Journal of Machine Learning Research 23 (2022) 1-6 Submitted 9/21; Revised 5/22; Published 8/22

# Tianshou: A Highly Modularized Deep Reinforcement Learning Library

**Jiayi Weng**[‡*] TRINKLE23897@GMAIL.COM
**Huayu Chen**[‡*] CHENHUAY21@MAILS.TSINGHUA.EDU.CN
**Dong Yan**[‡] SPROBLVEM@GMAIL.COM
**Kaichao You**[§] YOUKAICHAO@GMAIL.COM
**Alexis Duburcq**[¶] ALEXIS.DUBURCQ@GMAIL.COM
**Minghao Zhang**[§] MEHOOZHANG@GMAIL.COM
**Yi Su**[♯] NUANCE@GMAIL.COM
**Hang Su**[‡†] SUHANGSS@TSINGHUA.EDU.CN
**Jun Zhu**[‡†] DCSZJ@TSINGHUA.EDU.CN
[‡]*Dept. of Comp. Sci. & Tech., BNRist Center, Institute for AI, Tsinghua-Bosch Joint ML Center, THBI Lab, Tsinghua University, Beijing, 100084, China*
[§]*School of Software, Tsinghua University, Beijing, 100084, China*
[¶]*Wandercraft, 88 Rue de Rivoli, Paris, 75004, France*
[♯]*Ant Group, 525 Almanor Ave, Sunnyvale, CA, 94085, United States of America*

**Editor:** Antti Honkela

## Abstract

In this paper, we present `Tianshou`, a highly modularized Python library for deep reinforcement learning (DRL) that uses PyTorch as its backend. Tianshou intends to be research-friendly by providing a flexible and reliable infrastructure of DRL algorithms. It supports online and offline training with more than 20 classic algorithms through a unified interface. To facilitate related research and prove Tianshou's reliability, we have released Tianshou's benchmark of MuJoCo environments, covering eight classic algorithms with state-of-the-art performance. We open-sourced Tianshou at `https://github.com/thu-ml/tianshou/`.

**Keywords:** Deep Reinforcement Learning, Library, PyTorch, Modularized, Benchmark

## 1. Introduction

Recent advances in deep reinforcement learning (DRL) have ignited enthusiasm of both academia and industry. This is accompanied by the flourish of many newly-emerged DRL algorithms (Mnih et al., 2015; Silver et al., 2017; Duan et al., 2016), together with numerous libraries that try to provide reference implementations with the representative ones including `RLlib` (Liang et al., 2018), `rlpyt` (Stooke and Abbeel, 2019), `Stable-Baselines3` (Raffin et al., 2021), `MushroomRL` (D'Eramo et al., 2020), and `PFRL` (Fujita et al., 2021).

Though most DRL libraries are comprehensive, many researchers still tend to use their own DRL code base for fast prototyping in practice. The reasons behind this are various. Some libraries may choose to highly encapsulate the supported algorithms, leaving out

∗. J. Weng and H. Chen contributed equally to this article.
†. H. Su and J. Zhu are corresponding authors.

©2022 Jiayi Weng, Huayu Chen, Dong Yan, Kaichao You, Alexis Duburcq, Minghao Zhang, Yi Su, Hang Su and Jun Zhu.

License: CC-BY 4.0, see `https://creativecommons.org/licenses/by/4.0/`. Attribution requirements are provided at `http://jmlr.org/papers/v23/21-1127.html`.

several options to be tweaked. This assists algorithms' application, but harms the flexibility because it is impossible to provide exhaustive options. The usability is also a concern. Several libraries prioritize supporting highly distributed training, but bring complex code structure and difficulties in debugging. However, parallelized data sampling is still needed in research because of the typically unbalanced numbers of CPUs and GPUs in a server. Another reason might be that many libraries are only comprehensive for a certain type of algorithms (e.g., only support online or offline algorithms) to keep a unified API.

To address these issues, we present Tianshou, a highly modularized Python library for deep reinforcement learning based on PyTorch. Tianshou has the following characteristics:

- **Highly modularized.** Tianshou aims to provide building blocks rather than training scripts. It can be easily used for fast prototyping because the shared infrastructure commonly used in DRL is factored out (Figure 1). Users only need to change a few variables to apply techniques commonly used in DRL (e.g., parallel data sampling).

