Tensorforce Documentation

Release 0.6.1

Tensorforce Team

Sep 19, 2020

Basics

Tensorforce is an open-source deep reinforcement learning framework, with an emphasis on modularized flexible library design and straightforward usability for applications in research and practice. Tensorforce is built on top of Google's TensorFlow framework and requires Python 3.

Tensorforce follows a set of high-level design choices which differentiate it from other similar libraries:

- **Modular component-based design**: Feature implementations, above all, strive to be as generally applicable and configurable as possible, potentially at some cost of faithfully resembling details of the introducing paper.
- **Separation of RL algorithm and application**: Algorithms are agnostic to the type and structure of inputs (states/observations) and outputs (actions/decisions), as well as the interaction with the application environment.
- **Full-on TensorFlow models**: The entire reinforcement learning logic, including control flow, is implemented in TensorFlow, to enable portable computation graphs independent of application programming language, and to facilitate the deployment of models.

CHAPTER 1

Installation

A stable version of Tensorforce is periodically updated on PyPI and installed as follows:

```
pip3 install tensorforce
```

To always use the latest version of Tensorforce, install the GitHub version instead:

```
git clone https://github.com/tensorforce/tensorforce.git
cd tensorforce
pip3 install -e .
```

Environments require additional packages for which there are setup options available (`ale`, `gym`, `retro`, `vizdoom`, `carla`; or `envs` for all environments), however, some require additional tools to be installed separately (see environments documentation). Other setup options include `tfa` for TensorFlow Addons and `tune` for HpBandSter required for the `tune.py` script.

Dockerfile

If you want to use Tensorforce within a Docker container, the following is a minimal `Dockerfile` to get started:

```
FROM python:3.8
RUN \
  pip3 install tensorforce
```

Or alternatively for the latest version:

```
FROM python:3.8
RUN \
  git clone https://github.com/tensorforce/tensorforce.git && \
  pip3 install -e tensorforce
```

Subsequently, the container can be built via:

```
docker build .
```

CHAPTER 2

Getting started

Quickstart example

2.1 Initializing an environment

It is recommended to initialize an environment via the `Environment.create(...)` interface.

```
from tensorforce.environments import Environment
```

For instance, the OpenAI CartPole environment can be initialized as follows (see environment docs for available environments and arguments):

```
environment = Environment.create(
    environment='gym', level='CartPole', max_episode_timesteps=500
)
```

Gym's pre-defined versions are also accessible:

```
environment = Environment.create(environment='gym', level='CartPole-v1')
```

Alternatively, an environment can be specified as a config file:

```
{
    "environment": "gym",
    "level": "CartPole"
}
```

Environment config files can be loaded by passing their file path:

```
environment = Environment.create(
    environment='environment.json', max_episode_timesteps=500
)
```

Custom Gym environments can be used in the same way, but require the corresponding class(es) to be imported and registered accordingly.

Finally, it is possible to implement a custom environment using Tensorforce's `Environment` interface:

```
class CustomEnvironment(Environment):

    def __init__(self):
        super().__init__()

    def states(self):
        return dict(type='float', shape=(8,))

    def actions(self):
        return dict(type='int', num_values=4)

    # Optional: should only be defined if environment has a natural fixed
    # maximum episode length; restrict training timesteps via
    #     Environment.create(..., max_episode_timesteps=???)
    def max_episode_timesteps(self):
        return super().max_episode_timesteps()

    # Optional additional steps to close environment
    def close(self):
        super().close()

    def reset(self):
        state = np.random.random(size=(8,))
        return state

    def execute(self, actions):
        next_state = np.random.random(size=(8,))
        terminal = np.random.random() < 0.5
        reward = np.random.random()
        return next_state, terminal, reward
```

Custom environment implementations can be loaded by passing either the environment object itself:

```
environment = Environment.create(
    environment=CustomEnvironment, max_episode_timesteps=100
)
```

or its module path:

```
environment = Environment.create(
    environment='custom_env.CustomEnvironment', max_episode_timesteps=100
)
```

It is generally recommended to specify the `max_episode_timesteps` argument of `Environment.create(...)` (at least for training), as some agent parameters may rely on this value.

2.2 Initializing an agent

Similarly to environments, it is recommended to initialize an agent via the `Agent.create(...)` interface.

```
from tensorforce.agents import Agent
```

For instance, the generic Tensorforce agent can be initialized as follows (see agent docs for available agents and arguments):

```
agent = Agent.create(
    agent='tensorforce', environment=environment, update=64,
    optimizer=dict(optimizer='adam', learning_rate=1e-3),
    objective='policy_gradient', reward_estimation=dict(horizon=20)
)
```

Other pre-defined agent classes can alternatively be used, for instance, Proximal Policy Optimization:

```
agent = Agent.create(
    agent='ppo', environment=environment, batch_size=10, learning_rate=1e-3
)
```

Alternatively, an agent can be specified as a config file:

```
{
    "agent": "tensorforce",
    "update": 64,
    "optimizer": {
        "optimizer": "adam",
        "learning_rate": 1e-3
    },
    "objective": "policy_gradient",
    "reward_estimation": {
        "horizon": 20
    }
}
```

Agent config files can be loaded by passing their file path:

```
agent = Agent.create(agent='agent.json', environment=environment)
```

While it is possible to specify the agent arguments `states`, `actions` and `max_episode_timesteps`, it is generally recommended to specify the `environment` argument instead (which will automatically infer the other values accordingly), by passing the environment object as returned by `Environment.create(...)`.

2.3 Training and evaluation

It is recommended to use the execution utilities for training and evaluation, like the Runner utility, which offer a range of configuration options:

```
from tensorforce.execution import Runner
```

A basic experiment consisting of training and subsequent evaluation can be written in a few lines of code:

```
runner = Runner(
    agent='agent.json',
    environment=dict(environment='gym', level='CartPole'),
    max_episode_timesteps=500
)
```

(continues on next page)

(continued from previous page)

```
runner.run(num_episodes=200)

runner.run(num_episodes=100, evaluation=True)

runner.close()
```

The same interface also makes it possible to run experiments involving multiple parallelized environments:

```
runner = Runner(
    agent='agent.json',
    environment=dict(environment='gym', level='CartPole'),
    max_episode_timesteps=500,
    num_parallel=5, remote='multiprocessing'
)

runner.run(num_episodes=100)

runner.close()
```

Note that in this case both agent and environment are created as part of `Runner`, not via `Agent.create(...)` and `Environment.create(...)`. If agent and environment are specified separately, the user is required to take care of passing the agent arguments `environment` and `parallel_interactions` (in the parallelized case) as well as closing both agent and environment separately at the end.

The execution utility classes take care of handling the agent-environment interaction correctly, and thus should be used where possible. Alternatively, if more detailed control over the agent-environment interaction is required, a simple training loop can be defined as follows, using the act-observe interaction pattern (see also the act-observe example):

```
# Create agent and environment
environment = Environment.create(
    environment='environment.json', max_episode_timesteps=500
)
agent = Agent.create(agent='agent.json', environment=environment)

# Train for 100 episodes
for _ in range(100):
    states = environment.reset()
    terminal = False
    while not terminal:
        actions = agent.act(states=states)
        states, terminal, reward = environment.execute(actions=actions)
        agent.observe(terminal=terminal, reward=reward)
```

Alternatively, the act-experience-update interface offers even more flexibility (see also the act-experience-update example), however, note that a few stateful network layers will not be updated correctly in independent-mode (currently, `exponential_normalization`):

```
# Train for 100 episodes
for _ in range(100):
    episode_states = list()
    episode_internals = list()
    episode_actions = list()
    episode_terminal = list()
    episode_reward = list()
```

(continues on next page)

(continued from previous page)

```
    states = environment.reset()
    internals = agent.initial_internals()
    terminal = False
    while not terminal:
        episode_states.append(states)
        episode_internals.append(internals)
        actions, internals = agent.act(
            states=states, internals=internals, independent=True
        )
        episode_actions.append(actions)
        states, terminal, reward = environment.execute(actions=actions)
        episode_terminal.append(terminal)
        episode_reward.append(reward)

    agent.experience(
        states=episode_states, internals=episode_internals,
        actions=episode_actions, terminal=episode_terminal,
        reward=episode_reward
    )
    agent.update()
```

Finally, the evaluation loop can be defined as follows:

```
# Evaluate for 100 episodes
sum_rewards = 0.0
for _ in range(100):
    states = environment.reset()
    internals = agent.initial_internals()
    terminal = False
    while not terminal:
        actions, internals = agent.act(
            states=states, internals=internals,
            independent=True, deterministic=True
        )
        states, terminal, reward = environment.execute(actions=actions)
        sum_rewards += reward

print('Mean episode reward:', sum_rewards / 100)

# Close agent and environment
agent.close()
environment.close()
```

CHAPTER 3

Agent specification

Agents are instantiated via `Agent.create(agent=...)`, with either of the specification alternatives presented below (`agent` acts as `type` argument). It is recommended to pass as second argument `environment` the application `Environment` implementation, which automatically extracts the corresponding `states`, `actions` and `max_episode_timesteps` arguments of the agent.

3.1 States and actions specification

A state/action value is specified as dictionary with mandatory attributes `type` (one of `'bool'`: binary, `'int'`: discrete, or `'float'`: continuous) and `shape` (a positive number or tuple thereof). Moreover, `'int'` values should additionally specify `num_values` (the fixed number of discrete options), whereas `'float'` values can specify bounds via `min/max_value`. If the state or action consists of multiple components, these are specified via an additional dictionary layer. The following example illustrates both possibilities:

```
states = dict(
    observation=dict(type='float', shape=(16, 16, 3)),
    attributes=dict(type='int', shape=(4, 2), num_values=5)
)
actions = dict(type='float', shape=10)
```

Note: Ideally, the agent arguments `states` and `actions` are specified implicitly by passing the `environment` argument.

3.2 How to specify modules

3.2.1 Dictionary with module type and arguments

```
Agent.create(...
    policy=dict(network=dict(type='layered', layers=[dict(type='dense', size=32)])),
```

(continues on next page)

(continued from previous page)

```
    memory=dict(type='replay', capacity=10000), ...
)
```

3.2.2 JSON specification file (plus additional arguments)

```
Agent.create(...
    policy=dict(network='network.json'),
    memory=dict(type='memory.json', capacity=10000), ...
)
```

3.2.3 Module path (plus additional arguments)

```
Agent.create(...
    policy=dict(network='my_module.TestNetwork'),
    memory=dict(type='tensorforce.core.memories.Replay', capacity=10000), ...
)
```

3.2.4 Callable or Type (plus additional arguments)

```
Agent.create(...
    policy=dict(network=TestNetwork),
    memory=dict(type=Replay, capacity=10000), ...
)
```

3.2.5 Default module: only arguments or first argument

```
Agent.create(...
    policy=dict(network=[dict(type='dense', size=32)]),
    memory=dict(capacity=10000), ...
)
```

CHAPTER 4

Features

4.1 Multi-input and non-sequential network architectures

See networks documentation.

4.2 Abort-terminal due to timestep limit

Besides `terminal=False` or `=0` for non-terminal and `terminal=True` or `=1` for true terminal, Tensorforce recognizes `terminal=2` as abort-terminal and handles it accordingly for reward estimation. Environments created via `Environment.create(..., max_episode_timesteps=?, ...)` will automatically return the appropriate terminal depending on whether an episode truly terminates or is aborted because it reached the time limit.

4.3 Action masking

```
agent = Agent.create(
    states=dict(type='float', shape=(10,)),
    actions=dict(type='int', shape=(), num_values=3),
    ...
)
...
states = dict(
    state=np.random.random_sample(size=(10,)),  # state (default name: "state")
    action_mask=[True, False, True]  # mask as'[ACTION-NAME]_mask' (default name:
↪"action")
)
action = agent.act(states=states)
assert action != 1
```

4.4 Parallel environment execution

See also the parallelization example for details on how to use this feature.

Execute multiple environments running locally in one call / batched:

```
Runner(
    agent='benchmarks/configs/ppo1.json', environment='CartPole-v1',
    num_parallel=4
)
runner.run(num_episodes=100, batch_agent_calls=True)
```

Execute environments running in different processes whenever ready / unbatched:

```
Runner(
    agent='benchmarks/configs/ppo1.json', environment='CartPole-v1',
    num_parallel=4, remote='multiprocessing'
)
runner.run(num_episodes=100)
```

Execute environments running on different machines, here using `run.py` instead of `Runner`:

```
# Environment machine 1
python run.py --environment gym --level CartPole-v1 --remote socket-server \
    --port 65432

# Environment machine 2
python run.py --environment gym --level CartPole-v1 --remote socket-server \
    --port 65433

# Agent machine
python run.py --agent benchmarks/configs/ppo1.json --episodes 100 \
    --num-parallel 2 --remote socket-client --host 127.0.0.1,127.0.0.1 \
    --port 65432,65433 --batch-agent-calls
```

4.5 Save & restore

4.5.1 TensorFlow saver (full model)

```
agent = Agent.create(...
    saver=dict(
        directory='data/checkpoints',
        frequency=100  # save checkpoint every 100 updates
    ), ...
)
...
agent.close()

# Restore latest agent checkpoint
agent = Agent.load(directory='data/checkpoints')
```

See also the save-load example.

4.5.2 NumPy / HDF5 (only weights)

```
agent = Agent.create(...)
...
agent.save(directory='data/checkpoints', format='numpy', append='episodes')

# Restore latest agent checkpoint
agent = Agent.load(directory='data/checkpoints', format='numpy')
```

See also the save-load example.

4.5.3 SavedModel export

See the SavedModel example for details on how to use this feature.

4.6 TensorBoard

```
Agent.create(...
    summarizer=dict(
        directory='data/summaries',
        # list of labels, or 'all'
        labels=['entropy', 'kl-divergence', 'loss', 'reward', 'update-norm']
    ), ...
)
```

4.7 Act-experience-update interaction

Instead of the default act-observe interaction pattern or the Runner utility, one can alternatively use the act-experience-update interface, which allows for more control over the experience the agent stores. See the act-experience-update example for details on how to use this feature. Note that a few stateful network layers will not be updated correctly in independent-mode (currently, `exponential_normalization`).

4.8 Record & pretrain

See the record-and-pretrain example for details on how to use this feature.

CHAPTER 5

run.py – Runner

5.1 Agent arguments

–[a]gent (*string*, **required** *unless "socket-server" remote mode*) – Agent (name, configuration JSON file, or library module) **–[c]heckpoints** (*string, default: not specified*) – TensorFlow checkpoints directory **–[s]ummaries** (*string, default: not specified*) – TensorBoard summaries directory **–recordings** (*string, default: not specified*) – Traces recordings directory

5.2 Environment arguments

–[e]nvironment (*string*, **required** *unless "socket-client" remote mode*) – Environment (name, configuration JSON file, or library module)

–[l]evel (*string, default: not specified*) – Level or game id, like `CartPole-v1`, if supported

–[m]ax-episode-timesteps (*int, default: not specified*) – Maximum number of timesteps per episode

–visualize (*bool, default: false*) – Visualize agent–environment interaction, if supported

–visualize-directory (*bool, default: not specified*) – Directory to store videos of agent–environment interaction, if supported

–import-modules (*string, default: not specified*) – Import comma-separated modules required for environment

5.3 Parallel execution arguments

–num-parallel (*int, default: no parallel execution*) – Number of environment instances to execute in parallel

–batch-agent-calls (*bool, default: false*) – Batch agent calls for parallel environment execution

–sync-timesteps (*bool, default: false*) – Synchronize parallel environment execution on timestep-level

–sync-episodes (*bool, default: false*) – Synchronize parallel environment execution on episode-level

–remote (*str, default: local execution*) – Communication mode for remote environment execution of parallelized environment execution: *"multiprocessing"* | *"socket-client"* | *"socket-server"*. In case of *"socket-server"*, runs environment in server communication loop until closed.

–blocking (*bool, default: false*) – Remote environments should be blocking

–host (*str, only for "socket-client" remote mode*) – Socket server hostname(s) or IP address(es), single value or comma-separated list

–port (*str, only for "socket-client/server" remote mode*) – Socket server port(s), single value or comma-separated list, increasing sequence if single host and port given

5.4 Runner arguments

–e[v]aluation (*bool, default: false*) – Run environment (last if multiple) in evaluation mode

–episodes [n] (*int, default: not specified*) – Number of episodes

–[t]imesteps (*int, default: not specified*) – Number of timesteps

–[u]pdates (*int, default: not specified*) – Number of agent updates

–mean-horizon (*int, default: 1*) – Number of episodes progress bar values and evaluation score are averaged over

–save-best-agent (*bool, default: false*) – Directory to save the best version of the agent according to the evaluation score

5.5 Logging arguments

–[r]epeat (*int, default: 1*) – Number of repetitions

–path (*string, default: not specified*) – Logging path, directory plus filename without extension

–seaborn (*bool, default: false*) – Use seaborn

CHAPTER 6

tune.py – Hyperparameter tuner

Uses the BOHB optimizer (Bayesian Optimization and Hyperband) internally.

6.1 Environment arguments

–[e]nvironment (*string*, **required**) – Environment (name, configuration JSON file, or library module)

–[l]evel (*string, default: not specified*) – Level or game id, like `CartPole-v1`, if supported

–[m]ax-episode-timesteps (*int, default: not specified*) – Maximum number of timesteps per episode

–import-modules (*string, default: not specified*) – Import comma-separated modules required for environment

6.2 Runner arguments

–episodes [n] (*int*, **required**) – Number of episodes

–num-[p]arallel (*int, default: no parallel execution*) – Number of environment instances to execute in parallel

6.3 Tuner arguments

–[r]uns-per-round (*string, default: 1,2,5,10*) – Comma-separated number of runs per optimization round, each with a successively smaller number of candidates

–[s]election-factor (*int, default: 3*) – Selection factor n, meaning that one out of n candidates in each round advances to the next optimization round

–num-[i]terations (*int, default: 1*) – Number of optimization iterations, each consisting of a series of optimization rounds with an increasingly reduced candidate pool

–[d]irectory (*string, default: "tuner"*) – Output directory

–restore (*string, default: not specified*) – Restore from given directory

–id (*string, default: "worker"*) – Unique worker id

CHAPTER 7

General agent interface

7.1 Initialization and termination

static TensorforceAgent.**create**(*agent='tensorforce'*, *environment=None*, ***kwargs*)
Creates an agent from a specification.

