Journal of Machine Learning Research 22 (2021) 1-8 Submitted 12/20; Revised 10/21; Published 11/21

Stable-Baselines3: Reliable Reinforcement Learning Implementations

Antonin Raffin[1] ANTONIN.RAFFIN@DLR.DE
Ashley Hill[2] ASHLEY.HILL@CEA.FR
Adam Gleave[3] GLEAVE@BERKELEY.EDU
Anssi Kanervisto[4] ANSSK@CS.UEF.FI
Maximilian Ernestus[5] MAXIMILIAN@ERNESTUS.COM
Noah Dormann[1] NOAH.DORMANN@DLR.DE

[1] *Robotics and Mechatronics Center (RMC), German Aerospace Center (DLR), Weßling, Germany*
[2] *Interactive Robotics Laboratory, University Paris-Saclay, CEA, Palaiseau, France*
[3] *Electrical Engineering and Computer Science, University of California, Berkeley, CA, USA*
[4] *School of Computing, University of Eastern Finland, Joensuu, Finland*
[5] *Kiteswarms GmbH, Freiburg, Germany*

Editor: Andreas Mueller

Abstract

Stable-Baselines3 provides open-source implementations of deep reinforcement learning (RL) algorithms in Python. The implementations have been benchmarked against reference codebases, and automated unit tests cover 95% of the code. The algorithms follow a consistent interface and are accompanied by extensive documentation, making it simple to train and compare different RL algorithms. Our documentation, examples, and source-code are available at `https://github.com/DLR-RM/stable-baselines3`.

Keywords: Reinforcement Learning, Baselines, Software, Open-Source, Python, PyTorch

1. Introduction

Deep reinforcement learning (RL) research has grown rapidly in recent years, yet results are often difficult to reproduce (Henderson et al., 2018). A major challenge is that small implementation details can have a substantial effect on performance – often greater than the difference between algorithms (Engstrom et al., 2020). It is particularly important that implementations used as experimental *baselines* are reliable; otherwise, novel algorithms compared to weak baselines lead to inflated estimates of performance improvements.

To address this challenge, we propose Stable-Baselines3 (SB3), an open-source framework implementing seven commonly used model-free deep RL algorithms (see Section 2). We take great care to adhere to software engineering best practices to achieve high-quality implementations that match prior results. Each algorithm has been benchmarked on common environments (Raffin and Stulp, 2020) and compared to prior implementations. Our test suite covers 95% of the code and, together with our active user base[1] scrutinizing changes, ensures that any implementation errors are minimized.

1. At the time of writing, SB3 had 800+ stars on GitHub, 100+ closed issues and 80+ merged pull requests

```python
import gym
from stable_baselines3 import SAC
# Train an agent using Soft Actor-Critic on Pendulum-v0
env = gym.make("Pendulum-v0")
model = SAC("MlpPolicy", env).learn(total_timesteps=20000)
# Save the model
model.save("sac_pendulum")
# Load the trained model
model = SAC.load("sac_pendulum")
# Start a new episode
obs = env.reset()
# What action to take in state `obs`?
action, _ = model.predict(obs, deterministic=True)
```

Figure 1: Using Stable-Baselines3 to train, save, load, and infer an action from a policy.

Stable-Baselines3 builds on the experience gained from maintaining our previous implementation, Stable-Baselines2 (SB2; Hill et al., 2018)[2], that was forked from OpenAI Baselines (Dhariwal et al., 2017) and uses TensorFlow (Abadi et al., 2016). SB3 is a complete rewrite of the codebase implemented in PyTorch (Paszke et al., 2019), the framework preferred by a majority of our users in a survey (Raffin, 2020a). SB3 maintains a similar API, allowing a seamless upgrade pathway from SB2.[3]

Our main goal is to provide a user-friendly and reliable RL library. To keep SB3 simple to use and maintain, we focus on model-free, single-agent RL algorithms, and rely on external projects to extend the scope to imitation (Wang et al., 2020) and offline learning (Seno, 2020). We prioritize maintaining *stable* implementations over adding new features or algorithms, and avoid making breaking changes. We provide a consistent, clean and fully documented API, inspired by the `scikit-learn` API (Pedregosa et al., 2011). Our code is easily modifiable by users as we favour readability and simplicity over modularity, although we make use of object-oriented programming to reduce code duplication.

