Journal of Machine Learning Research 24 (2023) 1-9 Submitted 1/23; Published 8/23

skrl: Modular and Flexible Library for Reinforcement Learning

Antonio Serrano-Muñoz[1] ASERRANO@MONDRAGON.EDU
Dimitrios Chrysostomou[2] DIMI@MP.AAU.DK
Simon Bøgh[2] SB@MP.AAU.DK
Nestor Arana-Arexolaleiba[1,2] NARANA@MONDRAGON.EDU

[1] *Department of Robotics and Automation, Mondragon Unibertsitatea, Arrasate, Spain*
[2] *Department of Materials and Production, Aalborg University, Aalborg, Denmark*

Editor: Joaquin Vanschoren

Abstract

`skrl` is an open-source modular library for reinforcement learning written in Python and designed with a focus on readability, simplicity, and transparency of algorithm implementations. In addition to supporting environments that use the traditional interfaces from OpenAI Gym / Farama Gymnasium, DeepMind and others, it provides the facility to load, configure, and operate NVIDIA Isaac Gym, Isaac Orbit, and Omniverse Isaac Gym environments. Furthermore, it enables the simultaneous training of several agents with customizable scopes (subsets of environments among all available ones), which may or may not share resources, in the same run. The library's documentation can be found at `https://skrl.readthedocs.io` and its source code is available on GitHub at `https://github.com/Toni-SM/skrl`.

Keywords: Reinforcement Learning, Software, Open Source, Python, PyTorch, JAX

1. Introduction

As a Machine Learning subfield, Reinforcement Learning (RL) is a paradigm to learn, improve and generalize the decision-making capabilities of autonomous agents through interaction with their environments. Its rise is marked by three fundamental milestones: 1) The development of new learning algorithms, especially those that use artificial neural networks as approximation functions (Deep RL). 2) The development of Gym by OpenAI. It exposes a common interface for designing and standardizing environments (Brockman et al., 2016). 3) The development of benchmarking scenarios in areas such as video games and gaming, autonomous navigation, and robotics.

Particularly in robotics and autonomous systems, physics-based simulators play an essential role. Simulation enables better time management, cost reduction, and safety in safety-critical and/or complex settings (Körber et al., 2021). MuJoCo (Todorov et al., 2012) and PyBullet (Coumans and Bai, 2016–2021) are among the most widely used physics engines in robotics. These are used by the OpenAI Gym and DeepMind environments (Muldal et al., 2019; Tunyasuvunakool et al., 2020) for RL tasks.

With the release of Isaac Gym Preview (Makoviychuk et al., 2021), and recently Omniverse Isaac Gym and Isaac Orbit (Mittal et al., 2023), a GPU-based physics simulation platform from NVIDIA, a new generation of robotic simulation with tens of thousands of simultaneous environments on a single GPU has emerged. They allow researchers to easily run massive experiments using an OpenAI Gym-like API by offloading both physics simulation and neural network training onto the GPU. While Isaac Gym, Isaac Orbit and Omniverse Isaac Gym provide some examples for modeling the environment, a streamlined interface towards implementing RL algorithms in a flexible and modular way is needed.

In this work, we present `skrl`, an RL library designed with the following principles in mind: 1) modularity, leaving room for each component to be interchangeable and making it possible to create more complex systems. 2) readability, simplicity, and transparency of the algorithm implementations, which reduces the learning curve with an educational approach. 3) support for different environment interfaces and 4) simultaneous learning on NVIDIA Isaac Gym, Isaac Orbit and Omniverse Isaac Gym environments.

2. Related Work

Modularity is a desirable feature for the scalability and flexibility of a system and the reusability of its constituent components. ChainerRL (Fujita et al., 2021) and PyTorchRL (Bou and De Fabritiis, 2020) are developed around the idea of agent composability. They provide a set of building blocks for the development of new agents. rlpyt (Stooke and Abbeel, 2019), Tonic (Pardo, 2020), and MushroomRL (D'Eramo et al., 2021) also offer building blocks as configurable modules, but their designs are based on a hierarchy of inheritances involving many files and lack consistent naming in various implementations.