Parameters

- **agent** (*specification | Agent class/object | lambda[states -> actions]*) – JSON file, specification key, configuration dictionary, library module, or Agent class/object. Alternatively, an act-function mapping states to actions which is supposed to be recorded. (: Tensorforce base agent).
- **environment** (*Environment object*) – Environment which the agent is supposed to be trained on, environment-related arguments like state/action space specifications and maximum episode length will be extract if given ().
- **kwargs** – Additional agent arguments.

TensorforceAgent.**reset**()
Resets possibly inconsistent internal values, for instance, after saving and restoring an agent. Automatically triggered as part of Agent.create/load/initialize/restore.

TensorforceAgent.**close**()
Closes the agent.

7.2 Reinforcement learning interface

TensorforceAgent.**act**(*states*, *internals=None*, *parallel=0*, *independent=False*, *deterministic=False*, *evaluation=None*)
Returns action(s) for the given state(s), needs to be followed by observe() unless independent mode.

See the act-observe script for an example application as part of the act-observe interface.

Parameters

- **states** (*dict[state] | iter[dict[state]]*) – Dictionary containing state(s) to be acted on ().
- **internals** (*dict[internal] | iter[dict[internal]]*) – Dictionary containing current internal agent state(s), either given by `initial_internals()` at the beginning of an episode or as return value of the preceding `act()` call (if independent mode and agent has internal states).
- **parallel** (*int | iter[int]*) – Parallel execution index (: 0).
- **independent** (*bool*) – Whether act is not part of the main agent-environment interaction, and this call is thus not followed by observe() (: false).
- **deterministic** (*bool*) – Whether action should be chosen deterministically, so no sampling and no exploration, only valid in independent mode (: false).

Returns dict[action] | iter[dict[action]], dict[internal] | iter[dict[internal]] if `internals` argument given: Dictionary containing action(s), dictionary containing next internal agent state(s) if independent mode.

`TensorforceAgent.`**observe**(*reward=0.0*, *terminal=False*, *parallel=0*)

Observes reward and whether a terminal state is reached, needs to be preceded by `act()`.

See the act-observe script for an example application as part of the act-observe interface.

Parameters

- **reward** (*float | iter[float]*) – Reward (: 0.0).
- **terminal** (*bool | 0 | 1 | 2 | iter[..]*) – Whether a terminal state is reached, or 2 if the episode was aborted (: false).
- **parallel** (*int, iter[int]*) – Parallel execution index (: 0).

Returns Number of performed updates.

Return type int

7.3 Get initial internals (for independent-act)

`TensorforceAgent.`**initial_internals**()

Returns the initial internal agent state(s), to be used at the beginning of an episode as `internals` argument for `act()` in independent mode

Returns Dictionary containing initial internal agent state(s).

Return type dict[internal]

7.4 Experience - update interface

`TensorforceAgent.`**experience**(*states*, *actions*, *terminal*, *reward*, *internals=None*)

Feed experience traces.

See the act-experience-update script for an example application as part of the act-experience-update interface, which is an alternative to the act-observe interaction pattern.

Parameters

- **states** (*dict[array[state]]*) – Dictionary containing arrays of states ().

- **actions** (*dict[array[action]]*) – Dictionary containing arrays of actions ().
- **terminal** (*array[bool]*) – Array of terminals ().
- **reward** (*array[float]*) – Array of rewards ().
- **internals** (*dict[state]*) – Dictionary containing arrays of internal agent states (if agent has internal states).

`TensorforceAgent.`**`update`**(*query=None*, ***kwargs*)

Perform an update.

See the act-experience-update script for an example application as part of the act-experience-update interface, which is an alternative to the act-observe interaction pattern.

7.5 Pretraining

`TensorforceAgent.`**`pretrain`**(*directory*, *num_iterations*, *num_traces=1*, *num_updates=1*, *extension='.npz'*)

Simple pretraining approach as a combination of `experience()` and `update`, akin to behavioral cloning, using experience traces obtained e.g. via recording agent interactions (see documentation).

For the given number of iterations, load the given number of trace files (which each contain recorder[frequency] episodes), feed the experience to the agent's internal memory, and subsequently trigger the given number of updates (which will use the experience in the internal memory, fed in this or potentially previous iterations).

See the record-and-pretrain script for an example application.

Parameters

- **directory** (*path*) – Directory with experience traces, e.g. obtained via recorder; episode length has to be consistent with agent configuration ().
- **num_iterations** (*int > 0*) – Number of iterations consisting of loading new traces and performing multiple updates ().
- **num_traces** (*int > 0*) – Number of traces to load per iteration; has to at least satisfy the update batch size (: 1).
- **num_updates** (*int > 0*) – Number of updates per iteration (: 1).
- **extension** (*str*) – Traces file extension to filter the given directory for (: ".npz").

7.6 Loading and saving

static `TensorforceAgent.`**`load`**(*directory=None*, *filename=None*, *format=None*, *environment=None*, ***kwargs*)

Restores an agent from a directory/file.

Parameters

- **directory** (*str*) – Checkpoint directory (, unless saver is specified).
- **filename** (*str*) – Checkpoint filename, with or without append and extension (: "agent").
- **format** (*"checkpoint" | "saved-model" | "numpy" | "hdf5"*) – File format, "saved-model" loads an act-only agent based on a Protobuf model (: format matching directory and filename, required to be unambiguous).

- **environment** (*Environment object*) – Environment which the agent is supposed to be trained on, environment-related arguments like state/action space specifications and maximum episode length will be extract if given ().
- **kwargs** – Additional agent arguments.

TensorforceAgent.**save**(*directory*, *filename=None*, *format='checkpoint'*, *append=None*)

Saves the agent to a checkpoint.

Parameters

- **directory** (*str*) – Checkpoint directory ().
- **filename** (*str*) – Checkpoint filename, without extension (, unless "saved-model" format).
- **format** (*"checkpoint" | "saved-model" | "numpy" | "hdf5"*) – File format, "checkpoint" uses TensorFlow Checkpoint to save model, "saved-model" uses TensorFlow SavedModel to save an optimized act-only model, whereas the others store only variables as NumPy/HDF5 file (: TensorFlow Checkpoint).
- **append** (*"timesteps" | "episodes" | "updates"*) – Append timestep/episode/update to checkpoint filename (: none).

Returns Checkpoint path.

Return type str

CHAPTER 8

Constant Agent

class tensorforce.agents.**ConstantAgent**(*states*, *actions*, *max_episode_timesteps=None*, *action_values=None*, *config=None*, *recorder=None*)

Agent returning constant action values (specification key: `constant`).

Parameters

- **states** (*specification*) – States specification (, better implicitly specified via `environment` argument for `Agent.create(...)`), arbitrarily nested dictionary of state descriptions (usually taken from `Environment.states()`) with the following attributes:
- **actions** (*specification*) – Actions specification (, better implicitly specified via `environment` argument for `Agent.create(...)`), arbitrarily nested dictionary of action descriptions (usually taken from `Environment.actions()`) with the following attributes:
- **max_episode_timesteps** (*int > 0*) – Upper bound for numer of timesteps per episode (: not given, better implicitly specified via `environment` argument for `Agent.create(...)`).
- **action_values** (*dict[value]*) – Constant value per action (: false for binary boolean actions, 0 for discrete integer actions, 0.0 for continuous actions).
- **config** (*specification*) – Additional configuration options: calls (final value in case of schedule) (default: false). always_apply_variable_noise (bool) – Whether to always apply variable noise, also for independent *act()* calls (final value in case of schedule) (default: false). enable_int_action_masking (bool) – Whether int action options can be masked via an optional "[ACTION-NAME]_mask" state input (default: true). create_tf_assertions (bool) – Whether to create internal TensorFlow assertion operations (default: true). ‘

- **recorder** (*path | specification*) – Traces recordings directory, or recorder configuration with the following attributes (see record-and-pretrain script for example application) (: no recorder):

CHAPTER 9

Random Agent

class tensorforce.agents.**RandomAgent**(*states*, *actions*, *max_episode_timesteps=None*, *config=None*, *recorder=None*)

Agent returning random action values (specification key: `random`).

Parameters

- **states** (*specification*) – States specification (, better implicitly specified via `environment` argument for `Agent.create(...)`), arbitrarily nested dictionary of state descriptions (usually taken from `Environment.states()`) with the following attributes:
- **actions** (*specification*) – Actions specification (, better implicitly specified via `environment` argument for `Agent.create(...)`), arbitrarily nested dictionary of action descriptions (usually taken from `Environment.actions()`) with the following attributes:
- **max_episode_timesteps** (*int > 0*) – Upper bound for numer of timesteps per episode (: not given, better implicitly specified via `environment` argument for `Agent.create(...)`).
- **config** (*specification*) – Additional configuration options: calls (final value in case of schedule) (default: false). always_apply_variable_noise (bool) – Whether to always apply variable noise, also for independent *act()* calls (final value in case of schedule) (default: false). enable_int_action_masking (bool) – Whether int action options can be masked via an optional "[ACTION-NAME]_mask" state input (default: true). create_tf_assertions (bool) – Whether to create internal TensorFlow assertion operations (default: true). ‘
- **recorder** (*path | specification*) – Traces recordings directory, or recorder configuration with the following attributes (see record-and-pretrain script for example application) (: no recorder):

CHAPTER 10

Tensorforce Agent

class tensorforce.agents.**TensorforceAgent**(*states*, *actions*, *update*, *optimizer*, *objective*, *reward_estimation*, *max_episode_timesteps=None*, *policy='auto'*, *memory=None*, *baseline=None*, *baseline_optimizer=None*, *baseline_objective=None*, *l2_regularization=0.0*, *entropy_regularization=0.0*, *state_preprocessing='linear_normalization'*, *reward_preprocessing=None*, *exploration=0.0*, *variable_noise=0.0*, *parallel_interactions=1*, *config=None*, *saver=None*, *summarizer=None*, *recorder=None*, *baseline_policy=None*, *name=None*, *buffer_observe=None*, *device=None*, *seed=None*)

Tensorforce agent (specification key: `tensorforce`).

Highly configurable agent and basis for a broad class of deep reinforcement learning agents, which act according to a policy parametrized by a neural network, leverage a memory module for periodic updates based on batches of experience, and optionally employ a baseline/critic/target policy for improved reward estimation.

Parameters

- **states** (*specification*) – States specification (, better implicitly specified via `environment` argument for `Agent.create()`), arbitrarily nested dictionary of state descriptions (usually taken from `Environment.states()`) with the following attributes:
- **actions** (*specification*) – Actions specification (, better implicitly specified via `environment` argument for `Agent.create()`), arbitrarily nested dictionary of action descriptions (usually taken from `Environment.actions()`) with the following attributes:
- **max_episode_timesteps** (*int > 0*) – Upper bound for numer of timesteps per episode (: not given, better implicitly specified via `environment` argument for `Agent.create()`).

- **policy** (*specification*) – Policy configuration, see networks and policies documentation (: action distributions or value functions parametrized by an automatically configured network).
- **memory** (*int | specification*) – Replay memory capacity, or memory configuration, see the memories documentation (: minimum capacity recent memory).
- **update** (*int | specification*) – Model update configuration with the following attributes (: timesteps batch size):
- **optimizer** (*specification*) – Optimizer configuration, see the optimizers documentation (: Adam optimizer).
- **objective** (*specification*) – Optimization objective configuration, see the objectives documentation ().
- **reward_estimation** (*specification*) – Reward estimation configuration with the following attributes ():
- **baseline** (*specification*) – Baseline configuration, policy will be used as baseline if none, see networks and potentially policies documentation (: none).
- **baseline_optimizer** (specification | , float > 0.0) – Baseline optimizer configuration, see the optimizers documentation, main optimizer will be used for baseline if none, a float implies none and specifies a custom weight for the baseline loss (: none).
- **baseline_objective** (*specification*) – Baseline optimization objective configuration, see the objectives documentation, required if baseline optimizer is specified, main objective will be used for baseline if baseline objective and optimizer are not specified (: none).
- **l2_regularization** (, float >= 0.0) – L2 regularization loss weight (: no L2 regularization).
- **entropy_regularization** (, float >= 0.0) – Entropy regularization loss weight, to discourage the policy distribution from being "too certain" (: no entropy regularization).
- **state_preprocessing** (*dict[specification]*) – State preprocessing as layer or list of layers, see the preprocessing documentation, specified per state-type or -name (: linear normalization of bounded float states to [-2.0, 2.0]).
- **reward_preprocessing** (*specification*) – Reward preprocessing as layer or list of layers, see the preprocessing documentation (: no reward preprocessing).
- **exploration** (| dict[], float >= 0.0) – Exploration, defined as the probability for uniformly random output in case of `bool` and `int` actions, and the standard deviation of Gaussian noise added to every output in case of `float` actions, specified globally or per action-type or -name (: no exploration).
- **variable_noise** (, float >= 0.0) – Add Gaussian noise with given standard deviation to all trainable variables, as alternative exploration mechanism (: no variable noise).
- **parallel_interactions** (*int > 0*) – Maximum number of parallel interactions to support, for instance, to enable multiple parallel episodes, environments or agents within an environment (: 1).
- **config** (*specification*) – Additional configuration options:
- **saver** (*path | specification*) – TensorFlow checkpoints directory, or checkpoint manager configuration with the following attributes, for periodic implicit saving as alternative to explicit saving via agent.save() (: no saver):

- **summarizer** (*path | specification*) – TensorBoard summaries directory, or summarizer configuration with the following attributes (: no summarizer):). max_summaries (int > 0) – maximum number of (generically-named) summaries to keep (default: 7, number of different colors in Tensorboard). flush (int > 0) – how frequently in seconds to flush the summary writer (default: 10). summaries ("all" | iter[string]) – which summaries to record, "all" implies all numerical summaries, so all summaries except "graph" (default: "all"): "action-value": value of each action (timestep-based) "distribution": distribution parameters like probabilities or mean and stddev (timestep-based) "entropy": entropy of (per-action) policy distribution(s) (timestep-based) "graph": computation graph "kl-divergence": KL-divergence of previous and updated (per-action) policy distribution(s) (update-based) "loss": policy and baseline loss plus loss components (update-based) "parameters": parameter values (according to parameter unit) "reward": timestep and episode reward, plus intermediate reward/return estimates (timestep/episode/update-based) "update-norm": global norm of update (update-based) "updates": mean and variance of update tensors per variable (update-based) "variables": mean of trainable variables tensors (update-based) ‘
- **recorder** (*path | specification*) – Traces recordings directory, or recorder configuration with the following attributes (see record-and-pretrain script for example application) (: no recorder):

CHAPTER 11

Vanilla Policy Gradient

class tensorforce.agents.**VanillaPolicyGradient**(*states*, *actions*, *max_episode_timesteps*, *batch_size*, *network='auto'*, *use_beta_distribution=False*, *memory='minimum'*, *update_frequency='batch_size'*, *learning_rate=0.001*, *discount=0.99*, *predict_terminal_values=False*, *baseline=None*, *baseline_optimizer=None*, *state_preprocessing='linear_normalization'*, *reward_preprocessing=None*, *exploration=0.0*, *variable_noise=0.0*, *l2_regularization=0.0*, *entropy_regularization=0.0*, *parallel_interactions=1*, *config=None*, *saver=None*, *summarizer=None*, *recorder=None*, *estimate_terminal=None*, *baseline_network=None*, ***kwargs*)

Vanilla Policy Gradient aka REINFORCE agent (specification key: `vpg` or `reinforce`).

Parameters

- **states** (*specification*) – States specification (, better implicitly specified via `environment` argument for `Agent.create(...)`), arbitrarily nested dictionary of state descriptions (usually taken from `Environment.states()`) with the following attributes:
- **actions** (*specification*) – Actions specification (, better implicitly specified via `environment` argument for `Agent.create(...)`), arbitrarily nested dictionary of action descriptions (usually taken from `Environment.actions()`) with the following attributes:
- **max_episode_timesteps** (*int > 0*) – Upper bound for numer of timesteps per episode (: not given, better implicitly specified via `environment` argument for `Agent.`

`create(...)`).

- **batch_size** (, int > 0) – Number of episodes per update batch ().
- **network** (*"auto" | specification*) – Policy network configuration, see the networks documentation (: "auto", automatically configured network).
- **use_beta_distribution** (*bool*) – Whether to use the Beta distribution for bounded continuous actions by default. (: false).
- **memory** (*int > 0*) – Batch memory capacity, has to fit at least maximum batch_size + 1 episodes (: minimum capacity, usually does not need to be changed).
- **update_frequency** ("never" | , int > 0) – Frequency of updates (: batch_size).
- **learning_rate** (, float > 0.0) – Optimizer learning rate (: 1e-3).
- **discount** (, 0.0 <= float <= 1.0) – Discount factor for future rewards of discounted-sum reward estimation (: 0.99).
- **predict_terminal_values** (*bool*) – Whether to predict the value of terminal states (: false).
- **baseline** (*specification*) – Baseline network configuration, see the networks documentation, main policy will be used as baseline if none (: none).
- **baseline_optimizer** (*float > 0.0 | specification*) – Baseline optimizer configuration, see the optimizers documentation, main optimizer will be used for baseline if none, a float implies none and specifies a custom weight for the baseline loss (: none).
- **l2_regularization** (, float >= 0.0) – L2 regularization loss weight (: no L2 regularization).
- **entropy_regularization** (, float >= 0.0) – Entropy regularization loss weight, to discourage the policy distribution from being "too certain" (: no entropy regularization).
- **state_preprocessing** (*dict[specification]*) – State preprocessing as layer or list of layers, see the preprocessing documentation, specified per state-type or -name (: linear normalization of bounded float states to [-2.0, 2.0]).
- **reward_preprocessing** (*specification*) – Reward preprocessing as layer or list of layers, see the preprocessing documentation (: no reward preprocessing).
- **exploration** (| dict[], float >= 0.0) – Exploration, defined as the probability for uniformly random output in case of `bool` and `int` actions, and the standard deviation of Gaussian noise added to every output in case of `float` actions, specified globally or per action-type or -name (: no exploration).
- **variable_noise** (, float >= 0.0) – Add Gaussian noise with given standard deviation to all trainable variables, as alternative exploration mechanism (: no variable noise).
- **others** – See the Tensorforce agent documentation.