2. Features

Simple API. Figure 1 shows that training agents in Stable-Baselines3 takes just a few lines of code, after which the agent can be queried for actions. This allows researchers to easily use the baseline algorithms and components in their experiments (e.g. Klink et al. (2020); Nair et al. (2019); Gleave et al. (2020)), as well as apply RL to novel tasks and environments, like continual learning when attacking WiFi networks (Margaritelli, 2020) or dampening bridge vibrations (Berkowitz, 2019).

Documentation. SB3 comes with extensive documentation of the code API.[4] We also include a user guide, covering both basic and more advanced usage with a collection of concrete examples. Moreover, we have developed a Colab notebook based RL tutorial,[5] enabling users to demo the library directly in the browser. Additionally, we include common

2. SB2 has 650+ closed issues, 220+ merged pull requests and 200+ citations on Google Scholar.
3. Upgrade guide: `https://stable-baselines3.readthedocs.io/en/master/guide/migration.html`.
4. `https://stable-baselines3.readthedocs.io/en/master/`
5. `https://github.com/araffin/rl-tutorial-jnrr19`

tips for running RL experiments and a developer guide. We also pay close attention to questions and uncertainties from SB3 users, updating the documentation to address these.

High-Quality Implementations. Algorithms are verified against published results by comparing the agent learning curves[6]. Moreover, all functions are typed (parameter and return types) and documented with a consistent style, and most functions are covered by unit tests. Continuous integration checks that all changes pass unit tests and type check, as well as validating the code style and documentation.

Comprehensive. Stable-Baselines3 contains the following state-of-the-art on- and off-policy algorithms, commonly used as experimental baselines: A2C (Mnih et al., 2016), PPO (Schulman et al., 2017), DDPG (Lillicrap et al., 2016), SAC (Haarnoja et al., 2018), TD3 (Fujimoto et al., 2018), HER (Andrychowicz et al., 2017) and DQN (Mnih et al., 2015).

Moreover, SB3 provides various algorithm-independent features. We support logging to CSV files and TensorBoard. Users can log custom metrics and modify training via user-provided callbacks. To speed up training, we support parallel (or "vectorized") environments. To simplify training, we implement common environment wrappers, like preprocessing Atari observations to match the original DQN experiments (Mnih et al., 2015).

Experimental Framework. RL Baselines Zoo (Raffin, 2018, 2020b) provides scripts to train and evaluate agents, tune hyperparameters, record videos, store experiment setup and visualize results. We also include a collection of pre-trained reinforcement learning agents together with tuned hyperparameters for simple control tasks, PyBullet environments (Coumans and Bai, 2016–2019) and Atari games, optimized using Optuna (Akiba et al., 2019). We follow best practices for training and evaluation (Henderson et al., 2018), such as evaluating in a separate environment, using deterministic evaluation where required (SAC) and storing all hyperparameters necessary to replicate the experiment.

Stable-Baselines3 Contrib. We implement experimental features in a separate contrib repository (Raffin et al., 2020). This allows SB3 to maintain a stable and compact core, while still providing the latest features, like Truncated Quantile Critics (Kuznetsov et al., 2020). Implementations in contrib need not be tightly integrated with the main SB3 codebase, but we maintain the same stringent review requirements to ensure users can trust the contrib implementations. Implementations from contrib that have stood the test of time may be integrated into the main repository.

3. Comparison to Related Software

Most libraries are targeted at experienced RL researchers, requiring expert knowledge to use (Weng et al., 2020; Hoffman et al., 2020; Fujita et al., 2021; Castro et al., 2018; Guadarrama et al., 2018; Gauci et al., 2018; Stooke and Abbeel, 2019; Kolesnikov, 2018). Only a few RL libraries offer more than a brief API documentation (garage contributors, 2019; Liang et al., 2018; Kuhnle et al., 2017; Guadarrama et al., 2018), and some are notoriously hard to understand.[7] By contrast, Stable-Baselines3 is designed to be easy to use and comes with extensive documentation and tutorials.

