Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

# rl_reach: Reproducible reinforcement learning experiments for robotic reaching tasks

Pierre Aumjaud [a,*], David McAuliffe [b], Francisco J. Rodríguez Lera [c], Philip Cardiff [a]

[a] *School of Mechanical and Materials Engineering, University College Dublin, Belfield, Dublin 4, Ireland*
[b] *Resero Ltd, Dublin 7, Ireland*
[c] *Módulo de Investigación en Cibernética, Avenida de los Jesuitas, 24007 León, Spain*

ARTICLE INFO

*Keywords:*
Reinforcement learning
Experiments
Robotics
Artificial intelligence

ABSTRACT

Training reinforcement learning agents at solving a given task is highly dependent on identifying optimal sets of hyperparameters and selecting suitable environment input/output configurations. This tedious process could be eased with a straightforward toolbox allowing its user to quickly compare different training parameter sets. We present `rl_reach`, a self-contained, open-source and easy-to-use software package designed to run reproducible reinforcement learning experiments for customisable robotic reaching tasks. `rl_reach` packs together training environments, agents, hyperparameter optimisation tools and policy evaluation scripts, allowing its users to quickly investigate and identify optimal training configurations. `rl_reach` is publicly available at this URL: https://github.com/PierreExeter/rl_reach.

## Code metadata

| | |
|---|---|
| Current code version | v1.0 |
| Permanent link to code/repository used for this code version | https://github.com/SoftwareImpacts/SIMPAC-2021-9 |
| Permanent link to Reproducible Capsule | https://codeocean.com/capsule/4112840/tree/ |
| Legal Code License | MIT License |
| Code versioning system used | git |
| Software code languages, tools, and services used | Python 3 |
| Compilation requirements, operating environments & dependencies | Docker OR Python 3, Conda, CUDA (optional) |
| If available Link to developer documentation/manual | https://rl-reach.readthedocs.io/en/latest/index.html |
| Support email for questions | pierre.aumjaud@ucd.ie |

## 1. Context and motivations

Industrial processes have seen their productivity and efficiency increase considerably in recent decades thanks to the automation of repetitive tasks, notably with the advances in robotics. This productivity can be further improved by enabling robotic agents to solve tasks independently, without being explicitly programmed by humans.

Reinforcement Learning (RL) is a general framework for solving sequential decision-making tasks through self-learning and as such, it has found natural applications in robotics. In RL, an agent interacts with an environment by sending actions and receiving an observation – describing the current state of the world – and a reward – describing the quality of the action taken. The agent's objective is to maximise the expected cumulative return by learning a policy that will select the appropriate actions in each situation.

RL has found many successful applications, however, experiments are notoriously hard to reproduce as the learning process is highly dependent on weight initialisation and environment stochasticity [1]. In order to improve reproducibility and compare RL solutions objectively, various standard toy problems have been implemented [2–7]. A number of software suites provide training environments for

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.
* Corresponding author.
*E-mail address:* pierre.aumjaud@ucd.ie (P. Aumjaud).

https://doi.org/10.1016/j.simpa.2021.100061
Received 28 January 2021; Received in revised form 6 February 2021; Accepted 8 February 2021

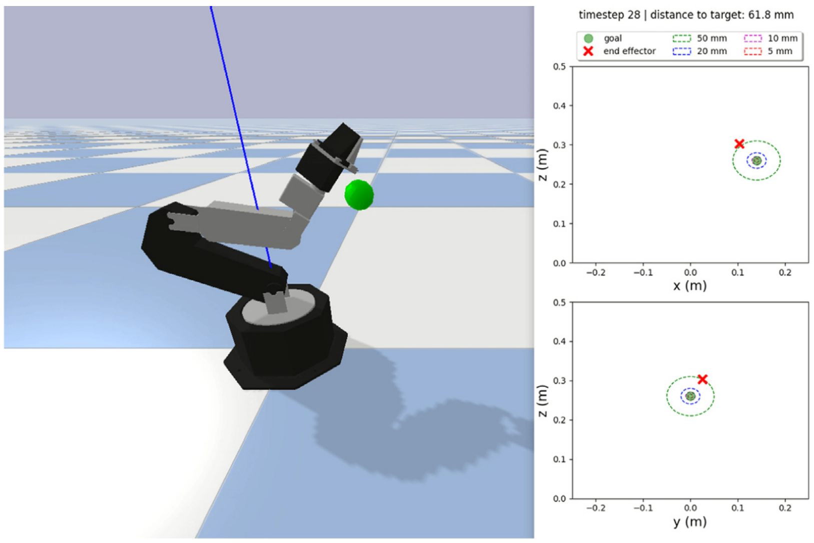

**Fig. 1.** The training environment with live visualisation of the end-effector and target position.

continuous control tasks in robotics, such as dm_control [8,9], Meta-World [10], SURREAL [11], RLBench [12], D4RL [13], robosuite [14] and robo-gym [15].