CHAPTER 12

Proximal Policy Optimization

class tensorforce.agents.**ProximalPolicyOptimization**(*states*, *actions*, *max_episode_timesteps*, *batch_size*, *network='auto'*, *use_beta_distribution=False*, *memory='minimum'*, *update_frequency='batch_size'*, *learning_rate=0.001*, *multi_step=10*, *subsampling_fraction=0.33*, *likelihood_ratio_clipping=0.25*, *discount=0.99*, *predict_terminal_values=False*, *baseline=None*, *baseline_optimizer=None*, *state_preprocessing='linear_normalization'*, *reward_preprocessing=None*, *exploration=0.0*, *variable_noise=0.0*, *l2_regularization=0.0*, *entropy_regularization=0.0*, *parallel_interactions=1*, *config=None*, *saver=None*, *summarizer=None*, *recorder=None*, *optimization_steps=None*, *estimate_terminal=None*, *critic_network=None*, *baseline_network=None*, *critic_optimizer=None*, ***kwargs*)

Proximal Policy Optimization agent (specification key: `ppo`).

Parameters

- **states** (*specification*) – States specification (, better implicitly specified via `environment` argument for `Agent.create(...)`), arbitrarily nested dictionary of state descriptions (usually taken from `Environment.states()`) with the following attributes:
- **actions** (*specification*) – Actions specification (, better implicitly specified via `environment` argument for `Agent.create(...)`), arbitrarily nested dictionary of action descriptions (usually taken from `Environment.actions()`) with the following attributes:
- **max_episode_timesteps** (*int > 0*) – Upper bound for numer of timesteps per episode (: not given, better implicitly specified via `environment` argument for `Agent.create(...)`).
- **batch_size** (, int > 0) – Number of episodes per update batch ().
- **network** (*"auto" | specification*) – Policy network configuration, see the networks documentation (: "auto", automatically configured network).
- **use_beta_distribution** (*bool*) – Whether to use the Beta distribution for bounded continuous actions by default. (: false).
- **memory** (*int > 0*) – Batch memory capacity, has to fit at least maximum batch_size + 1 episodes (: minimum capacity, usually does not need to be changed).
- **update_frequency** ("never" | , int > 0) – Frequency of updates (: batch_size).
- **learning_rate** (, float > 0.0) – Optimizer learning rate (: 1e-3).
- **multi_step** (, int >= 1) – Number of optimization steps (: 10).
- **subsampling_fraction** (, int > 0 | 0.0 < float <= 1.0) – Absolute/relative fraction of batch timesteps to subsample (: 0.33).
- **likelihood_ratio_clipping** (, float > 0.0) – Likelihood-ratio clipping threshold (: 0.25).
- **discount** (, 0.0 <= float <= 1.0) – Discount factor for future rewards of discounted-sum reward estimation (: 0.99).
- **predict_terminal_values** (*bool*) – Whether to predict the value of terminal states (: false).
- **baseline** (*specification*) – Baseline network configuration, see the networks documentation, main policy will be used as baseline if none (: none).
- **baseline_optimizer** (*float > 0.0 | specification*) – Baseline optimizer configuration, see the optimizers documentation, main optimizer will be used for baseline if none, a float implies none and specifies a custom weight for the baseline loss (: none).
- **l2_regularization** (, float >= 0.0) – L2 regularization loss weight (: no L2 regularization).
- **entropy_regularization** (, float >= 0.0) – Entropy regularization loss weight, to discourage the policy distribution from being "too certain" (: no entropy regularization).
- **state_preprocessing** (*dict[specification]*) – State preprocessing as layer or list of layers, see the preprocessing documentation, specified per state-type or -name (: linear normalization of bounded float states to [-2.0, 2.0]).
- **reward_preprocessing** (*specification*) – Reward preprocessing as layer or list of layers, see the preprocessing documentation (: no reward preprocessing).

- **exploration** (| dict[], float >= 0.0) – Exploration, defined as the probability for uniformly random output in case of `bool` and `int` actions, and the standard deviation of Gaussian noise added to every output in case of `float` actions, specified globally or per action-type or -name (: no exploration).
- **variable_noise** (, float >= 0.0) – Add Gaussian noise with given standard deviation to all trainable variables, as alternative exploration mechanism (: no variable noise).
- **others** – See the Tensorforce agent documentation.

CHAPTER 13

Trust-Region Policy Optimization

class tensorforce.agents.**TrustRegionPolicyOptimization**(*states*, *actions*, *max_episode_timesteps*, *batch_size*, *network='auto'*, *use_beta_distribution=False*, *memory='minimum'*, *update_frequency='batch_size'*, *learning_rate=0.01*, *line_search_iterations=10*, *subsampling_fraction=1.0*, *discount=0.99*, *predict_terminal_values=False*, *baseline=None*, *baseline_optimizer=None*, *state_preprocessing='linear_normalization'*, *reward_preprocessing=None*, *exploration=0.0*, *variable_noise=0.0*, *l2_regularization=0.0*, *entropy_regularization=0.0*, *parallel_interactions=1*, *config=None*, *saver=None*, *summarizer=None*, *recorder=None*, *estimate_terminal=None*, *critic_network=None*, *baseline_network=None*, *critic_optimizer=None*, ***kwargs*)

Trust Region Policy Optimization agent (specification key: `trpo`).

Parameters

- **states** (*specification*) – States specification (, better implicitly specified via `environment` argument for `Agent.create(...)`), arbitrarily nested dictionary of state descriptions (usually taken from `Environment.states()`) with the following attributes:
- **actions** (*specification*) – Actions specification (, better implicitly specified via `environment` argument for `Agent.create(...)`), arbitrarily nested dictionary of action descriptions (usually taken from `Environment.actions()`) with the following attributes:
- **max_episode_timesteps** (*int > 0*) – Upper bound for numer of timesteps per episode (: not given, better implicitly specified via `environment` argument for `Agent.create(...)`).
- **batch_size** (, int > 0) – Number of episodes per update batch ().
- **network** (*"auto" | specification*) – Policy network configuration, see the networks documentation (: "auto", automatically configured network).
- **use_beta_distribution** (*bool*) – Whether to use the Beta distribution for bounded continuous actions by default. (: false).
- **memory** (*int > 0*) – Batch memory capacity, has to fit at least maximum batch_size + 1 episodes (: minimum capacity, usually does not need to be changed).
- **update_frequency** ("never" | , int > 0) – Frequency of updates (: batch_size).
- **learning_rate** (, float > 0.0) – Optimizer learning rate (: 1e-2).
- **linesearch_iterations** (, int >= 0) – Maximum number of line search iterations (: 10).
- **subsampling_fraction** (, int > 0 | 0.0 < float <= 1.0) – Absolute/relative fraction of batch timesteps to subsample for computation of natural gradient update (: no subsampling).
- **discount** (, 0.0 <= float <= 1.0) – Discount factor for future rewards of discounted-sum reward estimation (: 0.99).
- **predict_terminal_values** (*bool*) – Whether to predict the value of terminal states (: false).
- **baseline** (*specification*) – Baseline network configuration, see the networks documentation, main policy will be used as baseline if none (: none).
- **baseline_optimizer** (*float > 0.0 | specification*) – Baseline optimizer configuration, see the optimizers documentation, main optimizer will be used for baseline if none, a float implies none and specifies a custom weight for the baseline loss (: none).
- **l2_regularization** (, float >= 0.0) – L2 regularization loss weight (: no L2 regularization).
- **entropy_regularization** (, float >= 0.0) – Entropy regularization loss weight, to discourage the policy distribution from being "too certain" (: no entropy regularization).
- **state_preprocessing** (*dict[specification]*) – State preprocessing as layer or list of layers, see the preprocessing documentation, specified per state-type or -name (: linear normalization of bounded float states to [-2.0, 2.0]).
- **reward_preprocessing** (*specification*) – Reward preprocessing as layer or list of layers, see the preprocessing documentation (: no reward preprocessing).
- **exploration** (| dict[], float >= 0.0) – Exploration, defined as the probability for uniformly random output in case of `bool` and `int` actions, and the standard deviation of

Gaussian noise added to every output in case of `float` actions, specified globally or per action-type or -name (: no exploration).

- **`variable_noise`** (, float >= 0.0) – Add Gaussian noise with given standard deviation to all trainable variables, as alternative exploration mechanism (: no variable noise).
- **`others`** – See the Tensorforce agent documentation.

CHAPTER 14

Deterministic Policy Gradient

class tensorforce.agents.**DeterministicPolicyGradient**(*states*, *actions*, *memory*, *batch_size*, *max_episode_timesteps=None*, *network='auto'*, *use_beta_distribution=True*, *update_frequency='batch_size'*, *start_updating=None*, *learning_rate=0.001*, *horizon=1*, *discount=0.99*, *predict_terminal_values=False*, *critic='auto'*, *critic_optimizer=1.0*, *state_preprocessing='linear_normalization'*, *reward_preprocessing=None*, *exploration=0.1*, *variable_noise=0.0*, *l2_regularization=0.0*, *entropy_regularization=0.0*, *parallel_interactions=1*, *config=None*, *saver=None*, *summarizer=None*, *recorder=None*, *estimate_terminal=None*, *critic_network=None*, ***kwargs*)

Deterministic Policy Gradient agent (specification key: `dpg` or `ddpg`). Action space is required to consist of only a single float action.

Parameters

- **states** (*specification*) – States specification (, better implicitly specified via `environment` argument for `Agent.create(...)`), arbitrarily nested dictionary of

state descriptions (usually taken from `Environment.states()`) with the following attributes:

- **actions** (*specification*) – Actions specification (, better implicitly specified via `environment` argument for `Agent.create(...)`), arbitrarily nested dictionary of action descriptions (usually taken from `Environment.actions()`) with the following attributes:
- **max_episode_timesteps** (*int > 0*) – Upper bound for numer of timesteps per episode (: not given, better implicitly specified via `environment` argument for `Agent.create(...)`).
- **memory** (*int > 0*) – Replay memory capacity, has to fit at least maximum batch_size + maximum network/estimator horizon + 1 timesteps ().
- **batch_size** (, int > 0) – Number of timesteps per update batch ().
- **network** (*"auto" | specification*) – Policy network configuration, see the networks documentation (: "auto", automatically configured network).
- **use_beta_distribution** (*bool*) – Whether to use the Beta distribution for bounded continuous actions by default. (: true).
- **update_frequency** ("never" | , int > 0) – Frequency of updates (: batch_size).
- **start_updating** (, int >= batch_size) – Number of timesteps before first update (: none).
- **learning_rate** (, float > 0.0) – Optimizer learning rate (: 1e-3).
- **horizon** (, int >= 1) – Horizon of discounted-sum reward estimation before critic estimate (: 1).
- **discount** (, 0.0 <= float <= 1.0) – Discount factor for future rewards of discounted-sum reward estimation (: 0.99).
- **predict_terminal_values** (*bool*) – Whether to predict the value of terminal states (: false).
- **critic** (*specification*) – Critic network configuration, see the networks documentation (: none).
- **critic_optimizer** (*float > 0.0 | specification*) – Critic optimizer configuration, see the optimizers documentation, a float instead specifies a custom weight for the critic loss (: 1.0).
- **l2_regularization** (, float >= 0.0) – L2 regularization loss weight (: no L2 regularization).
- **entropy_regularization** (, float >= 0.0) – Entropy regularization loss weight, to discourage the policy distribution from being "too certain" (: no entropy regularization).
- **state_preprocessing** (*dict[specification]*) – State preprocessing as layer or list of layers, see the preprocessing documentation, specified per state-type or -name (: linear normalization of bounded float states to [-2.0, 2.0]).
- **reward_preprocessing** (*specification*) – Reward preprocessing as layer or list of layers, see the preprocessing documentation (: no reward preprocessing).
- **exploration** (| dict[], float >= 0.0) – Exploration, defined as the probability for uniformly random output in case of `bool` and `int` actions, and the standard deviation of Gaussian noise added to every output in case of `float` actions, specified globally or per action-type or -name (: 0.1 standard deviation).

- **variable_noise** (, float >= 0.0) – Add Gaussian noise with given standard deviation to all trainable variables, as alternative exploration mechanism (: no variable noise).
- **others** – See the Tensorforce agent documentation.

CHAPTER 15

Deep Q-Network

class tensorforce.agents.**DeepQNetwork**(*states*, *actions*, *memory*, *batch_size*, *max_episode_timesteps=None*, *network='auto'*, *update_frequency='batch_size'*, *start_updating=None*, *learning_rate=0.001*, *huber_loss=None*, *horizon=1*, *discount=0.99*, *predict_terminal_values=False*, *target_update_weight=1.0*, *target_sync_frequency=1*, *state_preprocessing='linear_normalization'*, *reward_preprocessing=None*, *exploration=0.0*, *variable_noise=0.0*, *l2_regularization=0.0*, *entropy_regularization=0.0*, *parallel_interactions=1*, *config=None*, *saver=None*, *summarizer=None*, *recorder=None*, *estimate_terminal=None*, ***kwargs*)

Deep Q-Network agent (specification key: `dqn`).

Parameters

- **states** (*specification*) – States specification (, better implicitly specified via `environment` argument for `Agent.create(...)`), arbitrarily nested dictionary of state descriptions (usually taken from `Environment.states()`) with the following attributes:
- **actions** (*specification*) – Actions specification (, better implicitly specified via `environment` argument for `Agent.create(...)`), arbitrarily nested dictionary of action descriptions (usually taken from `Environment.actions()`) with the following attributes:
- **max_episode_timesteps** (*int > 0*) – Upper bound for numer of timesteps per episode (: not given, better implicitly specified via `environment` argument for `Agent.create(...)`).
- **memory** (*int > 0*) – Replay memory capacity, has to fit at least maximum batch_size + maximum network/estimator horizon + 1 timesteps ().
- **batch_size** (, int > 0) – Number of timesteps per update batch ().

- **network** (*"auto" | specification*) – Policy network configuration, see the networks documentation (: "auto", automatically configured network).
- **update_frequency** ("never" | , int > 0) – Frequency of updates (: batch_size).
- **start_updating** (, int >= batch_size) – Number of timesteps before first update (: none).
- **learning_rate** (, float > 0.0) – Optimizer learning rate (: 1e-3).
- **huber_loss** (, float > 0.0) – Huber loss threshold (: no huber loss).
- **horizon** (, int >= 1) – n-step DQN, horizon of discounted-sum reward estimation before target network estimate (: 1).
- **discount** (, 0.0 <= float <= 1.0) – Discount factor for future rewards of discounted-sum reward estimation (: 0.99).
- **predict_terminal_values** (*bool*) – Whether to predict the value of terminal states (: false).
- **target_update_weight** (, 0.0 < float <= 1.0) – Target network update weight (: 1.0).
- **target_sync_frequency** (, int >= 1) – Interval between target network updates (: every update).
- **l2_regularization** (, float >= 0.0) – L2 regularization loss weight (: no L2 regularization).
- **entropy_regularization** (, float >= 0.0) – Entropy regularization loss weight, to discourage the policy distribution from being "too certain" (: no entropy regularization).
- **state_preprocessing** (*dict[specification]*) – State preprocessing as layer or list of layers, see the preprocessing documentation, specified per state-type or -name (: linear normalization of bounded float states to [-2.0, 2.0]).
- **reward_preprocessing** (*specification*) – Reward preprocessing as layer or list of layers, see the preprocessing documentation (: no reward preprocessing).
- **exploration** (| dict[], float >= 0.0) – Exploration, defined as the probability for uniformly random output in case of `bool` and `int` actions, and the standard deviation of Gaussian noise added to every output in case of `float` actions, specified globally or per action-type or -name (: no exploration).
- **variable_noise** (, float >= 0.0) – Add Gaussian noise with given standard deviation to all trainable variables, as alternative exploration mechanism (: no variable noise).
- **others** – See the Tensorforce agent documentation.

CHAPTER 16

Double DQN

class tensorforce.agents.**DoubleDQN**(*states*, *actions*, *memory*, *batch_size*, *max_episode_timesteps=None*, *network='auto'*, *update_frequency='batch_size'*, *start_updating=None*, *learning_rate=0.001*, *huber_loss=None*, *horizon=1*, *discount=0.99*, *predict_terminal_values=False*, *target_update_weight=1.0*, *target_sync_frequency=1*, *state_preprocessing='linear_normalization'*, *reward_preprocessing=None*, *exploration=0.0*, *variable_noise=0.0*, *l2_regularization=0.0*, *entropy_regularization=0.0*, *parallel_interactions=1*, *config=None*, *saver=None*, *summarizer=None*, *recorder=None*, *estimate_terminal=None*, ***kwargs*)

Double DQN agent (specification key: `double_dqn` or `ddqn`).