6. For example, issue #48 or issue #49.
7. OpenAI Baselines (Dhariwal et al., 2017), see `https://www.reddit.com/r/MachineLearning/comments/95ft1j/`. This was a major starting point for Stable-Baselines2 (Hill et al., 2018)

The previous version of Stable-Baselines3, Stable-Baselines2, was created as a fork of OpenAI Baselines (Dhariwal et al., 2017) but the two codebases quickly diverged (see PR #481). SB3 is a complete rewrite of Stable-Baselines2 in PyTorch that keeps the major improvements and new algorithms from SB2 while going even further into improving code quality (e.g. cleaner codebase, better test coverage, type hints). More precisely, compared to Baselines, SB3 is fully documented, commented, tested, has 4 additional algorithms (SAC, TD3, QR-DQN, TQC[8]) and many additional features (e.g. dictionary observation support, callbacks, evaluation with multiple environments, environment checker). The only legacy features of OpenAI Baselines are the code structure (one folder per algorithm), the use of code-level optimizations and the environment tools which are greatly improved[9] (additional features, bug fixes, comments, documentation and more testing).

Many libraries have a modular design (Caspi et al., 2017; Keng and Graesser, 2017; Hoffman et al., 2020; garage contributors, 2019). This allows them to quickly combine advances from different papers, but forces new users to understand the full code structure before being able to tweak the library. On the other extreme, educational implementations like Spinning Up (Achiam, 2018) are self-contained but hard to maintain due to code duplication. SB3 strives to strike a balance: factoring out widely used components like replay buffers, but minimizing the amount of code that needs to be understood to modify an algorithm.

As an exhaustive comparison to all RL libraries is not possible, in Table 1 we compare SB3 to a subset of other active or popular libraries, with a focus on quality of implementation and openness to new users.

RLlib (Liang et al., 2018) scores highly in the table, but is targeted at a different use-case from SB3. Whereas SB3 focuses on simplicity and reliability, RLlib (Liang et al., 2018) prioritizes scalability and support for distributed training. Additionally, RLlib includes both a PyTorch and TensorFlow backend, and includes support for multi-agent training. This versatility comes at a cost of a larger and more complex codebase.

Overall, we find SB3 compares favourably to other libraries in terms of documentation, testing and activity.

	SB3	OAI Baselines	PFRL	RLlib	Tianshou	Acme	Tensorforce
Backend	PyTorch	TF	PyTorch	PyTorch/TF	PyTorch	Jax/TF	TF
User Guide / Tutorials	✓/ ✓	✗/ ▬	▬/ ✓	✓/ ✓	▬/ ✓	▬/ ✓	✓/ ▬
API Documentation	✓	✗	✓	✓	✓	✗	✓
Benchmark	✓	✓	✓	✓	▬	▬	▬
Pretrained models	✓	✗	✓	✗	✗	✗	✗
Test Coverage	95%	49%	?	?	94%	74%	81%
Type Checking	✓	✗	✗	✓	✓	✓	✗
Issue / PR Template	✓	✗	✗	✓	✓	✗	✗
Last Commit (age)	< 1 week	> 6 months	< 1 month	< 1 week	< 1 month	< 1 week	< 1 month
Approved PRs (6 mo.)	75	0	13	222	85	5	7

Table 1: Comparison of SB3 to a representative subset of active or popular RL libraries. **Key**: ▬ means that the feature is only partially present; OAI: OpenAI; TF: TensorFlow; PR: Pull Request.

8. QR-DQN and TQC are in the contrib repo.
9. As an example, one can compare "VecNormalize" in OAI Baselines vs SB3.

Acknowledgments

The work described in this paper was partially funded by the project "Reduced Complexity Models" from the "Helmholtz-Gemeinschaft Deutscher Forschungszentren" and by the EU H2020 project "VERtical Innovation in the Domain of Robotics Enabled by Artificial intelligence Methods".

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: A system for large-scale machine learning. In *USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, page 265–283, 2016.

Joshua Achiam. Spinning up in deep reinforcement learning. `https://github.com/openai/spinningup`, 2018.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2019.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Neural Information Processing Systems*, pages 5048–5058, 2017.

Jack Berkowitz. WaveRL. `https://github.com/jaberkow/WaveRL`, 2019.

Itai Caspi, Gal Leibovich, Gal Novik, and Shadi Endrawis. Reinforcement learning Coach. `https://doi.org/10.5281/zenodo.1134899`, December 2017.

Pablo Samuel Castro, Subhodeep Moitra, Carles Gelada, Saurabh Kumar, and Marc G. Bellemare. Dopamine: A research framework for deep reinforcement learning. arXiv: 1812.06110v1 [cs.LG], 2018.

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2019.

Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. OpenAI Baselines. `https://github.com/openai/baselines`, 2017.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep rl: A case study on ppo and trpo. In *International Conference on Learning Representations*, 2020.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, volume 80, pages 1587–1596, 2018.