The code's readability, simplicity and transparency are indispensable for understanding implementations and using existing code or APIs to develop new RL methods; even more when small implementation details can significantly affect the performance of the algorithms (Engstrom et al., 2019). Many libraries encapsulate great features deep in their coding, leading to difficulties in reproducibility such as RLlib (Liang et al., 2018) or RLzoo (Ding et al., 2021). Nevertheless, there are efforts in favor of readability, simplicity and transparency. Spinning Up (Achiam, 2018), from OpenAI, was implemented with an educational approach and detailed documentation. Stable Baselines3 (Raffin et al., 2021) offers readability and simplicity over modularity, focusing on model-free, single-agent algorithms. CleanRL (Huang et al., 2022) includes all the details of the algorithm and environment in a single file, arguing that it helps researchers understand the implementation and prototype new features. Although such compact implementation facilitates the setup of simple applications, library maintenance and addition of new features remain challenging.

Almost all RL libraries support the OpenAI Gym interface for learning environments. However, the same cannot be said for DeepMind Environment, Isaac Gym, Isaac Orbit and Omniverse Isaac Gym. The last three are recent and have a slightly different interface with OpenAI Gym. In Isaac Gym's latest releases (preview 3 and 4), Isaac Orbit and Omniverse Isaac Gym, RL Games (Makoviichuk and Makoviychu, 2021) is presented as the default library to run the example environments. ElegantRl (Liu et al., 2021) offers support for Isaac Gym environments. However, it only allows working with the previous

release (preview 2), since it explicitly includes, within its source code, the original files of that preview.

3. Implementation and Features

`skrl` is an open-source modular library for RL written in Python (on PyTorch (Paszke et al., 2019) and JAX (Bradbury et al., 2018)) and designed with a focus on readability, simplicity, and transparency of algorithm implementation. In addition to supporting the OpenAI Gym / Farama Gymnasium, DeepMind and other interfaces, it allows loading and configuring NVIDIA Isaac Gym, Isaac Orbit and Omniverse Isaac Gym environments as shown in Figure 1. Furthermore, it enables agents' simultaneous training by scopes (subsets of environments among all available environments), which may or may not share resources, in the same run.

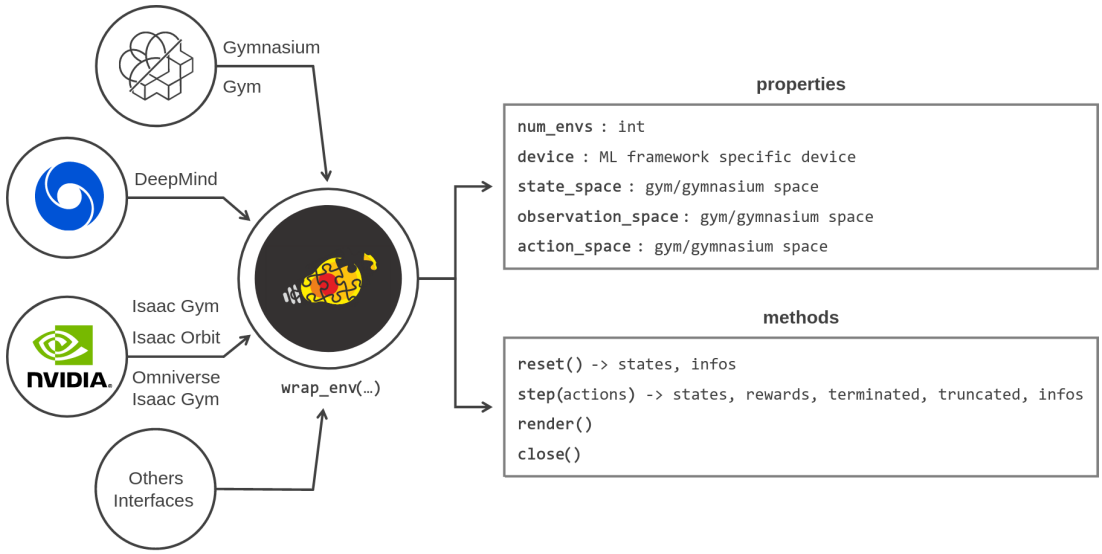

Figure 1: Wrapped environment interface based on the Gym/Gymnasium interface.