Parameters

- **states** (*specification*) – States specification (, better implicitly specified via `environment` argument for `Agent.create(...)`), arbitrarily nested dictionary of state descriptions (usually taken from `Environment.states()`) with the following attributes:
- **actions** (*specification*) – Actions specification (, better implicitly specified via `environment` argument for `Agent.create(...)`), arbitrarily nested dictionary of action descriptions (usually taken from `Environment.actions()`) with the following attributes:
- **max_episode_timesteps** (*int > 0*) – Upper bound for numer of timesteps per episode (: not given, better implicitly specified via `environment` argument for `Agent.create(...)`).
- **memory** (*int > 0*) – Replay memory capacity, has to fit at least maximum batch_size + maximum network/estimator horizon + 1 timesteps ().
- **batch_size** (, int > 0) – Number of timesteps per update batch ().

- **network** (*"auto" | specification*) – Policy network configuration, see the networks documentation (: "auto", automatically configured network).
- **update_frequency** ("never" | , int > 0) – Frequency of updates (: batch_size).
- **start_updating** (, int >= batch_size) – Number of timesteps before first update (: none).
- **learning_rate** (, float > 0.0) – Optimizer learning rate (: 1e-3).
- **huber_loss** (, float > 0.0) – Huber loss threshold (: no huber loss).
- **horizon** (, int >= 1) – n-step DQN, horizon of discounted-sum reward estimation before target network estimate (: 1).
- **discount** (, 0.0 <= float <= 1.0) – Discount factor for future rewards of discounted-sum reward estimation (: 0.99).
- **predict_terminal_values** (*bool*) – Whether to predict the value of terminal states (: false).
- **target_update_weight** (, 0.0 < float <= 1.0) – Target network update weight (: 1.0).
- **target_sync_frequency** (, int >= 1) – Interval between target network updates (: every update).
- **l2_regularization** (, float >= 0.0) – L2 regularization loss weight (: no L2 regularization).
- **entropy_regularization** (, float >= 0.0) – Entropy regularization loss weight, to discourage the policy distribution from being "too certain" (: no entropy regularization).
- **state_preprocessing** (*dict[specification]*) – State preprocessing as layer or list of layers, see the preprocessing documentation, specified per state-type or -name (: linear normalization of bounded float states to [-2.0, 2.0]).
- **reward_preprocessing** (*specification*) – Reward preprocessing as layer or list of layers, see the preprocessing documentation (: no reward preprocessing).
- **exploration** (| dict[], float >= 0.0) – Exploration, defined as the probability for uniformly random output in case of `bool` and `int` actions, and the standard deviation of Gaussian noise added to every output in case of `float` actions, specified globally or per action-type or -name (: no exploration).
- **variable_noise** (, float >= 0.0) – Add Gaussian noise with given standard deviation to all trainable variables, as alternative exploration mechanism (: no variable noise).
- **others** – See the Tensorforce agent documentation.

CHAPTER 17

Dueling DQN

class tensorforce.agents.**DuelingDQN**(*states*, *actions*, *memory*, *batch_size*, *max_episode_timesteps=None*, *network='auto'*, *update_frequency='batch_size'*, *start_updating=None*, *learning_rate=0.001*, *huber_loss=None*, *horizon=1*, *discount=0.99*, *predict_terminal_values=False*, *target_update_weight=1.0*, *target_sync_frequency=1*, *state_preprocessing='linear_normalization'*, *reward_preprocessing=None*, *exploration=0.0*, *variable_noise=0.0*, *l2_regularization=0.0*, *entropy_regularization=0.0*, *parallel_interactions=1*, *config=None*, *saver=None*, *summarizer=None*, *recorder=None*, *estimate_terminal=None*, ***kwargs*)

Dueling DQN agent (specification key: `dueling_dqn`).

Parameters

- **states** (*specification*) – States specification (, better implicitly specified via `environment` argument for `Agent.create(...)`), arbitrarily nested dictionary of state descriptions (usually taken from `Environment.states()`) with the following attributes:
- **actions** (*specification*) – Actions specification (, better implicitly specified via `environment` argument for `Agent.create(...)`), arbitrarily nested dictionary of action descriptions (usually taken from `Environment.actions()`) with the following attributes:
- **max_episode_timesteps** (*int > 0*) – Upper bound for numer of timesteps per episode (: not given, better implicitly specified via `environment` argument for `Agent.create(...)`).
- **memory** (*int > 0*) – Replay memory capacity, has to fit at least maximum batch_size + maximum network/estimator horizon + 1 timesteps ().
- **batch_size** (, int > 0) – Number of timesteps per update batch ().

- **network** (*"auto" | specification*) – Policy network configuration, see the networks documentation (: "auto", automatically configured network).
- **update_frequency** ("never" | , int > 0) – Frequency of updates (: batch_size).
- **start_updating** (, int >= batch_size) – Number of timesteps before first update (: none).
- **learning_rate** (, float > 0.0) – Optimizer learning rate (: 1e-3).
- **huber_loss** (, float > 0.0) – Huber loss threshold (: no huber loss).
- **horizon** (, int >= 1) – n-step DQN, horizon of discounted-sum reward estimation before target network estimate (: 1).
- **discount** (, 0.0 <= float <= 1.0) – Discount factor for future rewards of discounted-sum reward estimation (: 0.99).
- **predict_terminal_values** (*bool*) – Whether to predict the value of terminal states (: false).
- **target_update_weight** (, 0.0 < float <= 1.0) – Target network update weight (: 1.0).
- **target_sync_frequency** (, int >= 1) – Interval between target network updates (: every update).
- **l2_regularization** (, float >= 0.0) – L2 regularization loss weight (: no L2 regularization).
- **entropy_regularization** (, float >= 0.0) – Entropy regularization loss weight, to discourage the policy distribution from being "too certain" (: no entropy regularization).
- **state_preprocessing** (*dict[specification]*) – State preprocessing as layer or list of layers, see the preprocessing documentation, specified per state-type or -name (: linear normalization of bounded float states to [-2.0, 2.0]).
- **reward_preprocessing** (*specification*) – Reward preprocessing as layer or list of layers, see the preprocessing documentation (: no reward preprocessing).
- **exploration** (| dict[], float >= 0.0) – Exploration, defined as the probability for uniformly random output in case of `bool` and `int` actions, and the standard deviation of Gaussian noise added to every output in case of `float` actions, specified globally or per action-type or -name (: no exploration).
- **variable_noise** (, float >= 0.0) – Add Gaussian noise with given standard deviation to all trainable variables, as alternative exploration mechanism (: no variable noise).
- **others** – See the Tensorforce agent documentation.

CHAPTER 18

Actor-Critic

class tensorforce.agents.**ActorCritic**(*states*, *actions*, *batch_size*, *max_episode_timesteps=None*, *network='auto'*, *use_beta_distribution=False*, *memory='minimum'*, *update_frequency='batch_size'*, *learning_rate=0.001*, *horizon=1*, *discount=0.99*, *predict_terminal_values=False*, *critic='auto'*, *critic_optimizer=1.0*, *state_preprocessing='linear_normalization'*, *reward_preprocessing=None*, *exploration=0.0*, *variable_noise=0.0*, *l2_regularization=0.0*, *entropy_regularization=0.0*, *parallel_interactions=1*, *config=None*, *saver=None*, *summarizer=None*, *recorder=None*, *estimate_terminal=None*, *critic_network=None*, ***kwargs*)

Actor-Critic agent (specification key: `ac`).

Parameters

- **states** (*specification*) – States specification (, better implicitly specified via `environment` argument for `Agent.create(...)`), arbitrarily nested dictionary of state descriptions (usually taken from `Environment.states()`) with the following attributes:
- **actions** (*specification*) – Actions specification (, better implicitly specified via `environment` argument for `Agent.create(...)`), arbitrarily nested dictionary of action descriptions (usually taken from `Environment.actions()`) with the following attributes:
- **max_episode_timesteps** (*int > 0*) – Upper bound for numer of timesteps per episode (: not given, better implicitly specified via `environment` argument for `Agent.create(...)`).
- **batch_size** (, int > 0) – Number of timesteps per update batch ().

- **network** (*"auto" | specification*) – Policy network configuration, see the networks documentation (: "auto", automatically configured network).
- **use_beta_distribution** (*bool*) – Whether to use the Beta distribution for bounded continuous actions by default. (: false).
- **memory** (*int > 0*) – Batch memory capacity, has to fit at least maximum batch_size + maximum network/estimator horizon + 1 timesteps (: minimum capacity, usually does not need to be changed).
- **update_frequency** ("never" | , int > 0) – Frequency of updates (: batch_size).
- **learning_rate** (, float > 0.0) – Optimizer learning rate (: 1e-3).
- **horizon** (, int >= 1) – Horizon of discounted-sum reward estimation before critic estimate (: 1).
- **discount** (, 0.0 <= float <= 1.0) – Discount factor for future rewards of discounted-sum reward estimation (: 0.99).
- **predict_terminal_values** (*bool*) – Whether to predict the value of terminal states (: false).
- **critic** (*specification*) – Critic network configuration, see the networks documentation (: "auto").
- **critic_optimizer** (*float > 0.0 | specification*) – Critic optimizer configuration, see the optimizers documentation, a float instead specifies a custom weight for the critic loss (: 1.0).
- **l2_regularization** (, float >= 0.0) – L2 regularization loss weight (: no L2 regularization).
- **entropy_regularization** (, float >= 0.0) – Entropy regularization loss weight, to discourage the policy distribution from being "too certain" (: no entropy regularization).
- **state_preprocessing** (*dict[specification]*) – State preprocessing as layer or list of layers, see the preprocessing documentation, specified per state-type or -name (: linear normalization of bounded float states to [-2.0, 2.0]).
- **reward_preprocessing** (*specification*) – Reward preprocessing as layer or list of layers, see the preprocessing documentation (: no reward preprocessing).
- **exploration** (| dict[], float >= 0.0) – Exploration, defined as the probability for uniformly random output in case of `bool` and `int` actions, and the standard deviation of Gaussian noise added to every output in case of `float` actions, specified globally or per action-type or -name (: no exploration).
- **variable_noise** (, float >= 0.0) – Add Gaussian noise with given standard deviation to all trainable variables, as alternative exploration mechanism (: no variable noise).
- **others** – See the Tensorforce agent documentation.

CHAPTER 19

Advantage Actor-Critic

class tensorforce.agents.**AdvantageActorCritic**(*states*, *actions*, *batch_size*, *max_episode_timesteps=None*, *network='auto'*, *use_beta_distribution=False*, *memory='minimum'*, *update_frequency='batch_size'*, *learning_rate=0.001*, *horizon=1*, *discount=0.99*, *predict_terminal_values=False*, *critic='auto'*, *critic_optimizer=1.0*, *state_preprocessing='linear_normalization'*, *reward_preprocessing=None*, *exploration=0.0*, *variable_noise=0.0*, *l2_regularization=0.0*, *entropy_regularization=0.0*, *parallel_interactions=1*, *config=None*, *saver=None*, *summarizer=None*, *recorder=None*, *estimate_terminal=None*, *critic_network=None*, ***kwargs*)

Advantage Actor-Critic agent (specification key: `a2c`).

Parameters

- **states** (*specification*) – States specification (, better implicitly specified via `environment` argument for `Agent.create(...)`), arbitrarily nested dictionary of state descriptions (usually taken from `Environment.states()`) with the following attributes:
- **actions** (*specification*) – Actions specification (, better implicitly specified via `environment` argument for `Agent.create(...)`), arbitrarily nested dictionary of action descriptions (usually taken from `Environment.actions()`) with the following attributes:

- **max_episode_timesteps** (*int > 0*) – Upper bound for numer of timesteps per episode (: not given, better implicitly specified via `environment` argument for `Agent.create(...)`).
- **batch_size** (, int > 0) – Number of timesteps per update batch ().
- **network** (*"auto" | specification*) – Policy network configuration, see the networks documentation (: "auto", automatically configured network).
- **use_beta_distribution** (*bool*) – Whether to use the Beta distribution for bounded continuous actions by default. (: false).
- **memory** (*int > 0*) – Batch memory capacity, has to fit at least maximum batch_size + maximum network/estimator horizon + 1 timesteps (: minimum capacity, usually does not need to be changed).
- **update_frequency** ("never" | , int > 0) – Frequency of updates (: batch_size).
- **learning_rate** (, float > 0.0) – Optimizer learning rate (: 1e-3).
- **horizon** ("episode" | , int >= 0) – Horizon of discounted-sum reward estimation before critic estimate (: 1).
- **discount** (, 0.0 <= float <= 1.0) – Discount factor for future rewards of discounted-sum reward estimation (: 0.99).
- **predict_terminal_values** (*bool*) – Whether to predict the value of terminal states (: false).
- **critic** (*specification*) – Critic network configuration, see the networks documentation (: "auto").
- **critic_optimizer** (*float > 0.0 | specification*) – Critic optimizer configuration, see the optimizers documentation, a float instead specifies a custom weight for the critic loss (: 1.0).
- **l2_regularization** (, float >= 0.0) – L2 regularization loss weight (: no L2 regularization).
- **entropy_regularization** (, float >= 0.0) – Entropy regularization loss weight, to discourage the policy distribution from being "too certain" (: no entropy regularization).
- **state_preprocessing** (*dict[specification]*) – State preprocessing as layer or list of layers, see the preprocessing documentation, specified per state-type or -name (: linear normalization of bounded float states to [-2.0, 2.0]).
- **reward_preprocessing** (*specification*) – Reward preprocessing as layer or list of layers, see the preprocessing documentation (: no reward preprocessing).
- **exploration** (| dict[], float >= 0.0) – Exploration, defined as the probability for uniformly random output in case of `bool` and `int` actions, and the standard deviation of Gaussian noise added to every output in case of `float` actions, specified globally or per action-type or -name (: no exploration).
- **variable_noise** (, float >= 0.0) – Add Gaussian noise with given standard deviation to all trainable variables, as alternative exploration mechanism (: no variable noise).
- **others** – See the Tensorforce agent documentation.

CHAPTER 20

Distributions

Distributions are customized via the `distributions` argument of `policy`, for instance:

```
Agent.create(
    ...
    policy=dict(distributions=dict(
        float=dict(type='gaussian', global_stddev=True),
        bounded_action=dict(type='beta')
    ))
    ...
)
```

See the policies documentation for more information about how to specify a policy.

class tensorforce.core.distributions.**Categorical**(*, *name=None*, *action_spec=None*, *input_spec=None*)

Categorical distribution, for discrete integer actions (specification key: `categorical`).

Parameters

- **name** (*string*) – .
- **action_spec** (*specification*) – .
- **input_spec** (*specification*) – .

class tensorforce.core.distributions.**Gaussian**(*, *global_stddev=False*, *bounded_transform='tanh'*, *name=None*, *action_spec=None*, *input_spec=None*)

Gaussian distribution, for continuous actions (specification key: `gaussian`).

Parameters

- **global_stddev** (*bool*) – Whether to use a separate set of trainable weights to parametrize standard deviation, instead of a state-dependent linear transformation (: false).
- **bounded_transform** (*"clipping" | "tanh"*) – Transformation to adjust sampled actions in case of bounded action space (: tanh).

- **name** (*string*) – .
- **action_spec** (*specification*) – .
- **input_spec** (*specification*) – .

class tensorforce.core.distributions.**Bernoulli**(*, *name=None*, *action_spec=None*, *input_spec=None*)

Bernoulli distribution, for binary boolean actions (specification key: `bernoulli`).

Parameters

- **name** (*string*) – .
- **action_spec** (*specification*) – .
- **input_spec** (*specification*) – .

class tensorforce.core.distributions.**Beta**(*, *name=None*, *action_spec=None*, *input_spec=None*)

Beta distribution, for bounded continuous actions (specification key: `beta`).

Parameters

- **name** (*string*) – .
- **action_spec** (*specification*) – .
- **input_spec** (*specification*) – .

CHAPTER 21

Layers

See the networks documentation for more information about how to specify networks.

Default layer: `Function` with default argument `function`, so a `lambda` function is a short-form specification of a simple transformation layer:

```
Agent.create(
    ...
    policy=dict(network=[
        (lambda x: tf.clip_by_value(x, -1.0, 1.0)),
        ...
    ]),
    ...
)
```

21.1 Dense layers

class tensorforce.core.layers.**Dense**(*, *size*, *bias=True*, *activation='tanh'*, *dropout=0.0*, *initialization_scale=1.0*, *vars_trainable=True*, *l2_regularization=None*, *name=None*, *input_spec=None*)

Dense fully-connected layer (specification key: `dense`).

Parameters

- **size** (*int >= 0*) – Layer output size, 0 implies additionally removing the axis ().
- **bias** (*bool*) – Whether to add a trainable bias variable (: true).
- **('crelu' | 'elu' | 'leaky-relu' | 'none' | 'relu' | 'selu' | 'sigmoid' |** (*activation*) – 'softmax' | 'softplus' | 'softsign' | 'swish' | 'tanh'): Activation nonlinearity (: tanh).
- **dropout** (*parameter, 0.0 <= float < 1.0*) – Dropout rate (: 0.0).
- **initialization_scale** (*float > 0.0*) – Initialization scale (: 1.0).

- **vars_trainable** (*bool*) – Whether layer variables are trainable (: true).
- **l2_regularization** (*float >= 0.0*) – Scalar controlling L2 regularization (: inherit value of parent module).
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .

class tensorforce.core.layers.**Linear**(*, *size*, *bias=True*, *initialization_scale=1.0*, *vars_trainable=True*, *l2_regularization=None*, *name=None*, *input_spec=None*)

Linear layer (specification key: `linear`).

