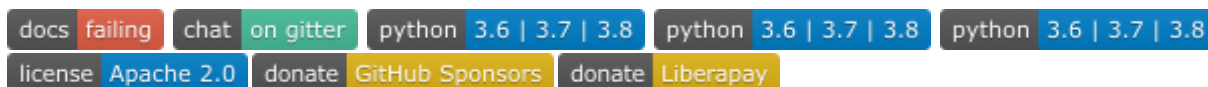

Tensorforce: a TensorFlow library for applied reinforcement learning



This project is not maintained any longer!

Introduction 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.

Quicklinks

- Documentation and update notes
- Contact and Gitter channel
- Benchmarks and projects using Tensorforce
- Roadmap and contribution guidelines
- GitHub Sponsors and Liberapay

Table of content

- Installation
- Quickstart example code
- Command line usage
- Features
- Environment adapters

-
- Support, feedback and donating
 - Core team and contributors
 - Cite Tensorforce

Installation

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

```
1 pip3 install tensorforce
```

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

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

Note on installation on M1 Macs: At the moment Tensorflow, which is a core dependency of Tensorforce, cannot be installed on M1 Macs directly. Follow the “M1 Macs” section in the documentation for a workaround.

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.

Note on GPU usage: Different from (un)supervised deep learning, RL does not always benefit from running on a GPU, depending on environment and agent configuration. In particular for environments with low-dimensional state spaces (i.e., no images), it is hence worth trying to run on CPU only.

Quickstart example code

```
1 from tensorforce import Agent, Environment
2
3 # Pre-defined or custom environment
4 environment = Environment.create(
5     environment='gym', level='CartPole', max_episode_timesteps=500
6 )
7
8 # Instantiate a Tensorforce agent
9 agent = Agent.create(
10     agent='tensorforce',
11     environment=environment, # alternatively: states, actions, (
12         max_episode_timesteps)
13     memory=10000,
14     update=dict(unit='timesteps', batch_size=64),
```

```
14     optimizer=dict(type='adam', learning_rate=3e-4),
15     policy=dict(network='auto'),
16     objective='policy_gradient',
17     reward_estimation=dict(horizon=20)
18 )
19
20 # Train for 300 episodes
21 for _ in range(300):
22
23     # Initialize episode
24     states = environment.reset()
25     terminal = False
26
27     while not terminal:
28         # Episode timestep
29         actions = agent.act(states=states)
30         states, terminal, reward = environment.execute(actions=actions)
31         agent.observe(terminal=terminal, reward=reward)
32
33 agent.close()
34 environment.close()
```

Command line usage

Tensorforce comes with a range of example configurations for different popular reinforcement learning environments. For instance, to run Tensorforce's implementation of the popular Proximal Policy Optimization (PPO) algorithm on the OpenAI Gym CartPole environment, execute the following line:

```
1 python3 run.py --agent benchmarks/configs/ppo.json --environment gym \
2     --level CartPole-v1 --episodes 100
```

For more information check out the documentation.

Features

- **Network layers:** Fully-connected, 1- and 2-dimensional convolutions, embeddings, pooling, RNNs, dropout, normalization, and more; *plus* support of Keras layers.
- **Network architecture:** Support for multi-state inputs and layer (block) reuse, simple definition of directed acyclic graph structures via register/retrieve layer, plus support for arbitrary architectures.
- **Memory types:** Simple batch buffer memory, random replay memory.

-
- **Policy distributions:** Bernoulli distribution for boolean actions, categorical distribution for (finite) integer actions, Gaussian distribution for continuous actions, Beta distribution for range-constrained continuous actions, multi-action support.
 - **Reward estimation:** Configuration options for estimation horizon, future reward discount, state/state-action/advantage estimation, and for whether to consider terminal and horizon states.
 - **Training objectives:** (Deterministic) policy gradient, state-(action-)value approximation.
 - **Optimization algorithms:** Various gradient-based optimizers provided by TensorFlow like Adam/AdaDelta/RMSProp/etc, evolutionary optimizer, natural-gradient-based optimizer, plus a range of meta-optimizers.
 - **Exploration:** Randomized actions, sampling temperature, variable noise.
 - **Preprocessing:** Clipping, deltafier, sequence, image processing.
 - **Regularization:** L2 and entropy regularization.
 - **Execution modes:** Parallelized execution of multiple environments based on Python's `multiprocessing` and `socket`.
 - **Optimized act-only SavedModel extraction.**
 - **TensorBoard support.**

By combining these modular components in different ways, a variety of popular deep reinforcement learning models/features can be replicated:

- Q-learning: Deep Q-learning, Double-DQN, Dueling DQN, n-step DQN, Normalised Advantage Function (NAF)
- Policy gradient: vanilla policy-gradient / REINFORCE, Actor-critic and A3C, Proximal Policy Optimization, Trust Region Policy Optimization, Deterministic Policy Gradient

Note that in general the replication is not 100% faithful, since the models as described in the corresponding paper often involve additional minor tweaks and modifications which are hard to support with a modular design (and, arguably, also questionable whether it is important/desirable to support them). On the upside, these models are just a few examples from the multitude of module combinations supported by Tensorforce.