We introduce rl_reach, a self-contained, open-source and easy-to-use software package for running reproducible RL experiments applied to robotic reaching tasks. Its objective is to allow researchers to quickly investigate and identify promising sets of training parameters for a given task. rl_reach is built on top of Stable Baselines 3 [16] – a popular RL framework. The training environments are based on the WidowX MK-II robotic arm and are adapted from the Replab project [17], a benchmark platform for running RL robotics experiments. rl_reach encapsulates all the necessary elements for producing a robust performance benchmark of RL solutions for simple robotics reaching tasks. We aim to promote reproducible experimentation practice in RL research.

## 2. Functionalities and key features

The rl_reach software has been designed to quickly and reliably run RL experiments and compare the performance of trained RL agents against algorithms, hyperparameters and training environments. rl_reach's key features are:

- **Self-contained** : rl_reach packs together a widely-used RL framework – Stable Baselines 3, training environments, evaluation and hyperparameter tuning scripts (Fig. 2). In addition to its ease of usability, only a few other packages offer such self-contained code.
- **Free and open-source** : The source code is written in Python 3 and published under the permissive MIT license, with no commercial licensing restrictions. rl_reach only makes use of free and open-source projects such as the deep learning library PyTorch [18] or the physics simulator Pybullet [19]. Many RL frameworks require a paid MuJoCo license, which can be an obstacle for sharing research results. Code quality and legibility is guaranteed with standard software development tools, including the Git version control system, Pylint syntax checker, Travis continuous integration service and automated tests.
- **Easy-to-use** : A simple command-line interface is provided to train agents, evaluate policies, visualise the results and tune hyperparameters. Documentation is provided to assist end-users with the installation and main usage of rl_reach. The software and its dependencies can be installed from source with the Github repository and Conda environment provided. Portability is maximised across platforms by providing rl_reach as a Docker image, allowing it to run on any operating system that supports Docker. Finally, a reproducible code capsule is available online on the CodeOcean platform.
- **Customisable training environments** : rl_reach comes with a number of training environments for solving the reaching task with the WidowX robotic arm. These environments are easily customisable to experiment with different action, observation or reward functions. While many similar software packages exploit toy problems as benchmark tasks, rl_reach provides its users with a training environment that is closer to an industrial problem, namely reaching a target position with a robotic arm.
- **Stable Baselines inheritance** : Since rl_reach is built on top of Stable Baselines 3 [16] and its “Zoo”, it comes with the same functionalities. In particular, it supports recent model-free RL algorithms such as A2C, DDPG, HER, PPO, SAC and TD3 and automatic hyperparameter tuning with the Optuna optimisation framework [20].
- **Reproducible experiments** : Each experiment (with a unique identification number) consists of a number of runs with identical training parameters but initialised with different initialisation seeds. The evaluation metrics are averaged across all the seed runs to promote reproducible, reliable and robust experiments.
- **Straightforward benchmark** : When a trained policy is evaluated, the evaluation metrics, environment's variables and training hyperparameters are automatically logged in a CSV format. The performance of a selection of experiment runs can be visualised and compared graphically (Fig. 4).
- **Debugging tools** : It is possible to produce a 2D or 3D live plot of the end-effector and goal positions during an evaluation episode (Fig. 1), as well as a number of physical characteristics of the environment such as the end-effector and the target position, the joint's angular position, reward, distance, velocity or acceleration between the end-effector and the target (Fig. 3). It is also possible to plot the training curves for each individual seed run (Fig. 5). These plots have proven useful for debugging purposes, especially when testing a new training environment.

## 3. Impact overview

Reinforcement Learning is a recent and highly active research field, with a relatively large number of RL solutions published every year. Accurately evaluating and objectively comparing novel and existing RL