3.1 Structure and Design Concepts

The file system structure that conforms the library is designed to group the components, according to their functionality, without mixing them. This design, focused on modularity, allows a quick understanding and use of the components by the researchers. The current implementation is built on PyTorch and JAX. However, the design of the file system allows for future implementations using other deep learning libraries such as TensorFlow (Abadi et al., 2016) or Chainer (Tokui et al., 2015) among others.

The library is organized into six components (and some utilities). Except for the environments (*envs*), all other components inherit properties and methods from one (and only one) base class implemented in a common file for each group. Apart from providing a uniform interface, the base classes implement common functionalities (which are not tied to the implementation details of the algorithms), such as logging to TensorBoard (Abadi et al.,

2016) or Weights & Biases (Biewald, 2020), or saving and loading files to and from persistent storage. Focused on readability, simplicity, and transparency, each implementation within the same component is done standalone, even when two or more implementations may contain code in common.

The components that belong to `skrl` are:

- **envs**: Definition of Isaac Gym (preview 2, 3 and 4), Isaac Orbit and Omniverse Isaac Gym environment loaders. Wrappers for each supported environment type: OpenAI Gym / Farama Gymnasium, DeepMind, robosuite (Zhu et al., 2020), Isaac Gym, Isaac Orbit and Omniverse Isaac Gym.

- **memories**: Definition of generic memories that are not bound to any agent. The implementations can be used as rollout buffer or experience replay memory, for example.

- **models**: Definition of helpers for building tabular models and function approximators using artificial neural networks. In contrast to other libraries, and to put the RL system's control in the researchers' hands, `skrl` does not provide policy definitions (this practice typically hides and reduces the system's flexibility, forcing developers to deeply inspect the code to make changes). Mixins are provided to create discrete/continuous stochastic/deterministic policies within this component. In this case, the researcher is only concerned with the definition of artificial neural networks.

- **resources**: Definition of noises used by deterministic agents during the exploration stage, customized learning rate schedulers to adjust the learning rate of the optimizer between gradient steps or training epochs and input preprocessors.

- **agents**: Definition of the RL methods that compute an optimal policy. The learning and optimization algorithm is implemented within a single function in all cases. The following state-of-the-art methods are currently included as of this writing: A2C (Mnih et al., 2016), AMP (Peng et al., 2021), CEM (Szita and Lörincz, 2006), DDPG (Lillicrap et al., 2015), DQN (Mnih et al., 2015), DDQN (Van Hasselt et al., 2016), PPO (Schulman et al., 2017), Q-learning (Watkins, 1989), RPO (Rahman and Xue, 2022), SAC (Haarnoja et al., 2018), SARSA (Rummery and Niranjan, 1994), TD3 (Fujimoto et al., 2018) and TRPO (Schulman et al., 2015).

- **trainers**: Definition of the classes responsible for managing the agent's training and interaction with the environment. These definitions also allow the execution of simultaneous synchronous learning in Isaac Gym, Isaac Orbit and Omniverse Isaac Gym.

As mentioned above, a set of utilities are offered to perform, among others, the following operations: loading and post-processing of exported memory files and TensorBoard logs, downloading of trained models from Hugging Face Hub, fast model instantiators, visualization of the environment's configuration and computation of inverse kinematics for robotic manipulators in Isaac Gym and Omniverse Isaac Gym.

3.2 Simultaneous Learning by Scopes in Vectorized Environments

Isaac Gym, Isaac Orbit and Omniverse Isaac Gym simulate thousands of environments simultaneously by offering an API based on the vectorization of observations and actions. This library takes advantage of such parallelization by enabling the training and evaluation of simultaneous agents of the same or different classes. Each agent can define a working scope: a set of sub-environments among all available environments. Then, at each time step, the trainer collects the actions of each agent in their respective scopes and builds a single vector that is passed to the simulation pipeline. After simulating, the current state of observations, rewards and completed episodes are partitioned and passed back to each agent, according to its scope, to execute the learning and optimization stage.

This setup makes it possible to compare, in a single run, the performance of several agents, hyperparameters and other components. Nevertheless, given this library's modular and flexible design, it also enables sharing resources between the different agents (such as the memory, for example) that can help improve the learning process.