Environment adapters

- Arcade Learning Environment, a simple object-oriented framework that allows researchers and hobbyists to develop AI agents for Atari 2600 games.
- CARLA, is an open-source simulator for autonomous driving research.
- OpenAI Gym, a toolkit for developing and comparing reinforcement learning algorithms which supports teaching agents everything from walking to playing games like Pong or Pinball.

-
- OpenAI Retro, lets you turn classic video games into Gym environments for reinforcement learning and comes with integrations for ~1000 games.
 - OpenSim, reinforcement learning with musculoskeletal models.
 - PyGame Learning Environment, learning environment which allows a quick start to Reinforcement Learning in Python.
 - ViZDoom, allows developing AI bots that play Doom using only the visual information.

Support, feedback and donating

Please get in touch via mail or on Gitter if you have questions, feedback, ideas for features/collaboration, or if you seek support for applying Tensorforce to your problem.

If you want to support the Tensorforce core team (see below), please also consider donating: GitHub Sponsors or Liberapay.

Core team and contributors

Tensorforce is currently developed and maintained by Alexander Kuhnle.

Earlier versions of Tensorforce ($\leq 0.4.2$) were developed by Michael Schaarschmidt, Alexander Kuhnle and Kai Fricke.

The advanced parallel execution functionality was originally contributed by Jean Rabault (@jerabaul29) and Vincent Belus (@vbelus). Moreover, the pretraining feature was largely developed in collaboration with Hongwei Tang (@thw1021) and Jean Rabault (@jerabaul29).

The CARLA environment wrapper is currently developed by Luca Anzalone (@luca96).

We are very grateful for our open-source contributors (listed according to Github, updated periodically):

Islandman93, sven1977, Mazecreator, wassname, lefnire, daggertye, trickmeyer, mkempers, mryellow, ImpulseAdventure, janislavjankov, andrewekhalel, HassamSheikh, skervim, beflix, coord-e, benelot, tms1337, vwxyzjn, erniejunior, Deathn0t, petrbel, nrhodes, batu, yellowbee686, tgianko, AdamStelmaszczyk, BorisSchaeling, christianhidber, Davidnet, ekerazha, gitter-badger, kborozdin, Kismuz, mannsi, milesmcc, nagachika, neitzal, ngoodger, perara, sohakes, tomhennigan.

Cite Tensorforce

Please cite the framework as follows:

```
1 @misc{tensorflow,  
2   author      = {Kuhnle, Alexander and Schaarschmidt, Michael and  
3     Fricke, Kai},  
4   title       = {Tensorforce: a TensorFlow library for applied  
5     reinforcement learning},  
6   howpublished = {Web page},  
7   url         = {https://github.com/tensorforce/tensorforce},  
8   year        = {2017}  
9 }
```

If you use the parallel execution functionality, please additionally cite it as follows:

```
1 @article{rabault2019accelerating,  
2   title       = {Accelerating deep reinforcement learning strategies  
3     of flow control through a multi-environment approach},  
4   author      = {Rabault, Jean and Kuhnle, Alexander},  
5   journal     = {Physics of Fluids},  
6   volume      = {31},  
7   number      = {9},  
8   pages       = {094105},  
9   year        = {2019},  
10  publisher    = {AIP Publishing}  
11 }
```

If you use Tensorforce in your research, you may additionally consider citing the following paper:

```
1 @article{lift-tensorforce,  
2   author      = {Schaarschmidt, Michael and Kuhnle, Alexander and  
3     Ellis, Ben and Fricke, Kai and Gessert, Felix and Yoneki, Eiko},  
4   title       = {{LIFT}: Reinforcement Learning in Computer Systems by  
5     Learning From Demonstrations},  
6   journal     = {CoRR},  
7   volume      = {abs/1808.07903},  
8   year        = {2018},  
9   url         = {http://arxiv.org/abs/1808.07903},  
10  archivePrefix = {arXiv},  
11  eprint       = {1808.07903}  
12 }
```