner` attribute), 187
- `timestep` (`tensorforce.execution.runner.Runner` attribute), 184
- `timestep` (`tensorforce.execution.threaded_runner.ThreadedRunner` attribute), 185
- `timestep` (`tensorforce.execution.ThreadedRunner` attribute), 188
- `translate_abstract_actions_to_keys()` (`tensorforce.contrib.unreal_engine.UEnv4Environment` method), 83

TRPOAgent (class in `tensorforce.agents`), [68](#)

TRPOAgent (class in `tensorforce.agents.trpo_agent`), [46](#)

U

UE4Environment (class in `tensorforce.contrib.unreal_engine`), [81](#)

V

VPGAgent (class in `tensorforce.agents`), [70](#)

VPGAgent (class in `tensorforce.agents.vpg_agent`), [48](#)

W

WorkerAgentGenerator() (in module `tensorforce.execution`), [188](#)

WorkerAgentGenerator() (in module `tensorforce.execution.threaded_runner`), [185](#)