- **Reliable.** Tianshou has a code coverage of 94%. Every commit to Tianshou will go through unit tests on multiple platforms. We have also released a systematic benchmark of Gym's MuJoCo environment[1] (Todorov et al., 2012; Brockman et al., 2016). In this benchmark, Tianshou incorporates a comprehensive set of DRL techniques for 8 benchmarked algorithms and scores 15% higher on average in terms of median performance compared with reference implementations.

- **Comprehensive.** Besides comprehensive model-free algorithms, Tianshou also supports offline learning and many other DRL techniques such as GAIL (Ho and Ermon, 2016) and ICM (Pathak et al., 2017). Moreover, through a unified Python interface, Tianshou formulates the data collecting (e.g., both synchronous and asynchronous environment execution) and agent training paradigms in DRL. Lastly, Tianshou has plentiful functionalities that may extend its application (see Section 2).

## 2. Architecture of Tianshou

In this section, we will briefly introduce Tianshou's architecture as illustrated in Figure 1.

**Standardization of the Training Process.** We standardize the training paradigms of mainstream DRL algorithms by considering different experience replay mechanisms and classify them into three types: on-policy training, off-policy training, and offline learning. We use a replay buffer to store the transitions and a collector to collect transition data into the buffer. We use the policy's `update` function to update the parameter (Figure 2).

**Parallel Computing Infrastructure.** In concurrent research, Stooke and Abbeel (2019) addresses two phases of parallelization in DRL: environment sampling and agent training. Tianshou targets small- to medium-scale research, so it focuses on the first one. Following Clemente et al. (2017), we adopt their parallelization technique to balance simulation and inference loads. Note that our contributions to parallel sampling schemes exceed this work by allowing asynchronized sampling as an alternative way to ease the straggler effect instead of only stacking environment instances per process. Thanks to the modular-

1. `https://tianshou.readthedocs.io/en/master/tutorials/benchmark.html`

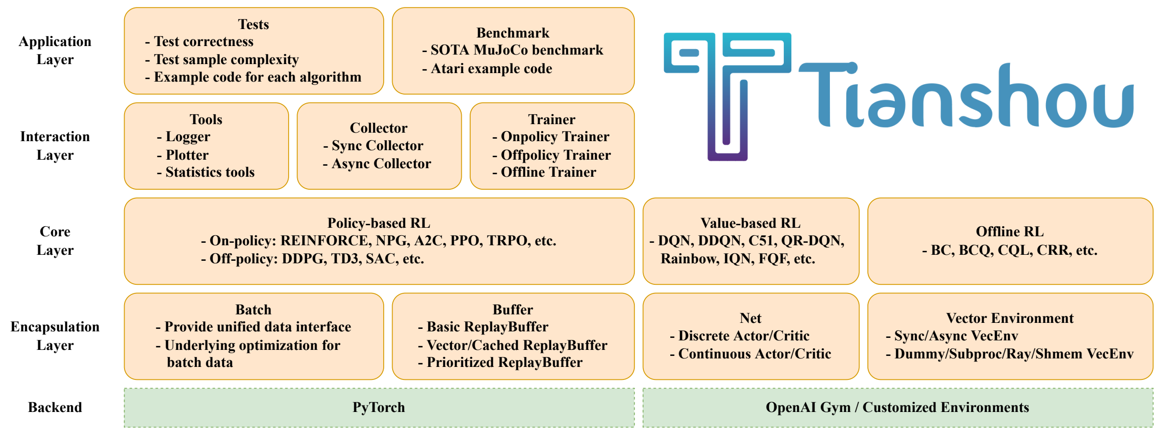

Figure 1: Tianshou's architecture. Tianshou consists of 4 layers: the bottom-level encapsulation layer provides encapsulations of third-party libraries; the core layer implements many kinds of DRL algorithms; the interaction layer aims to offer user-oriented high-level APIs; and finally, the application layer provides training scripts to demonstrate Tianshou's usage. Instances of the same building block share almost the same pythonic APIs.

