Tensorforce Documentation

Release 0.5.2

Tensorforce Team

Oct 16, 2019

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 compatible with Python 3 (Python 2 support was dropped with version 0.5).

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:

```
pip 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
pip install -e .
```

Tensorforce is built on top of Google's TensorFlow and requires that either `tensorflow` or `tensorflow-gpu` is installed, currently as version `1.13.1`. To include the correct version of TensorFlow with the installation of Tensorforce, simply add the flag `tf` for the normal CPU version or `tf_gpu` for the GPU version:

```
# PyPI version plus TensorFlow CPU version
pip install tensorforce[tf]

# GitHub version plus TensorFlow GPU version
pip install -e .[tf_gpu]
```

Some environments require additional packages, for which there are also options available (`mazeexp`, `gym`, `retro`, `vizdoom`; or `envs` for all environments), however, some require other tools to be installed (see environments documentation).

CHAPTER 2

Getting started

2.1 Training

```
from tensorforce.agents import Agent
from tensorforce.environments import Environment

# Setup environment
# (Tensorforce or custom implementation, ideally using the Environment interface)
environment = Environment.create(environment='environment.json')

# Create and initialize agent
agent = Agent.create(agent='agent.json', environment=environment)
agent.initialize()

# Reset agent and environment at the beginning of a new episode
agent.reset()
states = environment.reset()
terminal = False

# Agent-environment interaction training loop
while not terminal:
    actions = agent.act(states=states)
    states, terminal, reward = environment.execute(actions=actions)
    agent.observe(terminal=terminal, reward=reward)

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

2.2 Evaluation / application

```
# Agent-environment interaction evaluation loop
while not terminal:
    actions = agent.act(states=states, evaluation=True)
    states, terminal, reward = environment.execute(actions=actions)
```

2.3 Runner utility

```
from tensorforce.execution import Runner

# Tensorforce runner utility
runner = Runner(agent='agent.json', environment='environment.json')

# Run training
runner.run(num_episodes=500)

# Close runner
runner.close()
```

CHAPTER 3

Module 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 How to specify modules

3.1.1 Dictionary with module type and arguments

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

3.1.2 JSON specification file (plus additional arguments)

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

3.1.3 Module path (plus additional arguments)

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

3.1.4 Callable or Type (plus additional arguments)

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

3.1.5 Default module: only arguments or first argument

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

3.2 Static vs dynamic hyperparameters

Tensorforce distinguishes between agent/module arguments (primitive types: bool/int/long/float) which specify either 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 hyperparameters are indicated by `parameter` as part of their 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.

3.2.1 Example: exponentially decaying exploration

```
Agent.create(...
    exploration=dict(
        type='decaying', unit='timesteps', decay='exponential',
        initial_value=0.1, decay_steps=1000, decay_rate=0.5
    ), ...
)
```

3.2.2 Example: linearly increasing horizon

```
Agent.create(...
    reward_estimation=dict(horizon=dict(
        type='decaying', dtype='long', unit='episodes', decay='polynomial',
        initial_value=10.0, decay_steps=1000, final_value=50.0, power=1.0
    ), ...
)
```

CHAPTER 4

Features

4.1 Action masking

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

4.2 Record & pretrain

```
agent = Agent.create(...
    recorder=dict(
        directory='data/traces',
        frequency=100  # record a traces file every 100 episodes
    ), ...
)
...
agent.close()

# Pretrain agent on recorded traces
agent = Agent.create(...)
agent.pretrain(
    directory='data/traces',
```

(continues on next page)

(continued from previous page)

```
    num_updates=100  # perform 100 updates on traces (other configurations possible)
)
```

4.3 Save & restore

```
agent = Agent.create(...
    saver=dict(
        directory='data/checkpoints',
        frequency=600  # save checkpoint every 600 seconds (10 minutes)
    ), ...
)
...
agent.close()

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

4.4 TensorBoard

```
Agent.create(...
    summarizer=dict(
        directory='data/summaries',
        labels=['graph', 'losses', 'rewards'],  # list of labels, or 'all'
        frequency=100  # store values every 100 timesteps
        # (infrequent update summaries every update; other configurations possible)
    ), ...
)
```

CHAPTER 5

run.py – Runner

5.1 Required arguments

#1: **agent** (*string*) – Agent (configuration JSON file, name, or library module)

#2: **environment** (*string*) – Environment (name, configuration JSON file, or library module)

5.2 Optional arguments

5.2.1 Agent arguments

–[n]etwork (*string, default: not specified*) – Network (configuration JSON file, name, or library module)

5.2.2 Environment arguments

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

–[i]mport-modules (*string, default: not specified*) – Import comma-separated modules required for environment