Parameters

- **size** (*int >= 0*) – Layer output size, 0 implies additionally removing the axis ().
- **bias** (*bool*) – Whether to add a trainable bias variable (: true).
- **initialization_scale** (*float > 0.0*) – Initialization scale (: 1.0).
- **vars_trainable** (*bool*) – Whether layer variables are trainable (: true).
- **l2_regularization** (*float >= 0.0*) – Scalar controlling L2 regularization (: inherit value of parent module).
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .

21.2 Convolutional layers

class tensorforce.core.layers.**Conv1d**(*, *size*, *window=3*, *stride=1*, *padding='same'*, *dilation=1*, *bias=True*, *activation='relu'*, *dropout=0.0*, *initialization_scale=1.0*, *vars_trainable=True*, *l2_regularization=None*, *name=None*, *input_spec=None*)

1-dimensional convolutional layer (specification key: `conv1d`).

Parameters

- **size** (*int >= 0*) – Layer output size, 0 implies additionally removing the axis ().
- **window** (*int > 0*) – Window size (: 3).
- **stride** (*int > 0*) – Stride size (: 1).
- **padding** (*'same' | 'valid'*) – Padding type, see TensorFlow docs (: 'same').
- **dilation** (*int > 0 | (int > 0, int > 0)*) – Dilation value (: 1).
- **bias** (*bool*) – Whether to add a trainable bias variable (: true).
- **('crelu' | 'elu' | 'leaky-relu' | 'none' | 'relu' | 'selu' | 'sigmoid' |** (*activation*) – 'softmax' | 'softplus' | 'softsign' | 'swish' | 'tanh'): Activation nonlinearity (: relu).
- **dropout** (*parameter, 0.0 <= float < 1.0*) – Dropout rate (: 0.0).
- **initialization_scale** (*float > 0.0*) – Initialization scale (: 1.0).
- **vars_trainable** (*bool*) – Whether layer variables are trainable (: true).

- **l2_regularization** (*float >= 0.0*) – Scalar controlling L2 regularization (: inherit value of parent module).
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .

class tensorforce.core.layers.**Conv2d**(*, *size*, *window=3*, *stride=1*, *padding='same'*, *dilation=1*, *bias=True*, *activation='relu'*, *dropout=0.0*, *initialization_scale=1.0*, *vars_trainable=True*, *l2_regularization=None*, *name=None*, *input_spec=None*)

2-dimensional convolutional layer (specification key: `conv2d`).

Parameters

- **size** (*int >= 0*) – Layer output size, 0 implies additionally removing the axis ().
- **window** (*int > 0 | (int > 0, int > 0)*) – Window size (: 3).
- **stride** (*int > 0 | (int > 0, int > 0)*) – Stride size (: 1).
- **padding** (*'same' | 'valid'*) – Padding type, see TensorFlow docs (: 'same').
- **dilation** (*int > 0 | (int > 0, int > 0)*) – Dilation value (: 1).
- **bias** (*bool*) – Whether to add a trainable bias variable (: true).
- **('crelu' | 'elu' | 'leaky-relu' | 'none' | 'relu' | 'selu' | 'sigmoid' |** (*activation*) – 'softmax' | 'softplus' | 'softsign' | 'swish' | 'tanh'): Activation nonlinearity (: "relu").
- **dropout** (*parameter, 0.0 <= float < 1.0*) – Dropout rate (: 0.0).
- **initialization_scale** (*float > 0.0*) – Initialization scale (: 1.0).
- **vars_trainable** (*bool*) – Whether layer variables are trainable (: true).
- **l2_regularization** (*float >= 0.0*) – Scalar controlling L2 regularization (: inherit value of parent module).
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .

class tensorforce.core.layers.**Conv1dTranspose**(*, *size*, *window=3*, *output_width=None*, *stride=1*, *padding='same'*, *dilation=1*, *bias=True*, *activation='relu'*, *dropout=0.0*, *initialization_scale=1.0*, *vars_trainable=True*, *l2_regularization=None*, *name=None*, *input_spec=None*)

1-dimensional transposed convolutional layer, also known as deconvolution layer (specification key: `deconv1d`).

Parameters

- **size** (*int >= 0*) – Layer output size, 0 implies additionally removing the axis ().
- **window** (*int > 0*) – Window size (: 3).
- **output_width** (*int > 0*) – Output width (: same as input).
- **stride** (*int > 0*) – Stride size (: 1).
- **padding** (*'same' | 'valid'*) – Padding type, see TensorFlow docs (: 'same').

- **dilation** (*int > 0 | (int > 0, int > 0)*) – Dilation value (: 1).
- **bias** (*bool*) – Whether to add a trainable bias variable (: true).
- **('crelu' | 'elu' | 'leaky-relu' | 'none' | 'relu' | 'selu' | 'sigmoid' |** (*activation*) – 'softmax' | 'softplus' | 'softsign' | 'swish' | 'tanh'): Activation nonlinearity (: "relu").
- **dropout** (*parameter, 0.0 <= float < 1.0*) – Dropout rate (: 0.0).
- **initialization_scale** (*float > 0.0*) – Initialization scale (: 1.0).
- **vars_trainable** (*bool*) – Whether layer variables are trainable (: true).
- **l2_regularization** (*float >= 0.0*) – Scalar controlling L2 regularization (: inherit value of parent module).
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .

class tensorforce.core.layers.**Conv2dTranspose**(*, *size*, *window=3*, *output_shape=None*, *stride=1*, *padding='same'*, *dilation=1*, *bias=True*, *activation='relu'*, *dropout=0.0*, *initialization_scale=1.0*, *vars_trainable=True*, *l2_regularization=None*, *name=None*, *input_spec=None*)

2-dimensional transposed convolutional layer, also known as deconvolution layer (specification key: `deconv2d`).

Parameters

- **size** (*int >= 0*) – Layer output size, 0 implies additionally removing the axis ().
- **window** (*int > 0 | (int > 0, int > 0)*) – Window size (: 3).
- **output_shape** (*int > 0 | (int > 0, int > 0)*) – Output shape (: same as input).
- **stride** (*int > 0 | (int > 0, int > 0)*) – Stride size (: 1).
- **padding** (*'same' | 'valid'*) – Padding type, see TensorFlow docs (: 'same').
- **dilation** (*int > 0 | (int > 0, int > 0)*) – Dilation value (: 1).
- **bias** (*bool*) – Whether to add a trainable bias variable (: true).
- **('crelu' | 'elu' | 'leaky-relu' | 'none' | 'relu' | 'selu' | 'sigmoid' |** (*activation*) – 'softmax' | 'softplus' | 'softsign' | 'swish' | 'tanh'): Activation nonlinearity (: "relu").
- **dropout** (*parameter, 0.0 <= float < 1.0*) – Dropout rate (: 0.0).
- **initialization_scale** (*float > 0.0*) – Initialization scale (: 1.0).
- **vars_trainable** (*bool*) – Whether layer variables are trainable (: true).
- **l2_regularization** (*float >= 0.0*) – Scalar controlling L2 regularization (: inherit value of parent module).
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .

21.3 Embedding layers

class tensorforce.core.layers.**Embedding**(*, *size*, *num_embeddings=None*, *max_norm=None*, *bias=True*, *activation='tanh'*, *dropout=0.0*, *vars_trainable=True*, *l2_regularization=None*, *name=None*, *input_spec=None*)

Embedding layer (specification key: `embedding`).

Parameters

- **size** (*int >= 0*) – Layer output size, 0 implies additionally removing the axis ().
- **num_embeddings** (*int > 0*) – If set, specifies the number of embeddings (: none).
- **max_norm** (*float*) – If set, embeddings are clipped if their L2-norm is larger (: none).
- **bias** (*bool*) – Whether to add a trainable bias variable (: true).
- **('crelu' | 'elu' | 'leaky-relu' | 'none' | 'relu' | 'selu' | 'sigmoid' |** (*activation*) – 'softmax' | 'softplus' | 'softsign' | 'swish' | 'tanh'): Activation nonlinearity (: tanh).
- **dropout** (*parameter, 0.0 <= float < 1.0*) – Dropout rate (: 0.0).
- **vars_trainable** (*bool*) – Whether layer variables are trainable (: true).
- **l2_regularization** (*float >= 0.0*) – Scalar controlling L2 regularization (: inherit value of parent module).
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .

21.4 Recurrent layers (unrolled over timesteps)

class tensorforce.core.layers.**Rnn**(*, *cell*, *size*, *horizon*, *bias=True*, *activation='tanh'*, *dropout=0.0*, *vars_trainable=True*, *l2_regularization=None*, *name=None*, *input_spec=None*, ***kwargs*)

Recurrent neural network layer which is unrolled over the sequence of timesteps (per episode), that is, the RNN cell is applied to the layer input at each timestep and the RNN consequently maintains a temporal internal state over the course of an episode (specification key: `rnn`).

Parameters

- **cell** (*'gru' | 'lstm'*) – The recurrent cell type ().
- **size** (*int >= 0*) – Layer output size, 0 implies additionally removing the axis ().
- **horizon** (*parameter, int >= 0*) – Past horizon, for truncated backpropagation through time ().
- **bias** (*bool*) – Whether to add a trainable bias variable (: true).
- **('crelu' | 'elu' | 'leaky-relu' | 'none' | 'relu' | 'selu' | 'sigmoid' |** (*activation*) – 'softmax' | 'softplus' | 'softsign' | 'swish' | 'tanh'): Activation nonlinearity (: tanh).
- **dropout** (*parameter, 0.0 <= float < 1.0*) – Dropout rate (: 0.0).
- **vars_trainable** (*bool*) – Whether layer variables are trainable (: true).

- **l2_regularization** (*float >= 0.0*) – Scalar controlling L2 regularization (: inherit value of parent module).
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .
- **kwargs** – Additional arguments for Keras RNN cell layer, see TensorFlow docs.

class tensorforce.core.layers.**Lstm**(*, *size*, *horizon*, *bias=False*, *activation=None*, *dropout=0.0*, *vars_trainable=True*, *l2_regularization=None*, *name=None*, *input_spec=None*, ***kwargs*)

Long short-term memory layer which is unrolled over the sequence of timesteps (per episode), that is, the LSTM cell is applied to the layer input at each timestep and the LSTM consequently maintains a temporal internal state over the course of an episode (specification key: `lstm`).

Parameters

- **size** (*int >= 0*) – Layer output size, 0 implies additionally removing the axis ().
- **horizon** (*parameter, int >= 0*) – Past horizon, for truncated backpropagation through time ().
- **bias** (*bool*) – Whether to add a trainable bias variable (: true).
- **('crelu' | 'elu' | 'leaky-relu' | 'none' | 'relu' | 'selu' | 'sigmoid' |** (*activation*) – 'softmax' | 'softplus' | 'softsign' | 'swish' | 'tanh'): Activation nonlinearity (: tanh).
- **dropout** (*parameter, 0.0 <= float < 1.0*) – Dropout rate (: 0.0).
- **vars_trainable** (*bool*) – Whether layer variables are trainable (: true).
- **l2_regularization** (*float >= 0.0*) – Scalar controlling L2 regularization (: inherit value of parent module).
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .
- **kwargs** – Additional arguments for Keras LSTM layer, see TensorFlow docs.

class tensorforce.core.layers.**Gru**(*, *size*, *horizon*, *bias=False*, *activation=None*, *dropout=0.0*, *vars_trainable=True*, *l2_regularization=None*, *name=None*, *input_spec=None*, ***kwargs*)

Gated recurrent unit layer which is unrolled over the sequence of timesteps (per episode), that is, the GRU cell is applied to the layer input at each timestep and the GRU consequently maintains a temporal internal state over the course of an episode (specification key: `gru`).

Parameters

- **size** (*int >= 0*) – Layer output size, 0 implies additionally removing the axis ().
- **horizon** (*parameter, int >= 0*) – Past horizon, for truncated backpropagation through time ().
- **bias** (*bool*) – Whether to add a trainable bias variable (: true).
- **('crelu' | 'elu' | 'leaky-relu' | 'none' | 'relu' | 'selu' | 'sigmoid' |** (*activation*) – 'softmax' | 'softplus' | 'softsign' | 'swish' | 'tanh'): Activation nonlinearity (: tanh).
- **dropout** (*parameter, 0.0 <= float < 1.0*) – Dropout rate (: 0.0).
- **vars_trainable** (*bool*) – Whether layer variables are trainable (: true).

- **l2_regularization** (*float >= 0.0*) – Scalar controlling L2 regularization (: inherit value of parent module).
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .
- **kwargs** – Additional arguments for Keras GRU layer, see TensorFlow docs.

21.5 Input recurrent layers (unrolled over sequence input)

class tensorforce.core.layers.**InputRnn**(**, cell, size, return_final_state=True, bias=True, activation='tanh', dropout=0.0, vars_trainable=True, l2_regularization=None, name=None, input_spec=None, **kwargs*)

Recurrent neural network layer which is unrolled over a sequence input independently per timestep, and consequently does not maintain an internal state (specification key: `input_rnn`).

Parameters

- **cell** (*'gru' | 'lstm'*) – The recurrent cell type ().
- **size** (*int >= 0*) – Layer output size, 0 implies additionally removing the axis ().
- **return_final_state** (*bool*) – Whether to return the final state instead of the per-step outputs (: true).
- **bias** (*bool*) – Whether to add a trainable bias variable (: true).
- **('crelu' | 'elu' | 'leaky-relu' | 'none' | 'relu' | 'selu' | 'sigmoid' |** (*activation*) – 'softmax' | 'softplus' | 'softsign' | 'swish' | 'tanh'): Activation nonlinearity (: tanh).
- **dropout** (*parameter, 0.0 <= float < 1.0*) – Dropout rate (: 0.0).
- **vars_trainable** (*bool*) – Whether layer variables are trainable (: true).
- **l2_regularization** (*float >= 0.0*) – Scalar controlling L2 regularization (: inherit value of parent module).
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .
- **kwargs** – Additional arguments for Keras RNN layer, see TensorFlow docs.

class tensorforce.core.layers.**InputLstm**(**, size, return_final_state=True, bias=False, activation=None, dropout=0.0, vars_trainable=True, l2_regularization=None, name=None, input_spec=None, **kwargs*)

Long short-term memory layer which is unrolled over a sequence input independently per timestep, and consequently does not maintain an internal state (specification key: `input_lstm`).

Parameters

- **size** (*int >= 0*) – Layer output size, 0 implies additionally removing the axis ().
- **return_final_state** (*bool*) – Whether to return the final state instead of the per-step outputs (: true).
- **bias** (*bool*) – Whether to add a trainable bias variable (: true).

- **('crelu' | 'elu' | 'leaky-relu' | 'none' | 'relu' | 'selu' | 'sigmoid' |** (*activation*) – 'softmax' | 'softplus' | 'softsign' | 'swish' | 'tanh'): Activation nonlinearity (: tanh).
- **dropout** (*parameter, 0.0 <= float < 1.0*) – Dropout rate (: 0.0).
- **vars_trainable** (*bool*) – Whether layer variables are trainable (: true).
- **l2_regularization** (*float >= 0.0*) – Scalar controlling L2 regularization (: inherit value of parent module).
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .
- **kwargs** – Additional arguments for Keras LSTM layer, see TensorFlow docs.

class tensorforce.core.layers.**InputGru**(*, *size*, *return_final_state=True*, *bias=False*, *activation=None*, *dropout=0.0*, *vars_trainable=True*, *l2_regularization=None*, *name=None*, *input_spec=None*, ***kwargs*)

Gated recurrent unit layer which is unrolled over a sequence input independently per timestep, and consequently does not maintain an internal state (specification key: `input_gru`).

Parameters

- **size** (*int >= 0*) – Layer output size, 0 implies additionally removing the axis ().
- **return_final_state** (*bool*) – Whether to return the final state instead of the per-step outputs (: true).
- **bias** (*bool*) – Whether to add a trainable bias variable (: true).
- **('crelu' | 'elu' | 'leaky-relu' | 'none' | 'relu' | 'selu' | 'sigmoid' |** (*activation*) – 'softmax' | 'softplus' | 'softsign' | 'swish' | 'tanh'): Activation nonlinearity (: tanh).
- **dropout** (*parameter, 0.0 <= float < 1.0*) – Dropout rate (: 0.0).
- **vars_trainable** (*bool*) – Whether layer variables are trainable (: true).
- **l2_regularization** (*float >= 0.0*) – Scalar controlling L2 regularization (: inherit value of parent module).
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .
- **kwargs** – Additional arguments for Keras GRU layer, see TensorFlow docs.

21.6 Pooling layers

class tensorforce.core.layers.**Flatten**(*, *name=None*, *input_spec=None*)

Flatten layer (specification key: `flatten`).

Parameters

- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .

class tensorforce.core.layers.**Pooling**(*, *reduction*, *name=None*, *input_spec=None*)

Pooling layer (global pooling) (specification key: `pooling`).

Parameters

- **reduction** (*'concat' | 'max' | 'mean' | 'product' | 'sum'*) – Pooling type ().
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .

class tensorforce.core.layers.**Pool1d**(*, *reduction*, *window=2*, *stride=2*, *padding='same'*, *name=None*, *input_spec=None*)

1-dimensional pooling layer (local pooling) (specification key: `pool1d`).

Parameters

- **reduction** (*'average' | 'max'*) – Pooling type ().
- **window** (*int > 0*) – Window size (: 2).
- **stride** (*int > 0*) – Stride size (: 2).
- **padding** (*'same' | 'valid'*) – Padding type, see TensorFlow docs (: 'same').
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .

class tensorforce.core.layers.**Pool2d**(*, *reduction*, *window=2*, *stride=2*, *padding='same'*, *name=None*, *input_spec=None*)

2-dimensional pooling layer (local pooling) (specification key: `pool2d`).

Parameters

- **reduction** (*'average' | 'max'*) – Pooling type ().
- **window** (*int > 0 | (int > 0, int > 0)*) – Window size (: 2).
- **stride** (*int > 0 | (int > 0, int > 0)*) – Stride size (: 2).
- **padding** (*'same' | 'valid'*) – Padding type, see TensorFlow docs (: 'same').
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .