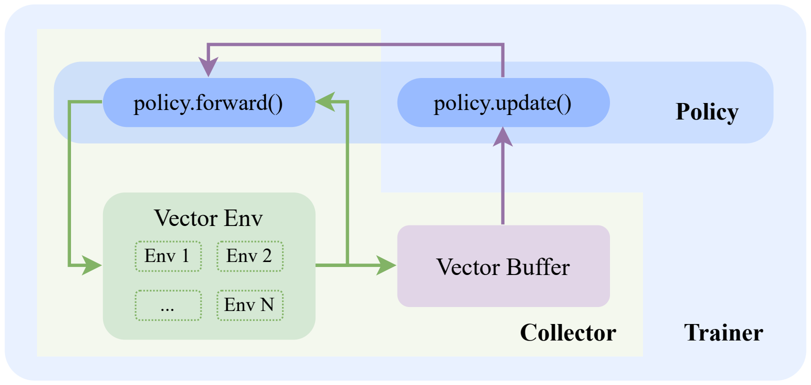

Figure 2: A high-level depiction of Tianshou's training process.

ized design, Tianshou can easily support the C++-based vectorized environment EnvPool (Weng et al., 2022) with free speed up.

**Utilities.** Tianshou intends to relieve users from imperceptible details critical to a desirable performance and, hence, incorporates a comprehensive set of DRL techniques as its infrastructure. Techniques include partial-episode bootstrapping (Pardo et al., 2018), observation/value normalization (van Hasselt et al., 2016), automatic action scaling and GAE (Schulman et al., 2016), etc. Besides, Tianshou has plentiful extra functionalities that users might find helpful. For instance, Tianshou has customizable loggers compatible with TensorBoard and W&B. Recurrent state representation, prioritized experience replay, training resumption, and buffer serialization are also supported.

**Reproduction Scripts and Performance.** We have released Tianshou's OpenAI Gym MuJoCo task suite benchmark, covering 8 classic algorithms and 9 environments. In this benchmark, Tianshou scores 15% higher on average compared with multiple reference implementations in terms of 9 environments' median performance, demonstrating its reliability. For discrete action space problems, we also provide example code and results with 7 supported algorithms in 7 Atari environments. All experiments are done with 10 random seeds. Some libraries (Fujita et al., 2021) devote themselves to faithfully replicating existing papers, while Tianshou aims to present an as-consistent-as-possible set of hyperparameters and low-level designs. While leaving the core algorithm untouched, we try to incorporate several known tricks in a specific algorithm to all similar algorithms supported by Tianshou. Hopefully, this will facilitate comparisons between algorithms.

## 3. Usability

Tianshou is lightweight and easy to install. Users can simply install Tianshou via Pip or Conda on different platforms (Windows, macOS, Linux). Full API documentation and a series of tutorials are provided at `https://tianshou.readthedocs.io/`. Only a few lines of code are required to start a simple experiment. Tianshou also strictly follows the PEP8 code style with the code commented and data type annotated. Contributing guidelines and extensive unit tests with GitHub Actions, including code-style, type, and performance checks, help Tianshou maintain its code quality.

## 4. Comparison to Related Works

While several TensorFlow-based DRL libraries (Kuhnle et al., 2017; Plappert, 2016; Caspi et al., 2017) are also worth mentioning, we limit our comparison to a few DRL libraries with the PyTorch backend due to page limit. `RLlib` (Liang et al., 2018) and `rlpyt` (Stooke and Abbeel, 2019) are libraries designed to be high-throughput software and support both multi-CPU parallel sampling and multi-GPU optimization, while `Stable-Baselines3` (Raffin et al., 2021), `PFRL` (Fujita et al., 2021) and Tianshou focus on small- to medium-scale application of DRL algorithms and support only parallel sampling. Libraries like `MushroomRL` (D'Eramo et al., 2020) are intentionally designed to be research-friendly, so no parallelization is supported. This leads to very different design choices and results in different code complexity. In terms of supported algorithms, most libraries are comprehensive, but each has a different focus. `MushroomRL` supports both classic RL and DRL algorithms to facilitate research. `d3rlpy` (Takuma Seno, 2021) prioritizes supporting offline DRL algorithms and other libraries mainly support online algorithms. While many libraries above are modular, Tianshou achieves modularity mainly by factoring out the infrastructure in DRL, compared to several libraries that focus on offering one highly encapsulated API for each algorithm (Raffin et al., 2021; Takuma Seno, 2021). Above all, the architecture of `PFRL` is most similar to Tianshou. However, they still have key differences like the implementation of lower-level data container Batch and how to support sequence Buffers for RNN. Other engineering features are compared in Tianshou's GitHub repository[2].