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

5.2.3 Runner arguments

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

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

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

–mean-horizon (*int, default: 10*) – Number of timesteps/episodes for mean reward computation

–e[v]aluation (*bool, default: false*) – Evaluation mode

–[s]ave-best-agent (*bool, default: false*) – Save best-performing agent

5.2.4 Logging arguments

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

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

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

CHAPTER 6

tune.py – Hyperparameter tuner

6.1 Required arguments

#1: **environment** (*string*) – Environment (name, configuration JSON file, or library module)

6.2 Optional arguments

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

–[m]ax-repeats (*int, default: 1*) – Maximum number of repetitions

–[n]um-iterations (*int, default: 1*) – Number of BOHB iterations

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

–[r]estore (*string, default: not specified*) – Restore from given directory

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

CHAPTER 7

Agent interface

CHAPTER 8

Constant Agent

CHAPTER 9

Random Agent

CHAPTER 10

Tensorforce Agent

CHAPTER 11

Deep Q-Network

CHAPTER 12

Dueling DQN

CHAPTER 13

Vanilla Policy Gradient

CHAPTER 14

Actor-Critic

CHAPTER 15

Advantage Actor-Critic

CHAPTER 16

Deterministic Policy Gradient

CHAPTER 17

Proximal Policy Optimization

CHAPTER 18

Trust-Region Policy Optimization

CHAPTER 19

Distributions

CHAPTER 20

Layers

Default layer: `Function` with default argument `function`

20.1 Convolutional layers

20.2 Dense layers

20.3 Embedding layers

20.4 Recurrent layers

20.5 Pooling layers

20.6 Normalization layers

20.7 Misc layers

20.8 Layers with internal states

20.9 Special layers

CHAPTER 21

Memories

Default memory: `Replay` with default argument `capacity`

CHAPTER 22

Networks

Default network: `LayeredNetwork` with default argument `layers`

CHAPTER 23

Objectives

CHAPTER 24

Optimizers

Default optimizer: `MetaOptimizerWrapper`

CHAPTER 25

Parameters

Default parameter: `Constant`

CHAPTER 26

Preprocessing

CHAPTER 27

Policies

Default policy: `ParametrizedDistributions`

CHAPTER 28

Environment interface

class tensorforce.environments.**Environment**

Tensorforce environment interface.

actions()

Returns the action space specification.

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

Return type specification

close()

Closes the environment.

static create(*environment*, ***kwargs*)

Creates an environment from a specification.

Parameters

- **environment** (*specification*) – JSON file, specification key, configuration dictionary, library module, or `Environment` subclass ().
- **kwargs** – Additional arguments.

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))

max_episode_timesteps()

Returns the maximum number of timesteps per episode.

Returns Maximum number of timesteps per episode.

Return type int

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

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

Return type dict[state]

states()
Returns the state space specification.

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

Return type specification

CHAPTER 29

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

git clone https://github.com/mgbellemare/Arcade-Learning-Environment.git
cd Arcade-Learning-Environment

mkdir build && cd build
cmake -DUSE_SDL=ON -DUSE_RLGLUE=OFF -DBUILD_EXAMPLES=ON ..
make -j 4
cd ..

pip install .
```

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 30

Maze Explorer

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

MazeExplorer environment adapter (specification key: `mazeexp`, `maze_explorer`).

May require:

```
sudo apt-get install freeglut3-dev

pip install mazeexp
```

Parameters

- **level** (*int*) – Game mode, see GitHub ().
- **visualize** (*bool*) – Whether to visualize interaction (: false).

CHAPTER 31

Open Sim

class tensorforce.environments.**OpenSim**(*level*, *visualize=False*, *integrator_accuracy=5e-05*)

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 32

OpenAI Gym

class tensorforce.environments.**OpenAIGym**(*level*, *visualize=False*, *max_episode_timesteps=None*, *terminal_reward=0.0*, *reward_threshold=None*, *tags=None*, *visualize_directory=None*, ***kwargs*)

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

May require:

```
pip install gym
pip install gym[all]
```

Parameters

- **level** (*string | gym.Env*) – Gym id or instance ().
- **visualize** (*bool*) – Whether to visualize interaction (: false).
- **max_episode_timesteps** (*false | int > 0*) – Whether to terminate an episode after a while, and if so, maximum number of timesteps per episode (: Gym default).
- **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).
- **tags** (*dict*) – Gym environment argument, a set of arbitrary key-value tags on this environment, including simple property=True tags (: Gym default).
- **visualize_directory** (*string*) – Visualization output directory (: none).
- **kwargs** – Additional Gym environment arguments.

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:

```
pip 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

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

pip install git+https://github.com/pygame/pygame.git

pip 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 'doom', 'flappybird', 'monsterkong', 'catcher', '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 35

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

pip 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