21.7 Normalization layers

class tensorforce.core.layers.**LinearNormalization**(*, *min_value=None*, *max_value=None*, *name=None*, *input_spec=None*)

Linear normalization layer which scales and shifts the input to [-2.0, 2.0], for bounded states with min/max_value (specification key: `linear_normalization`).

Parameters

- **min_value** (*float | array[float]*) – Lower bound of the value (: based on input_spec).
- **max_value** (*float | array[float]*) – Upper bound of the value range (: based on input_spec).
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .

class tensorforce.core.layers.**ExponentialNormalization**(*, *decay*, *axes=None*, *only_mean=False*, *min_variance=0.0001*, *name=None*, *input_spec=None*)

Normalization layer based on the exponential moving average of mean and variance over the temporal sequence of inputs (specification key: `exponential_normalization`).

Parameters

- **decay** (*parameter, 0.0 <= float <= 1.0*) – Decay rate ().
- **axes** (*iter[int >= 0]*) – Normalization axes, excluding batch axis (: all but last input axes).
- **only_mean** (*bool*) – Whether to normalize only with respect to mean, not variance (: false).
- **min_variance** (*float > 0.0*) – Clip variance lower than minimum (: 1e-4).
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .

class tensorforce.core.layers.**InstanceNormalization**(*, *axes=None*, *only_mean=False*, *min_variance=0.0001*, *name=None*, *input_spec=None*)

Instance normalization layer (specification key: `instance_normalization`).

Parameters

- **axes** (*iter[int >= 0]*) – Normalization axes, excluding batch axis (: all input axes).
- **only_mean** (*bool*) – Whether to normalize only with respect to mean, not variance (: false).
- **min_variance** (*float > 0.0*) – Clip variance lower than minimum (: 1e-4).
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .

class tensorforce.core.layers.**BatchNormalization**(*, *axes=None*, *only_mean=False*, *min_variance=0.0001*, *name=None*, *input_spec=None*)

Batch normalization layer, generally should only be used for the agent arguments `reward_processing[return_processing]` and `reward_processing[advantage_processing]` (specification key: `batch_normalization`).

Parameters

- **axes** (*iter[int >= 0]*) – Normalization axes, excluding batch axis (: all but last input axes).
- **only_mean** (*bool*) – Whether to normalize only with respect to mean, not variance (: false).
- **min_variance** (*float > 0.0*) – Clip variance lower than minimum (: 1e-4).
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .

21.8 Misc layers

class tensorforce.core.layers.**Reshape**(*, *shape*, *name=None*, *input_spec=None*)
Reshape layer (specification key: `reshape`).

Parameters

- **shape** () – New shape ().
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .

class tensorforce.core.layers.**Activation**(*, *nonlinearity*, *name=None*, *input_spec=None*)
Activation layer (specification key: `activation`).

Parameters

- **('crelu' | 'elu' | 'leaky-relu' | 'none' | 'relu' | 'selu' | 'sigmoid' |** (*nonlinearity*) – 'softmax' | 'softplus' | 'softsign' | 'swish' | 'tanh'): Nonlinearity ().
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .

class tensorforce.core.layers.**Dropout**(*, *rate*, *name=None*, *input_spec=None*)
Dropout layer (specification key: `dropout`).

Parameters

- **rate** (*parameter, 0.0 <= float < 1.0*) – Dropout rate ().
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .

class tensorforce.core.layers.**Clipping**(*, *lower=None*, *upper=None*, *name=None*, *input_spec=None*)
Clipping layer (specification key: `clipping`).

Parameters

- **lower** (*parameter, float*) – Lower clipping value (: no lower bound).
- **upper** (*parameter, float*) – Upper clipping value (: no upper bound).
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .

class tensorforce.core.layers.**Image**(*, *height=None*, *width=None*, *grayscale=False*, *name=None*, *input_spec=None*)
Image preprocessing layer (specification key: `image`).

Parameters

- **height** (*int*) – Height of resized image (: no resizing or relative to width).
- **width** (*int*) – Width of resized image (: no resizing or relative to height).
- **grayscale** (*bool | iter[float]*) – Turn into grayscale image, optionally using given weights (: false).
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .

class tensorforce.core.layers.**Deltafier**(*, *concatenate=False*, *name=None*, *input_spec=None*)

Deltafier layer computing the difference between the current and the previous input; can only be used as preprocessing layer (specification key: `deltafier`).

Parameters

- **concatenate** (*`False | int >= 0`*) – Whether to concatenate instead of replace deltas with input, and if so, concatenation axis (: false).
- **name** (*`string`*) – Layer name (: internally chosen).
- **input_spec** (*`specification`*) – .

class tensorforce.core.layers.**Sequence**(*, *length*, *axis=-1*, *concatenate=True*, *name=None*, *input_spec=None*)

Sequence layer stacking the current and previous inputs; can only be used as preprocessing layer (specification key: `sequence`).

Parameters

- **length** (*`int > 0`*) – Number of inputs to concatenate ().
- **axis** (*`int >= 0`*) – Concatenation axis, excluding batch axis (: last axis).
- **concatenate** (*`bool`*) – Whether to concatenate inputs at given axis, otherwise introduce new sequence axis (: true).
- **name** (*`string`*) – Layer name (: internally chosen).
- **input_spec** (*`specification`*) – .

21.9 Special layers

class tensorforce.core.layers.**Function**(*function*, *output_spec=None*, *l2_regularization=None*, *name=None*, *input_spec=None*)

Custom TensorFlow function layer (specification key: `function`).

Parameters

- **function** (*`lambda[x -> x]`*) – TensorFlow function ().
- **output_spec** (*`specification`*) – Output tensor specification containing type and/or shape information (: same as input).
- **l2_regularization** (*`float >= 0.0`*) – Scalar controlling L2 regularization (: inherit value of parent module).
- **name** (*`string`*) – Layer name (: internally chosen).
- **input_spec** (*`specification`*) – .

class tensorforce.core.layers.**Register**(*, *tensor*, *name=None*, *input_spec=None*)

Tensor retrieval layer, which is useful when defining more complex network architectures which do not follow the sequential layer-stack pattern, for instance, when handling multiple inputs (specification key: `register`).

Parameters

- **tensor** (*`string`*) – Name under which tensor will be registered ().
- **name** (*`string`*) – Layer name (: internally chosen).
- **input_spec** (*`specification`*) – .

class tensorforce.core.layers.**Retrieve**(*, *tensors*, *aggregation='concat'*, *axis=0*, *name=None*, *input_spec=None*)

Tensor retrieval layer, which is useful when defining more complex network architectures which do not follow the sequential layer-stack pattern, for instance, when handling multiple inputs (specification key: `retrieve`).

Parameters

- **tensors** (*iter[string]*) – Names of tensors to retrieve, either state names or previously registered tensors ().
- **aggregation** (*'concat' | 'product' | 'stack' | 'sum'*) – Aggregation type in case of multiple tensors (: 'concat').
- **axis** (*int >= 0*) – Aggregation axis, excluding batch axis (: 0).
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .

class tensorforce.core.layers.**Block**(*, *layers*, *name=None*, *input_spec=None*)

Block of layers (specification key: `block`).

Parameters

- **layers** (*iter[specification]*) – Layers configuration, see layers ().
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .

class tensorforce.core.layers.**Reuse**(*, *layer*, *name=None*, *input_spec=None*)

Reuse layer (specification key: `reuse`).

Parameters

- **layer** (*string*) – Name of a previously defined layer ().
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .

21.10 Keras layer

class tensorforce.core.layers.**Keras**(*, *layer*, *l2_regularization=None*, *name=None*, *input_spec=None*, ***kwargs*)

Keras layer (specification key: `keras`).

Parameters

- **layer** (*string*) – Keras layer class name, see TensorFlow docs ().
- **l2_regularization** (*float >= 0.0*) – Scalar controlling L2 regularization (: inherit value of parent module).
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .
- **kwargs** – Arguments for the Keras layer, see TensorFlow docs.

CHAPTER 22

Memories

Default memory: `Replay` with default argument `capacity`, so an `int` is a short-form specification of a replay memory with corresponding capacity:

```
Agent.create(
    ...
    memory=10000,
    ...
)
```

class tensorforce.core.memories.**Replay**(*capacity=None*, ***, *device='CPU:0'*, *name=None*, *values_spec=None*, *min_capacity=None*)

Replay memory which randomly retrieves experiences (specification key: `replay`).

Parameters

- **capacity** (*int > 0*) – Memory capacity (: minimum capacity).
- **device** (*string*) – Device name (: CPU:0).
- **name** (*string*) – .
- **values_spec** (*specification*) – .
- **min_capacity** (*int >= 0*) – .

class tensorforce.core.memories.**Recent**(*capacity=None*, ***, *device='CPU:0'*, *name=None*, *values_spec=None*, *min_capacity=None*)

Batching memory which always retrieves most recent experiences (specification key: `recent`).

Parameters

- **capacity** (*int > 0*) – Memory capacity (: minimum capacity).
- **device** (*string*) – Device name (: CPU:0).
- **name** (*string*) – .
- **values_spec** (*specification*) – .
- **min_capacity** (*int >= 0*) – .

CHAPTER 23

Networks

Default network: `LayeredNetwork` with default argument `layers`, so a `list` is a short-form specification of a sequential layer-stack network architecture:

```
Agent.create(
    ...
    policy=dict(network=[
        dict(type='dense', size=64, activation='tanh'),
        dict(type='dense', size=64, activation='tanh')
    ]),
    ...
)
```

Multi-input and other non-sequential networks are specified as nested list of lists of layers, where each of the inner lists forms a sequential component of the overall network architecture. The following example illustrates how to specify such a more complex network, by using the special layers `Register` and `Retrieve` to combine the sequential network components:

```
Agent.create(
    states=dict(
        observation=dict(type='float', shape=(16, 16, 3)),
        attributes=dict(type='int', shape=(4, 2), num_values=5)
    ),
    ...
    policy=[
        [
            dict(type='retrieve', tensors=['observation']),
            dict(type='conv2d', size=32),
            dict(type='flatten'),
            dict(type='register', tensor='obs-embedding')
        ],
        [
            dict(type='retrieve', tensors=['attributes']),
            dict(type='embedding', size=32),
            dict(type='flatten'),
```

(continues on next page)

(continued from previous page)

```
            dict(type='register', tensor='attr-embedding')
        ],
        [
            dict(
                type='retrieve', aggregation='concat',
                tensors=['obs-embedding', 'attr-embedding']
            ),
            dict(type='dense', size=64)
        ]
    ],
    ...
)
```

Note that the final action/value layer of the policy/baseline network is implicitly added, so the network output can be of arbitrary size and use any activation function, and is only required to be a rank-one embedding vector, or optionally have the same shape as the action in the case of a higher-rank action shape.

class tensorforce.core.networks.**AutoNetwork**(*, *size=64*, *depth=2*, *final_size=None*, *final_depth=1*, *rnn=False*, *device=None*, *l2_regularization=None*, *name=None*, *inputs_spec=None*, *internal_rnn=None*)

Network which is automatically configured based on its input tensors, offering high-level customization (specification key: `auto`).

Parameters

- **size** (*int > 0*) – Layer size, before concatenation if multiple states (: 64).
- **depth** (*int > 0*) – Number of layers per state, before concatenation if multiple states (: 2).
- **final_size** (*int > 0*) – Layer size after concatenation if multiple states (: layer size).
- **final_depth** (*int > 0*) – Number of layers after concatenation if multiple states (: 1).
- **rnn** (false | parameter, int >= 0) – Whether to add an LSTM cell with internal state as last layer, and if so, horizon of the LSTM for truncated backpropagation through time (: false).
- **device** (*string*) – Device name (: inherit value of parent module).
- **l2_regularization** (*float >= 0.0*) – Scalar controlling L2 regularization (: inherit value of parent module).
- **name** (*string*) – .
- **inputs_spec** (*specification*) – .

class tensorforce.core.networks.**LayeredNetwork**(*layers*, *, *device=None*, *l2_regularization=None*, *name=None*, *inputs_spec=None*)

Network consisting of Tensorforce layers (specification key: `custom` or `layered`), which can be specified as either a list of layer specifications in the case of a standard sequential layer-stack architecture, or as a list of list of layer specifications in the case of a more complex architecture consisting of multiple sequential layer-stacks. Note that the final action/value layer of the policy/baseline network is implicitly added, so the network output can be of arbitrary size and use any activation function, and is only required to be a rank-one embedding vector, or optionally have the same shape as the action in the case of a higher-rank action shape.

Parameters

- **layers** (*iter[specification] | iter[iter[specification]]*) – Layers configuration, see the layers documentation ().

- **device** (*string*) – Device name (: inherit value of parent module).
- **l2_regularization** (*float >= 0.0*) – Scalar controlling L2 regularization (: inherit value of parent module).
- **name** (*string*) – .
- **inputs_spec** (*specification*) – .

CHAPTER 24

Objectives

class tensorforce.core.objectives.**PolicyGradient**(*, *importance_sampling=False*, *clipping_value=None*, *early_reduce=True*, *name=None*, *states_spec=None*, *internals_spec=None*, *auxiliaries_spec=None*, *actions_spec=None*, *reward_spec=None*)

Policy gradient objective, which maximizes the log-likelihood or likelihood-ratio scaled by the target reward value (specification key: `policy_gradient`).

Parameters

- **importance_sampling** (*bool*) – Whether to use the importance sampling version of the policy gradient objective (: false).
- **clipping_value** (*parameter, float > 0.0*) – Clipping threshold for the maximized value (: no clipping).
- **early_reduce** (*bool*) – Whether to compute objective for aggregated likelihood instead of likelihood per action (: true).
- **name** (*string*) – .
- **states_spec** (*specification*) – .
- **internals_spec** (*specification*) – .
- **auxiliaries_spec** (*specification*) – .
- **actions_spec** (*specification*) – .
- **reward_spec** (*specification*) – .

class tensorforce.core.objectives.**Value**(*, *value*, *huber_loss=None*, *early_reduce=True*, *name=None*, *states_spec=None*, *internals_spec=None*, *auxiliaries_spec=None*, *actions_spec=None*, *reward_spec=None*)

Value approximation objective, which minimizes the L2-distance between the state-(action-)value estimate and the target reward value (specification key: `value`, `state_value`, `action_value`).

Parameters

- **value** (*"state" | "action"*) – Whether to approximate the state- or state-action-value ().
- **huber_loss** (*parameter, float > 0.0*) – Huber loss threshold (: no huber loss).
- **early_reduce** (*bool*) – Whether to compute objective for aggregated value instead of value per action (: true).
- **name** (*string*) – .
- **states_spec** (*specification*) – .
- **internals_spec** (*specification*) – .
- **auxiliaries_spec** (*specification*) – .
- **actions_spec** (*specification*) – .
- **reward_spec** (*specification*) – .

class tensorforce.core.objectives.**DeterministicPolicyGradient**(*, *name=None*, *states_spec=None*, *internals_spec=None*, *auxiliaries_spec=None*, *actions_spec=None*, *reward_spec=None*)

Deterministic policy gradient objective (specification key: `det_policy_gradient`).

Parameters

- **name** (*string*) – .
- **states_spec** (*specification*) – .
- **internals_spec** (*specification*) – .
- **auxiliaries_spec** (*specification*) – .
- **actions_spec** (*specification*) – .
- **reward_spec** (*specification*) – .

class tensorforce.core.objectives.**Plus**(*, *objective1*, *objective2*, *name=None*, *states_spec=None*, *internals_spec=None*, *auxiliaries_spec=None*, *actions_spec=None*, *reward_spec=None*)

Additive combination of two objectives (specification key: `plus`).

Parameters

- **objective1** (*specification*) – First objective configuration ().
- **objective2** (*specification*) – Second objective configuration ().
- **name** (*string*) – .
- **states_spec** (*specification*) – .

- **internals_spec**(*specification*) – .
- **auxiliaries_spec**(*specification*) – .
- **actions_spec**(*specification*) – .
- **reward_spec**(*specification*) – .

CHAPTER 25

Optimizers

Default optimizer: `OptimizerWrapper` which offers additional update modifier options, so instead of using `TFOptimizer` directly, a customized Adam optimizer can be specified via:

```
Agent.create(
    ...
    optimizer=dict(
        optimizer='adam', learning_rate=1e-3, clipping_threshold=1e-2,
        multi_step=10, subsampling_fraction=64, linesearch_iterations=5,
        doublecheck_update=True
    ),
    ...
)
```

class tensorforce.core.optimizers.**OptimizerWrapper**(*optimizer*, *, *learning_rate=0.001*, *clipping_threshold=None*, *multi_step=1*, *subsampling_fraction=1.0*, *linesearch_iterations=0*, *doublecheck_update=False*, *name=None*, *arguments_spec=None*, *optimizing_iterations=None*, ***kwargs*)

Optimizer wrapper, which performs additional update modifications, argument order indicates modifier nesting from outside to inside (specification key: `optimizer_wrapper`).

Parameters

- **optimizer** (*specification*) – Optimizer ().
- **learning_rate** (*parameter, float > 0.0*) – Learning rate (: 1e-3).
- **clipping_threshold** (*parameter, float > 0.0*) – Clipping threshold (: no clipping).
- **multi_step** (*parameter, int >= 1*) – Number of optimization steps (: single step).