2. `https://github.com/thu-ml/tianshou/blob/master/README.md#why-tianshou`

## 5. Conclusion

This paper briefly describes Tianshou, a flexible and reliable implementation of a modular DRL library. Tianshou sets up a framework for DRL research by factoring out the shared infrastructure commonly used in DRL as building blocks. We have also released a MuJoCo benchmark, covering many classic algorithms, demonstrating Tianshou's reliability.

## Acknowledgments

We thank Haosheng Zou for his early work on TensorFlow-based Tianshou before version 0.1.1. We thank Peng Zhong, Qiang He, Chengqi Duan, Qing Xiao, Qifan Li, Yan Li, and others for their valuable contributions to Tianshou.

This work was supported by the National Key Research and Development Program of China ( 2020AAA0106000, 2020AAA0104304, 2020AAA0106302, 2021YFB2701000), NSFC Projects (Nos. 62061136001, 62076147, U19B2034, U1811461, U19A2081, 61972224), Beijing NSF Project (No. JQ19016), BNRist (BNR2022RC01006), Tsinghua Institute for Guo Qiang, Beijing Academy of Artificial Intelligence (BAAI), Tsinghua-Huawei Joint Research Program, and the High Performance Computing Center, Tsinghua University.

## References

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

Itai Caspi, Gal Leibovich, Gal Novik, and Shadi Endrawis. Reinforcement learning Coach. `https://github.com/IntelLabs/coach`, 2017.

Alfredo V Clemente, Humberto N Castejón, and Arjun Chandra. Efficient parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1705.04862*, 2017.

Carlo D'Eramo, Davide Tateo, Andrea Bonarini, Marcello Restelli, and Jan Peters. MushroomRL: Simplifying reinforcement learning research. *arXiv preprint arXiv:2001.01102*, 2020.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, pages 1329–1338. PMLR, 2016.

Yasuhiro Fujita, Prabhat Nagarajan, Toshiki Kataoka, and Takahiro Ishikawa. ChainerRL: A deep reinforcement learning library. *Journal of Machine Learning Research*, 22(77): 1–14, 2021.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016.

Alexander Kuhnle, Michael Schaarschmidt, and Kai Fricke. Tensorforce: a tensorflow library for applied reinforcement learning. `https://github.com/tensorforce/tensorforce`, 2017.

Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael I. Jordan, and Ion Stoica. RLlib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning*, pages 3059–3068, 2018.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Fabio Pardo, Arash Tavakoli, Vitaly Levdik, and Petar Kormushev. Time limits in reinforcement learning. In *International Conference on Machine Learning*, pages 4045–4054. PMLR, 2018.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

Matthias Plappert. keras-rl. `https://github.com/keras-rl/keras-rl`, 2016.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-Baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 2021.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Adam Stooke and Pieter Abbeel. rlpyt: A research code base for deep reinforcement learning in pytorch. *arXiv preprint arXiv:1909.01500*, 2019.

Michita Imai Takuma Seno. d3rlpy: An offline deep reinforcement library. In *NeurIPS 2021 Offline Reinforcement Learning Workshop*, December 2021.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

Hado P van Hasselt, Arthur Guez, Matteo Hessel, Volodymyr Mnih, and David Silver. Learning values across many orders of magnitude. *Advances in Neural Information Processing Systems*, 29:4287–4295, 2016.

Jiayi Weng, Min Lin, Shengyi Huang, Bo Liu, Denys Makoviichuk, Viktor Makoviychuk, Zichen Liu, Yufan Song, Ting Luo, Yukun Jiang, Zhongwen Xu, and Shuicheng Yan. EnvPool: A highly parallel reinforcement learning environment execution engine. *arXiv preprint arXiv:2206.10558*, 2022.