3.3 Documentation

The documentation is written using reStructuredText and hosted online by Read the Docs under the url `https://skrl.readthedocs.io`. Apart from the library installation steps and API details (classes, functions, parameters and return values, etc.), snippets and diagrams are also included to guide the development of new components or algorithms. In addition, a detailed description (using mathematical notation) of the implementation of the RL agents is provided. Examples, in simulation and in the real world, of use cases with their respective scripts and description of functionalities are included as well as benchmarking results.

Acknowledgments

We would like to express our gratitude for the funding and support received from NVIDIA under a collaboration agreement with the Mondragon Unibertsitatea. This study was also partially financed by H2020-WIDE SPREAD project no. 857061 "Networking for Research and Development of Human Interactive and Sensitive Robotics Taking Advantage of Additive Manufacturing – R2P2" and the H2020-ECSEL JU project no. 876852 "Verification and Validation of Automated Systems' Safety and Security - VALU3S"'. More information about this disclosure can be found on the JMLR website.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Joshua Achiam. Spinning Up in Deep Reinforcement Learning. `https://github.com/openai/spinningup`, 2018.

Lukas Biewald. Experiment tracking with weights and biases, 2020. URL `https://www.wandb.com/`. Software available from wandb.com.

Albert Bou and Gianni De Fabritiis. Pytorchrl: Modular and distributed reinforcement learning in pytorch. *arXiv preprint arXiv:2007.02622*, 2020.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

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

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2021.

Carlo D'Eramo, Davide Tateo, Andrea Bonarini, Marcello Restelli, and Jan Peters. Mushroomrl: Simplifying reinforcement learning research. *The Journal of Machine Learning Research*, 22(1):5867–5871, 2021.

Zihan Ding, Tianyang Yu, Hongming Zhang, Yanhua Huang, Guo Li, Quancheng Guo, Luo Mai, and Hao Dong. Efficient reinforcement learning development with rlzoo. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 3759–3762, 2021.

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*, 2019.

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

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. PMLR, 2018.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274): 1–18, 2022. URL `http://jmlr.org/papers/v23/21-1342.html`.

Marian Körber, Johann Lange, Stephan Rediske, Simon Steinmann, and Roland Glück. Comparing popular simulation environments in the scope of robotics and reinforcement learning. *arXiv preprint arXiv:2103.04616*, 2021.

Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. RLlib: Abstractions for distributed reinforcement learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3053–3062. PMLR, 10–15 Jul 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. *arXiv preprint arXiv:1509.02971*, 2015.

Xiao-Yang Liu, Zechu Li, Zhuoran Yang, Jiahao Zheng, Zhaoran Wang, Anwar Walid, Jian Guo, and Michael I Jordan. Elegantrl-podracer: Scalable and elastic library for cloud-native deep reinforcement learning. *arXiv preprint arXiv:2112.05923*, 2021.

Denys Makoviichuk and Viktor Makoviychu. Rl games. `https://github.com/Denys88/rl_games`, 2021.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

Mayank Mittal, Calvin Yu, Qinxi Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Ritvik Singh, Yunrong Guo, Hammad Mazhar, et al. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 2023.

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. PMLR, 2016.

Alistair Muldal, Yotam Doron, John Aslanides, Tim Harley, Tom Ward, and Siqi Liu. dm_env: A python interface for reinforcement learning environments, 2019. URL `http://github.com/deepmind/dm_env`.

Fabio Pardo. Tonic: A deep reinforcement learning library for fast prototyping and benchmarking. *arXiv preprint arXiv:2011.07537*, 2020.

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 H. Wallach, H. Larochelle,

A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. Amp: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics (TOG)*, 40(4):1–20, 2021.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL `http://jmlr.org/papers/v22/20-1364.html`.

Md Masudur Rahman and Yexiang Xue. Robust policy optimization in deep reinforcement learning. *arXiv preprint arXiv:2212.07536*, 2022.

Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*, volume 37. Citeseer, 1994.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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

István Szita and András Lörincz. Learning tetris using the noisy cross-entropy method. *Neural computation*, 18(12):2936–2941, 2006.

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

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. Chainer: a next-generation open source framework for deep learning. In *Proceedings of workshop on machine learning systems (LearningSys) in the twenty-ninth annual conference on neural information processing systems (NIPS)*, volume 5, pages 1–6, 2015.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. King's College, Cambridge United Kingdom, 1989.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, and Roberto Martín-Martín. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.