- **subsampling_fraction** (*parameter, int > 0 | 0.0 < float <= 1.0*) – Absolute/relative fraction of batch timesteps to subsample (: no subsampling).
- **linesearch_iterations** (*parameter, int >= 0*) – Maximum number of line search iterations, using a backtracking factor of 0.75 (: no line search).
- **doublecheck_update** (*bool*) – Check whether update has decreased loss and otherwise reverse it
- **name** (*string*) – ().
- **arguments_spec** (*specification*) – .

class tensorforce.core.optimizers.**TFOptimizer**(*, *optimizer*, *learning_rate*, *gradient_norm_clipping=None*, *name=None*, *arguments_spec=None*, ***kwargs*)

TensorFlow optimizer (specification key: `tf_optimizer`, `adadelta`, `adagrad`, `adam`, `adamax`, `adamw`, `ftrl`, `lazyadam`, `nadam`, `radam`, `ranger`, `rmsprop`, `sgd`, `sgdw`)

Parameters

- **optimizer** (`adadelta` | `adagrad` | `adam` | `adamax` | `adamw` | `ftrl` | `lazyadam` | `nadam` | `radam` | `ranger` | `rmsprop` | `sgd` | `sgdw`) – TensorFlow optimizer name, see TensorFlow docs and TensorFlow Addons docs (unless given by specification key).
- **learning_rate** (*parameter, float > 0.0*) – Learning rate ().
- **gradient_norm_clipping** (*parameter, float > 0.0*) – Clip gradients by the ratio of the sum of their norms (: 1.0).
- **name** (*string*) – ().
- **arguments_spec** (*specification*) – .
- **kwargs** – Arguments for the TensorFlow optimizer, special values "decoupled_weight_decay", "lookahead" and "moving_average", see TensorFlow docs and TensorFlow Addons docs.

class tensorforce.core.optimizers.**NaturalGradient**(*, *learning_rate*, *cg_max_iterations=10*, *cg_damping=0.1*, *only_positive_updates=True*, *name=None*, *arguments_spec=None*)

Natural gradient optimizer (specification key: `natural_gradient`).

Parameters

- **learning_rate** (*parameter, float > 0.0*) – Learning rate as KL-divergence of distributions between optimization steps ().
- **cg_max_iterations** (*int >= 1*) – Maximum number of conjugate gradient iterations. (: 10).
- **cg_damping** (*0.0 <= float <= 1.0*) – Conjugate gradient damping factor. (: 0.1).
- **only_positive_updates** (*bool*) – Whether to only perform updates with positive improvement estimate (: true).
- **name** (*string*) – ().
- **arguments_spec** (*specification*) – .

class tensorforce.core.optimizers.**Evolutionary**(*, *learning_rate*, *num_samples=1*, *name=None*, *arguments_spec=None*)

Evolutionary optimizer, which samples random perturbations and applies them either as positive or negative update depending on their improvement of the loss (specification key: `evolutionary`).

Parameters

- **learning_rate** (*parameter, float > 0.0*) – Learning rate ().
- **num_samples** (*parameter, int >= 1*) – Number of sampled perturbations (: 1).
- **name** (*string*) – ().
- **arguments_spec** (*specification*) – .

class tensorforce.core.optimizers.**ClippingStep**(*, *optimizer*, *threshold*, *mode='global_norm'*, *name=None*, *arguments_spec=None*)

Clipping-step update modifier, which clips the updates of the given optimizer (specification key: `clipping_step`).

Parameters

- **optimizer** (*specification*) – Optimizer configuration ().
- **threshold** (*parameter, float > 0.0*) – Clipping threshold ().
- **mode** (*'global_norm' | 'norm' | 'value'*) – Clipping mode (: 'global_norm').
- **name** (*string*) – ().
- **arguments_spec** (*specification*) – .

class tensorforce.core.optimizers.**MultiStep**(*, *optimizer*, *num_steps*, *name=None*, *arguments_spec=None*)

Multi-step update modifier, which applies the given optimizer for a number of times (specification key: `multi_step`).

Parameters

- **optimizer** (*specification*) – Optimizer configuration ().
- **num_steps** (*parameter, int >= 1*) – Number of optimization steps ().
- **name** (*string*) – ().
- **arguments_spec** (*specification*) – .

class tensorforce.core.optimizers.**DoublecheckStep**(*, *optimizer*, *name=None*, *arguments_spec=None*)

Double-check update modifier, which checks whether the update of the given optimizer has decreased the loss and otherwise reverses it (specification key: `doublecheck_step`).

Parameters

- **optimizer** (*specification*) – Optimizer configuration ().
- **name** (*string*) – ().
- **arguments_spec** (*specification*) – .

class tensorforce.core.optimizers.**LinesearchStep**(*, *optimizer*, *max_iterations*, *backtracking_factor=0.75*, *name=None*, *arguments_spec=None*)

Line-search-step update modifier, which performs a line search on the update step returned by the given optimizer to find a potentially superior smaller step size (specification key: `linesearch_step`).

Parameters

- **optimizer** (*specification*) – Optimizer configuration ().
- **max_iterations** (*parameter, int >= 1*) – Maximum number of line search iterations ().
- **backtracking_factor** (*parameter, 0.0 < float < 1.0*) – Line search backtracking factor (: 0.75).
- **name** (*string*) – ().
- **arguments_spec** (*specification*) – .

class tensorforce.core.optimizers.**SubsamplingStep**(*, *optimizer*, *fraction*, *name=None*, *arguments_spec=None*)

Subsampling-step update modifier, which randomly samples a subset of batch instances before applying the given optimizer (specification key: `subsampling_step`).

Parameters

- **optimizer** (*specification*) – Optimizer configuration ().
- **fraction** (*parameter, int > 0 | 0.0 < float <= 1.0*) – Absolute/relative fraction of batch timesteps to subsample ().
- **name** (*string*) – ().
- **arguments_spec** (*specification*) – .

class tensorforce.core.optimizers.**Synchronization**(*, *update_weight*, *sync_frequency=None*, *name=None*, *arguments_spec=None*)

Synchronization optimizer, which updates variables periodically to the value of a corresponding set of source variables (specification key: `synchronization`).

Parameters

- **optimizer** (*specification*) – Optimizer configuration ().
- **update_weight** (*parameter, 0.0 < float <= 1.0*) – Update weight ().
- **sync_frequency** (*parameter, int >= 1*) – Interval between updates which also perform a synchronization step (: every update).
- **name** (*string*) – ().
- **arguments_spec** (*specification*) – .

class tensorforce.core.optimizers.**Plus**(*, *optimizer1*, *optimizer2*, *name=None*, *arguments_spec=None*)

Additive combination of two optimizers (specification key: `plus`).

Parameters

- **optimizer1** (*specification*) – First optimizer configuration ().
- **optimizer2** (*specification*) – Second optimizer configuration ().
- **name** (*string*) – ().
- **arguments_spec** (*specification*) – .

CHAPTER 26

Parameters

Tensorforce distinguishes between agent/module arguments (primitive types: bool/int/float) which either specify part of the TensorFlow model architecture, like the layer size, or a value within the architecture, like the learning rate. Whereas the former are statically defined as part of the agent initialization, the latter can be dynamically adjusted afterwards. These dynamic hyperparameter are indicated by `parameter` as part of their argument type specification in the documentation, and can alternatively be assigned a parameter module instead of a constant value, for instance, to specify a decaying learning rate.

Default parameter: `Constant`, so a `bool`/`int`/`float` value is a short-form specification of a constant (dynamic) parameter:

```
Agent.create(
    ...
    exploration=0.1,
    ...
)
```

Example of how to specify an exponentially decaying learning rate:

```
Agent.create(
    ...
    optimizer=dict(optimizer='adam', learning_rate=dict(
        type='decaying', decay='exponential', unit='timesteps',
        num_steps=1000, initial_value=0.01, decay_rate=0.5
    )),
    ...
)
```

Example of how to specify a linearly increasing reward horizon:

```
Agent.create(
    ...
    reward_estimation=dict(horizon=dict(
        type='linear', unit='episodes', num_steps=1000,
        initial_value=10, final_value=50
```

(continues on next page)

(continued from previous page)

```
    )),
    ...
)
```

class tensorforce.core.parameters.**Constant**(*value*, *, *name=None*, *dtype=None*, *min_value=None*, *max_value=None*)

Constant hyperparameter (specification key: `constant`).

Parameters

- **value** (*float | int | bool*) – Constant hyperparameter value ().
- **name** (*string*) – .
- **dtype** (*type*) – .
- **min_value** (*dtype-compatible value*) – .
- **max_value** (*dtype-compatible value*) – .

class tensorforce.core.parameters.**Linear**(*, *unit*, *num_steps*, *initial_value*, *final_value*, *name=None*, *dtype=None*, *min_value=None*, *max_value=None*)

Linear hyperparameter (specification key: `linear`).

Parameters

- **unit** (*"timesteps" | "episodes" | "updates"*) – Unit of decay schedule ().
- **num_steps** (*int*) – Number of decay steps ().
- **initial_value** (*float*) – Initial value ().
- **final_value** (*float*) – Final value ().
- **name** (*string*) – .
- **dtype** (*type*) – .
- **min_value** (*dtype-compatible value*) – .
- **max_value** (*dtype-compatible value*) – .

class tensorforce.core.parameters.**PiecewiseConstant**(*, *unit*, *boundaries*, *values*, *name=None*, *dtype=None*, *min_value=None*, *max_value=None*)

Piecewise-constant hyperparameter (specification key: `piecewise_constant`).

Parameters

- **unit** (*"timesteps" | "episodes" | "updates"*) – Unit of interval boundaries ().
- **boundaries** (*iter[long]*) – Strictly increasing interval boundaries for constant segments ().
- **values** (*iter[dtype-dependent]*) – Interval values of constant segments, one more than ().
- **name** (*string*) – .
- **dtype** (*type*) – .
- **min_value** (*dtype-compatible value*) – .

- **max_value** (*dtype-compatible value*) – .

class tensorforce.core.parameters.**Decaying**(*, *decay*, *unit*, *num_steps*, *initial_value*, *increasing=False*, *inverse=False*, *scale=1.0*, *name=None*, *dtype=None*, *min_value=None*, *max_value=None*, ***kwargs*)

Decaying hyperparameter (specification key: `decaying`, `exponential`, `polynomial`, `inverse_time`, `cosine`, `cosine_restarts`, `linear_cosine`, `linear_cosine_noisy`).

Parameters

- **decay** (*"linear" | "exponential" | "polynomial" | "inverse_time" | "cosine" | "cosine_restarts" | "linear_cosine" | "linear_cosine_noisy"*) – Decay type, see also TensorFlow docs ().
- **unit** (*"timesteps" | "episodes" | "updates"*) – Unit of decay schedule ().
- **num_steps** (*int*) – Number of decay steps ().
- **initial_value** (*float | int*) – Initial value ().
- **increasing** (*bool*) – Whether to subtract the decayed value from 1.0 (: false).
- **inverse** (*bool*) – Whether to take the inverse of the decayed value (: false).
- **scale** (*float*) – Scaling factor for (inverse) decayed value (: 1.0).
- **kwargs** – Additional arguments depend on decay mechanism. Linear decay:

  Exponential decay:

  Polynomial decay:

  Inverse time decay:

  Cosine decay:

  Cosine decay with restarts:

  Linear cosine decay:

  Noisy linear cosine decay:
- **name** (*string*) – .
- **dtype** (*type*) – .
- **min_value** (*dtype-compatible value*) – .
- **max_value** (*dtype-compatible value*) – .

class tensorforce.core.parameters.**OrnsteinUhlenbeck**(*, *theta=0.15*, *sigma=0.3*, *mu=0.0*, *absolute=False*, *name=None*, *dtype=None*, *min_value=None*, *max_value=None*)

Ornstein-Uhlenbeck process (specification key: `ornstein_uhlenbeck`).

Parameters

- **theta** (*float > 0.0*) – Theta value (: 0.15).
- **sigma** (*float > 0.0*) – Sigma value (: 0.3).
- **mu** (*float*) – Mu value (: 0.0).
- **absolute** (*bool*) – Absolute value (: false).

- **name** (*string*) – .
- **dtype** (*type*) – .
- **min_value** (*dtype-compatible value*) – .
- **max_value** (*dtype-compatible value*) – .

class tensorforce.core.parameters.**Random**(*, *distribution*, *name=None*, *dtype=None*, *shape=()*, *min_value=None*, *max_value=None*, ***kwargs*)

Random hyperparameter (specification key: `random`).

Parameters

- **distribution** (*"normal" | "uniform"*) – Distribution type for random hyperparameter value ().
- **kwargs** – Additional arguments dependent on distribution type. Normal distribution:

  Uniform distribution:

- **name** (*string*) – .
- **dtype** (*type*) – .
- **shape** (*iter[int > 0]*) – .
- **min_value** (*dtype-compatible value*) – .
- **max_value** (*dtype-compatible value*) – .

CHAPTER 27

Policies

Default policy: depends on agent configuration, but always with default argument `network` (with default argument `layers`), so a `list` is a short-form specification of a sequential layer-stack network architecture:

```
Agent.create(
    ...
    policy=[
        dict(type='dense', size=64, activation='tanh'),
        dict(type='dense', size=64, activation='tanh')
    ],
    ...
)
```

Or simply:

```
Agent.create(
    ...
    policy=dict(network='auto'),
    ...
)
```

See the networks documentation for more information about how to specify a network.

Example of a full parametrized-distributions policy specification with customized distribution and decaying temperature:

```
Agent.create(
    ...
    policy=dict(
        type='parametrized_distributions',
        network=[
            dict(type='dense', size=64, activation='tanh'),
            dict(type='dense', size=64, activation='tanh')
        ],
        distributions=dict(
```

(continues on next page)

(continued from previous page)

```
            float=dict(type='gaussian', global_stddev=True),
            bounded_action=dict(type='beta')
        ),
        temperature=dict(
            type='decaying', decay='exponential', unit='episodes',
            num_steps=100, initial_value=0.01, decay_rate=0.5
        )
    )
    ...
)
```

class tensorforce.core.policies.**ParametrizedActionValue**(*network='auto'*, ***, *device=None*, *l2_regularization=None*, *name=None*, *states_spec=None*, *auxiliaries_spec=None*, *internals_spec=None*, *actions_spec=None*)

Policy which parametrizes an action-value function, conditioned on the output of a neural network processing the input state (specification key: `parametrized_action_value`).

Parameters

- **network** (*'auto' | specification*) – Policy network configuration, see networks (: 'auto', automatically configured network).
- **device** (*string*) – Device name (: inherit value of parent module).
- **l2_regularization** (*float >= 0.0*) – Scalar controlling L2 regularization (: inherit value of parent module).
- **name** (*string*) – .
- **states_spec** (*specification*) – .
- **auxiliaries_spec** (*specification*) – .
- **internals_spec** (*specification*) – .
- **actions_spec** (*specification*) – .

class tensorforce.core.policies.**ParametrizedDistributions**(*network='auto'*, ***, *distributions=None*, *temperature=1.0*, *use_beta_distribution=False*, *device=None*, *l2_regularization=None*, *name=None*, *states_spec=None*, *auxiliaries_spec=None*, *internals_spec=None*, *actions_spec=None*)

Policy which parametrizes independent distributions per action, conditioned on the output of a central neural network processing the input state, supporting both a stochastic and value-based policy interface (specification key: `parametrized_distributions`).

Parameters

- **network** (*'auto' | specification*) – Policy network configuration, see networks (: 'auto', automatically configured network).
- **distributions** (*dict[specification]*) – Distributions configuration, see distributions, specified per action-type or -name (: per action-type, Bernoulli distribution for binary boolean actions, categorical distribution for discrete integer actions, Gaussian distribution for unbounded continuous actions, Beta distribution for bounded continuous actions).
- **temperature** (*parameter | dict[parameter], float >= 0.0*) – Sampling temperature, global or per action (: 1.0).
- **use_beta_distribution** (*bool*) – Whether to use the Beta distribution for bounded continuous actions by default. (: false).
- **device** (*string*) – Device name (: inherit value of parent module).
- **l2_regularization** (*float >= 0.0*) – Scalar controlling L2 regularization (: inherit value of parent module).
- **name** (*string*) – .
- **states_spec** (*specification*) – .
- **auxiliaries_spec** (*specification*) – .
- **internals_spec** (*specification*) – .
- **actions_spec** (*specification*) – .

class tensorforce.core.policies.**ParametrizedStateValue**(*network='auto'*, ***, *device=None*, *l2_regularization=None*, *name=None*, *states_spec=None*, *auxiliaries_spec=None*, *internals_spec=None*, *actions_spec=None*)

Policy which parametrizes a state-value function, conditioned on the output of a neural network processing the input state (specification key: `parametrized_state_value`).

Parameters

- **network** (*'auto' | specification*) – Policy network configuration, see networks (: 'auto', automatically configured network).
- **device** (*string*) – Device name (: inherit value of parent module).
- **l2_regularization** (*float >= 0.0*) – Scalar controlling L2 regularization (: inherit value of parent module).
- **name** (*string*) – .
- **states_spec** (*specification*) – .
- **auxiliaries_spec** (*specification*) – .
- **internals_spec** (*specification*) – .
- **actions_spec** (*specification*) – .

class tensorforce.core.policies.**ParametrizedValuePolicy**(*network='auto'*, ***, *state_value_mode='separate'*, *device=None*, *l2_regularization=None*, *name=None*, *states_spec=None*, *auxiliaries_spec=None*, *internals_spec=None*, *actions_spec=None*)

Policy which parametrizes independent action-/advantage-/state-value functions per action and optionally a state-value function, conditioned on the output of a central neural network processing the input state (specification key: `parametrized_value_policy`).

Parameters

- **network** (*'auto' | specification*) – Policy network configuration, see networks (: 'auto', automatically configured network).
- **state_value_mode** (*'implicit' | 'separate' | 'separate-per-action'*) – Whether to compute the state value implicitly as maximum action value (like DQN), or as either a single separate state-value function or a function per action (like DuelingDQN) (: single separate state-value function).
- **device** (*string*) – Device name (: inherit value of parent module).
- **l2_regularization** (*float >= 0.0*) – Scalar controlling L2 regularization (: inherit value of parent module).
- **name** (*string*) – .
- **states_spec** (*specification*) – .
- **auxiliaries_spec** (*specification*) – .
- **internals_spec** (*specification*) – .
- **actions_spec** (*specification*) – .