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. URL `http://jmlr.org/papers/v22/20-376.html`.

The garage contributors. Garage: A toolkit for reproducible reinforcement learning research. `https://github.com/rlworkgroup/garage`, 2019.

Jason Gauci, Edoardo Conti, Yitao Liang, Kittipat Virochsiri, Zhengxing Chen, Yuchen He, Zachary Kaden, Vivek Narayanan, and Xiaohui Ye. Horizon: Facebook's open source applied reinforcement learning platform. arXiv:1811.00260v5 [cs.LG], 2018.

Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. Adversarial policies: Attacking deep reinforcement learning. In *International Conference on Learning Representations*, 2020.

Sergio Guadarrama, Anoop Korattikara, Oscar Ramirez, Pablo Castro, Ethan Holly, Sam Fishman, Ke Wang, Ekaterina Gonina, Neal Wu, Efi Kokiopoulou, Luciano Sbaiz, Jamie Smith, Gábor Bartók, Jesse Berent, Chris Harris, Vincent Vanhoucke, and Eugene Brevdo. TF-Agents: A library for reinforcement learning in tensorflow. `https://github.com/tensorflow/agents`, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870, July 2018.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *The AAAI Conference on Artificial Intelligence*, pages 3207–3214, 2018.

Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable Baselines. `https://github.com/hill-a/stable-baselines`, 2018.

Matt Hoffman, Bobak Shahriari, John Aslanides, Gabriel Barth-Maron, Feryal Behbahani, Tamara Norman, Abbas Abdolmaleki, Albin Cassirer, Fan Yang, Kate Baumli, Sarah Henderson, Alex Novikov, Sergio Gómez Colmenarejo, Serkan Cabi, Caglar Gulcehre, Tom Le Paine, Andrew Cowie, Ziyu Wang, Bilal Piot, and Nando de Freitas. Acme: A research framework for distributed reinforcement learning. arXiv: 2006.00979v1 [cs.LG], 2020.

Wah Loon Keng and Laura Graesser. SLM lab. `https://github.com/kengz/SLM-Lab`, 2017.

Pascal Klink, Hany Abdulsamad, Boris Belousov, and Jan Peters. Self-paced contextual reinforcement learning. In *Conference on Robot Learning*, pages 513–529, 2020.

Sergey Kolesnikov. Accelerated rl. `https://github.com/catalyst-team/catalyst-rl`, 2018.

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

Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry Vetrov. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In *International Conference on Machine Learning*, 2020.

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

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

Simone Margaritelli. Pwnagotchi. https://github.com/evilsocket/pwnagotchi, 2020.

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.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

Suraj Nair, Yuke Zhu, Silvio Savarese, and Fei-Fei Li. Causal induction from visual observations for goal directed tasks. arXiv:1910.01751v1 [cs.LG], 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Neural Information Processing Systems*, pages 8024–8035, 2019.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Antonin Raffin. RL Baselines Zoo. https://github.com/araffin/rl-baselines-zoo, 2018.

Antonin Raffin. Ml framework poll for Stable-Baselines3. https://twitter.com/araffin2/status/1223310856471138306, 2020a.

Antonin Raffin. RL Baselines3 Zoo. https://github.com/DLR-RM/rl-baselines3-zoo, 2020b.

Antonin Raffin and Freek Stulp. Generalized state-dependent exploration for deep reinforcement learning in robotics. arXiv: 2005.05719v1 [cs.LG], 2020.

Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. Stable Baselines3 contrib. https://github.com/Stable-Baselines-Team/stable-baselines3-contrib, 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv:1707.06347v2 [cs.LG], 2017.

Takuma Seno. d3rlpy: A data-driven deep reinforcement learning library as an out-of-the-box tool. https://github.com/takuseno/d3rlpy, 2020.

Adam Stooke and Pieter Abbeel. rlpyt: A research code base for deep reinforcement learning in pytorch. arXiv: 1909.01500v2 [cs.LG], 2019.

Steven Wang, Sam Toyer, Adam Gleave, and Scott Emmons. The `imitation` library for imitation learning and inverse reinforcement learning. https://github.com/HumanCompatibleAI/imitation, 2020.

Jiayi Weng, Minghao Zhang, Alexis Duburcq, Kaichao You, Dong Yan, Hang Su, and Jun Zhu. Tianshou. https://github.com/thu-ml/tianshou, 2020.