CHAPTER 28

Preprocessing

Example of how to specify state and reward preprocessing:

```
Agent.create(
    ...
    state_preprocessing=[
        dict(type='image', height=4, width=4, grayscale=True),
        dict(type='exponential_normalization', decay=0.999)
    ],
    reward_preprocessing=dict(type='clipping', lower=-1.0, upper=1.0),
    ...
)
```

class tensorforce.core.layers.**Clipping**(*, *lower=None*, *upper=None*, *name=None*, *input_spec=None*)

Clipping layer (specification key: `clipping`).

Parameters

- **lower** (*parameter, float*) – Lower clipping value (: no lower bound).
- **upper** (*parameter, float*) – Upper clipping value (: no upper bound).
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .

class tensorforce.core.layers.**Image**(*, *height=None*, *width=None*, *grayscale=False*, *name=None*, *input_spec=None*)

Image preprocessing layer (specification key: `image`).

Parameters

- **height** (*int*) – Height of resized image (: no resizing or relative to width).
- **width** (*int*) – Width of resized image (: no resizing or relative to height).
- **grayscale** (*bool | iter[float]*) – Turn into grayscale image, optionally using given weights (: false).

- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .

class tensorforce.core.layers.**LinearNormalization**(*, *min_value=None*, *max_value=None*, *name=None*, *input_spec=None*)

Linear normalization layer which scales and shifts the input to [-2.0, 2.0], for bounded states with min/max_value (specification key: `linear_normalization`).

Parameters

- **min_value** (*float | array[float]*) – Lower bound of the value (: based on input_spec).
- **max_value** (*float | array[float]*) – Upper bound of the value range (: based on input_spec).
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .

class tensorforce.core.layers.**ExponentialNormalization**(*, *decay*, *axes=None*, *only_mean=False*, *min_variance=0.0001*, *name=None*, *input_spec=None*)

Normalization layer based on the exponential moving average of mean and variance over the temporal sequence of inputs (specification key: `exponential_normalization`).

Parameters

- **decay** (*parameter, 0.0 <= float <= 1.0*) – Decay rate ().
- **axes** (*iter[int >= 0]*) – Normalization axes, excluding batch axis (: all but last input axes).
- **only_mean** (*bool*) – Whether to normalize only with respect to mean, not variance (: false).
- **min_variance** (*float > 0.0*) – Clip variance lower than minimum (: 1e-4).
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .

class tensorforce.core.layers.**InstanceNormalization**(*, *axes=None*, *only_mean=False*, *min_variance=0.0001*, *name=None*, *input_spec=None*)

Instance normalization layer (specification key: `instance_normalization`).

Parameters

- **axes** (*iter[int >= 0]*) – Normalization axes, excluding batch axis (: all input axes).
- **only_mean** (*bool*) – Whether to normalize only with respect to mean, not variance (: false).
- **min_variance** (*float > 0.0*) – Clip variance lower than minimum (: 1e-4).
- **name** (*string*) – Layer name (: internally chosen).
- **input_spec** (*specification*) – .

class tensorforce.core.layers.**Deltafier**(*, *concatenate=False*, *name=None*, *input_spec=None*)

Deltafier layer computing the difference between the current and the previous input; can only be used as preprocessing layer (specification key: `deltafier`).

Parameters

- **concatenate** (*`False | int >= 0`*) – Whether to concatenate instead of replace deltas with input, and if so, concatenation axis (: false).
- **name** (*`string`*) – Layer name (: internally chosen).
- **input_spec** (*`specification`*) – .

class tensorforce.core.layers.**Sequence**(*, *length*, *axis=-1*, *concatenate=True*, *name=None*, *input_spec=None*)

Sequence layer stacking the current and previous inputs; can only be used as preprocessing layer (specification key: `sequence`).

Parameters

- **length** (*`int > 0`*) – Number of inputs to concatenate ().
- **axis** (*`int >= 0`*) – Concatenation axis, excluding batch axis (: last axis).
- **concatenate** (*`bool`*) – Whether to concatenate inputs at given axis, otherwise introduce new sequence axis (: true).
- **name** (*`string`*) – Layer name (: internally chosen).
- **input_spec** (*`specification`*) – .

class tensorforce.core.layers.**Activation**(*, *nonlinearity*, *name=None*, *input_spec=None*)

Activation layer (specification key: `activation`).

Parameters

- **('crelu' | 'elu' | 'leaky-relu' | 'none' | 'relu' | 'selu' | 'sigmoid' |** (*`nonlinearity`*) – 'softmax' | 'softplus' | 'softsign' | 'swish' | 'tanh'): Nonlinearity ().
- **name** (*`string`*) – Layer name (: internally chosen).
- **input_spec** (*`specification`*) – .

class tensorforce.core.layers.**Dropout**(*, *rate*, *name=None*, *input_spec=None*)

Dropout layer (specification key: `dropout`).

Parameters

- **rate** (*`parameter, 0.0 <= float < 1.0`*) – Dropout rate ().
- **name** (*`string`*) – Layer name (: internally chosen).
- **input_spec** (*`specification`*) – .

CHAPTER 29

Runner utility

class tensorforce.execution.**Runner**(*agent*, *environment=None*, *max_episode_timesteps=None*, *evaluation=False*, *num_parallel=None*, *environments=None*, *remote=None*, *blocking=False*, *host=None*, *port=None*)

Tensorforce runner utility.

Parameters

- **agent** (*specification | Agent object*) – Agent specification or object, the latter is not closed automatically as part of `runner.close()`, parallel_interactions is implicitly specified as / expected to be at least num_parallel, -1 if evaluation ().
- **environment** (*specification | Environment object*) – Environment specification or object, the latter is not closed automatically as part of `runner.close()` (, or alternatively `environments`, invalid for "socket-client" remote mode).
- **max_episode_timesteps** (*int > 0*) – Maximum number of timesteps per episode, overwrites the environment default if defined (: environment default, invalid for "socket-client" remote mode).
- **evaluation** (*bool*) – Whether to run the (last if multiple) environment in evaluation mode (: no evaluation).
- **num_parallel** (*int > 0*) – Number of environment instances to execute in parallel (: no parallel execution, implicitly specified by `environments`).
- **environments** (*list[specification | Environment object]*) – Environment specifications or objects to execute in parallel, the latter are not closed automatically as part of `runner.close()` (: no parallel execution, alternatively specified via `environment` and `num_parallel`, invalid for "socket-client" remote mode).
- **remote** (*"multiprocessing" | "socket-client"*) – Communication mode for remote environment execution of parallelized environment execution, not compatible with environment(s) given as Environment objects, "socket-client" mode requires a corresponding "socket-server" running (: local execution).

- **blocking** (*bool*) – Whether remote environment calls should be blocking, only valid if remote mode given (: not blocking, invalid unless "multiprocessing" or "socket-client" remote mode).
- **host** (*str, iter[str]*) – Socket server hostname(s) or IP address(es) (only for "socket-client" remote mode).
- **port** (*int, iter[int]*) – Socket server port(s), increasing sequence if single host and port given (only for "socket-client" remote mode).

run(*num_episodes=None*, *num_timesteps=None*, *num_updates=None*, *batch_agent_calls=False*, *sync_timesteps=False*, *sync_episodes=False*, *num_sleep_secs=0.001*, *callback=None*, *callback_episode_frequency=None*, *callback_timestep_frequency=None*, *use_tqdm=True*, *mean_horizon=1*, *evaluation=False*, *save_best_agent=None*, *evaluation_callback=None*)
Run experiment.

Parameters

- **num_episodes** (*int > 0*) – Number of episodes to run experiment (: no episode limit).
- **num_timesteps** (*int > 0*) – Number of timesteps to run experiment (: no timestep limit).
- **num_updates** (*int > 0*) – Number of agent updates to run experiment (: no update limit).
- **batch_agent_calls** (*bool*) – Whether to batch agent calls for parallel environment execution (: false, separate call per environment).
- **sync_timesteps** (*bool*) – Whether to synchronize parallel environment execution on timestep-level, implied by batch_agent_calls (: false, unless batch_agent_calls is true).
- **sync_episodes** (*bool*) – Whether to synchronize parallel environment execution on episode-level (: false).
- **num_sleep_secs** (*float*) – Sleep duration if no environment is ready (: one millisecond).
- **callback** (*(Runner, parallel) -> bool*) – Callback function taking the runner instance plus parallel index and returning a boolean value indicating whether execution should continue (: callback always true).
- **callback_episode_frequency** (*int*) – Episode interval between callbacks (: every episode).
- **callback_timestep_frequency** (*int*) – Timestep interval between callbacks (: not specified).
- **use_tqdm** (*bool*) – Whether to display a tqdm progress bar for the experiment run (: true), with the following additional information (averaged over number of episodes given via mean_horizon):
- **mean_horizon** (*int*) – Number of episodes progress bar values and evaluation score are averaged over (: not averaged).
- **evaluation** (*bool*) – Whether to run in evaluation mode, only valid if a single environment (: no evaluation).
- **save_best_agent** (*string*) – Directory to save the best version of the agent according to the evaluation score (: best agent is not saved).

- **evaluation_callback** (*int | Runner -> float*) – Callback function taking the runner instance and returning an evaluation score (: cumulative evaluation reward averaged over mean_horizon episodes).

CHAPTER 30

General environment interface

30.1 Initialization and termination

static Environment.**create**(*environment=None*, *max_episode_timesteps=None*, *remote=None*, *blocking=False*, *host=None*, *port=None*, ***kwargs*)

Creates an environment from a specification. In case of "socket-server" remote mode, runs environment in server communication loop until closed.

Parameters

- **environment** (*specification | Environment class/object*) – JSON file, specification key, configuration dictionary, library module, Environment class/object, or gym.Env ().
- **max_episode_timesteps** (*int > 0*) – Maximum number of timesteps per episode, overwrites the environment default if defined (: environment default, invalid for "socket-client" remote mode).
- **remote** (*"multiprocessing" | "socket-client" | "socket-server"*) – Communication mode for remote environment execution of parallelized environment execution, "socket-client" mode requires a corresponding "socket-server" running, and "socket-server" mode runs environment in server communication loop until closed (: local execution).
- **blocking** (*bool*) – Whether remote environment calls should be blocking (: not blocking, invalid unless "multiprocessing" or "socket-client" remote mode).
- **host** (*str*) – Socket server hostname or IP address (only for "socket-client" remote mode).
- **port** (*int*) – Socket server port (only for "socket-client/server" remote mode).
- **kwargs** – Additional arguments.

Environment.**close**()

Closes the environment.

30.2 Properties

`Environment.`**`states`**`()`
Returns the state space specification.

Returns Arbitrarily nested dictionary of state descriptions with the following attributes:

Return type specification

`Environment.`**`actions`**`()`
Returns the action space specification.

Returns Arbitrarily nested dictionary of action descriptions with the following attributes:

Return type specification

`Environment.`**`max_episode_timesteps`**`()`
Returns the maximum number of timesteps per episode.

Returns Maximum number of timesteps per episode.

Return type int

30.3 Interaction functions

`Environment.`**`reset`**`()`
Resets the environment to start a new episode.

Returns Dictionary containing initial state(s) and auxiliary information.

Return type dict[state]

`Environment.`**`execute`**`(`*`actions`*`)`
Executes the given action(s) and advances the environment by one step.

Parameters **`actions`** (*`dict[action]`*) – Dictionary containing action(s) to be executed ().

Returns Dictionary containing next state(s), whether a terminal state is reached or 2 if the episode was aborted, and observed reward.

Return type dict[state], bool | 0 | 1 | 2, float

CHAPTER 31

OpenAI Gym

class tensorforce.environments.**OpenAIGym**(*level*, *visualize=False*, *import_modules=None*, *min_value=None*, *max_value=None*, *terminal_reward=0.0*, *reward_threshold=None*, *drop_states_indices=None*, *visualize_directory=None*, ***kwargs*)

OpenAI Gym environment adapter (specification key: `gym`, `openai_gym`).

May require:

```
pip3 install gym
pip3 install gym[all]
```

Parameters

- **level** (*string | gym.Env*) – Gym id or instance ().
- **visualize** (*bool*) – Whether to visualize interaction (: false).
- **min_value** (*float*) – Lower bound clipping for otherwise unbounded state values (: no clipping).
- **max_value** (*float*) – Upper bound clipping for otherwise unbounded state values (: no clipping).
- **terminal_reward** (*float*) – Additional reward for early termination, if otherwise indistinguishable from termination due to maximum number of timesteps (: Gym default).
- **reward_threshold** (*float*) – Gym environment argument, the reward threshold before the task is considered solved (: Gym default).
- **drop_states_indices** (*list[int]*) – Drop states indices (: none).
- **visualize_directory** (*string*) – Visualization output directory (: none).
- **kwargs** – Additional Gym environment arguments.

CHAPTER 32

Arcade Learning Environment

class tensorforce.environments.**ArcadeLearningEnvironment**(*level*, *life_loss_terminal=False*, *life_loss_punishment=0.0*, *repeat_action_probability=0.0*, *visualize=False*, *frame_skip=1*, *seed=None*)

Arcade Learning Environment adapter (specification key: `ale`, `arcade_learning_environment`).

May require:

```
sudo apt-get install libsdl1.2-dev libsdl-gfx1.2-dev libsdl-image1.2-dev cmake
```

Parameters

- **level** (*string*) – ALE rom file ().
- **loss_of_life_termination** – Signals a terminal state on loss of life (: false).
- **loss_of_life_reward** (*float*) – Reward/Penalty on loss of life (negative values are a penalty) (: 0.0).
- **repeat_action_probability** (*float*) – Repeats last action with given probability (: 0.0).
- **visualize** (*bool*) – Whether to visualize interaction (: false).
- **frame_skip** (*int > 0*) – Number of times to repeat an action without observing (: 1).
- **seed** (*int*) – Random seed (: none).

CHAPTER 33

OpenAI Retro

class tensorforce.environments.**OpenAIRetro**(*level*, *visualize=False*, *visualize_directory=None*, ***kwargs*)

OpenAI Retro environment adapter (specification key: `retro`, `openai_retro`).

May require:

```
pip3 install gym-retro
```

Parameters

- **level** (*string*) – Game id ().
- **visualize** (*bool*) – Whether to visualize interaction (: false).
- **monitor_directory** (*string*) – Monitor output directory (: none).
- **kwargs** – Additional Retro environment arguments.

CHAPTER 34

Open Sim

class tensorforce.environments.**OpenSim**(*level*, *visualize=False*, ***kwargs*)

OpenSim environment adapter (specification key: `osim`, `open_sim`).

Parameters

- **level** (*'Arm2D' | 'L2Run' | 'Prosthetics'*) – Environment id ().
- **visualize** (*bool*) – Whether to visualize interaction (: false).
- **integrator_accuracy** (*float*) – Integrator accuracy (: 5e-5).

CHAPTER 35

PyGame Learning Environment

class tensorforce.environments.**PyGameLearningEnvironment**(*level*, *visualize=False*, *frame_skip=1*, *fps=30*)

PyGame Learning Environment environment adapter (specification key: `ple`, `pygame_learning_environment`).

May require:

```
sudo apt-get install git python3-dev python3-setuptools python3-numpy python3-
↪opengl     libsdl-image1.2-dev libsdl-mixer1.2-dev libsdl-ttf2.0-dev libsmpeg-
↪dev libsdl1.2-dev     libportmidi-dev libswscale-dev libavformat-dev libavcodec-
↪dev libtiff5-dev libx11-6     libx11-dev fluid-soundfont-gm timgm6mb-soundfont␣
↪xfonts-base xfonts-100dpi xfonts-75dpi     xfonts-cyrillic fontconfig fonts-
↪freefont-ttf libfreetype6-dev

pip3 install pygame
pip3 install git+https://github.com/ntasfi/PyGame-Learning-Environment.git
```

Parameters

- **level** (string | subclass of `ple.games.base`) – Game instance or name of class in `ple.games`, like "Catcher", "Doom", "FlappyBird", "MonsterKong", "Pixelcopter", "Pong", "PuckWorld", "RaycastMaze", "Snake", "WaterWorld" ().
- **visualize** (*bool*) – Whether to visualize interaction (: false).
- **frame_skip** (*int > 0*) – Number of times to repeat an action without observing (: 1).
- **fps** (*int > 0*) – The desired frames per second we want to run our game at (: 30).

CHAPTER 36

ViZDoom

class tensorforce.environments.**ViZDoom**(*level*, *visualize=False*, *include_variables=False*, *factored_action=False*, *frame_skip=12*, *seed=None*)

ViZDoom environment adapter (specification key: `vizdoom`).

May require:

```
sudo apt-get install g++ build-essential libsdl2-dev zlib1g-dev libmpg123-dev␣
↪libjpeg-dev     libsndfile1-dev nasm tar libbz2-dev libgtk2.0-dev make cmake␣
↪git chrpath timidity     libfluidsynth-dev libgme-dev libopenal-dev timidity␣
↪libwildmidi-dev unzip libboost-all-dev     liblua5.1-dev

pip3 install vizdoom
```

Parameters

- **level** (*string*) – ViZDoom configuration file ().
- **include_variables** (*bool*) – Whether to include game variables to state (: false).
- **factored_action** (*bool*) – Whether to use factored action representation (: false).
- **visualize** (*bool*) – Whether to visualize interaction (: false).
- **frame_skip** (*int > 0*) – Number of times to repeat an action without observing (: 12).
- **seed** (*int*) – Random seed (: none).

Index

I

K

L

M

N

O

P

R

S

T

U

V