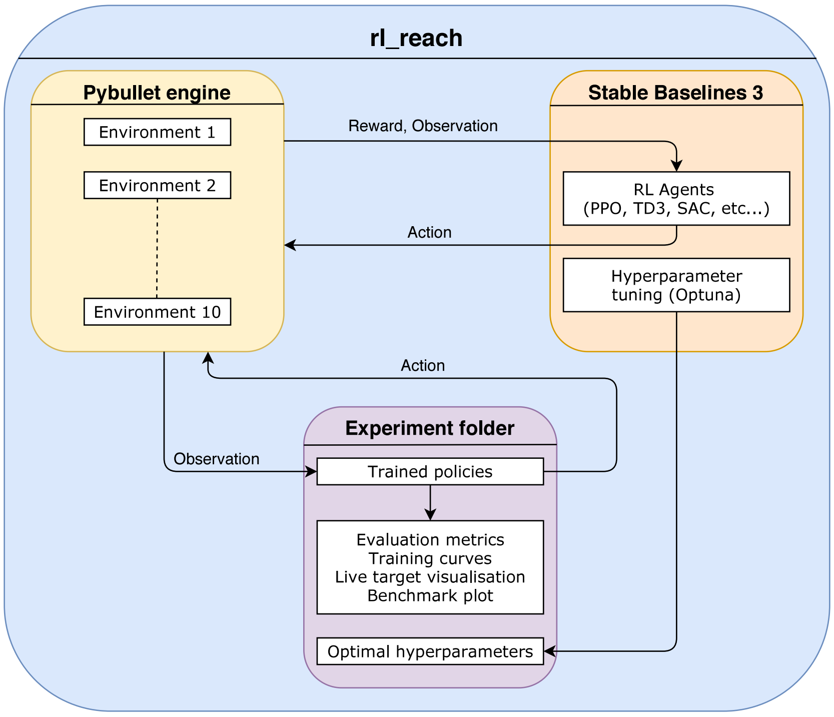

**Fig. 2.** `rl_reach`'s flowchart and components.

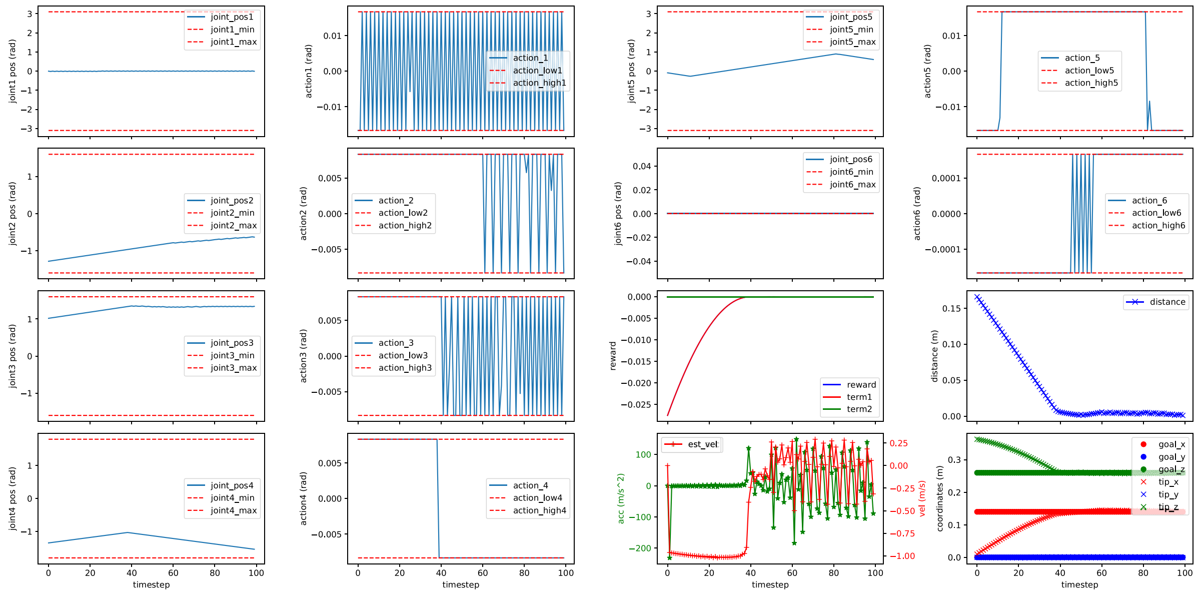

**Fig. 3.** An example of metadata plot after the evaluation of a trained policy.

approaches is crucial to ensure continued progress in the field. Reproducing RL experimental results is often challenging due to stochasticity in the training process and training environments [1]. By providing a systematic tool for carrying out reproducible RL experiments, we hope that `rl_reach` will promote better experimental practice in the RL research community and improve reporting and interpretation of results. Since `rl_reach`'s interface is straightforward, intuitive and allows for a quick graphical comparison of experiments, it can be used as an educational platform for learning the practicalities of RL training.

Training RL agents is highly dependent on a number of intrinsic (e.g. initialisation seeds, reward functions, action shape, number of time steps) and extrinsic (algorithm hyperparameters) variables. Identifying the critical parameters that control a successful training can be a daunting task. Thanks to its easily customisable learning environments and extensive logging of training parameters, `rl_reach` offers a unique solution to explore the effects of both intrinsic and extrinsic parameters on the training performance.

Finally, `rl_reach` provides learning environments designed to train a robotic manipulator to reach a target position. This task is

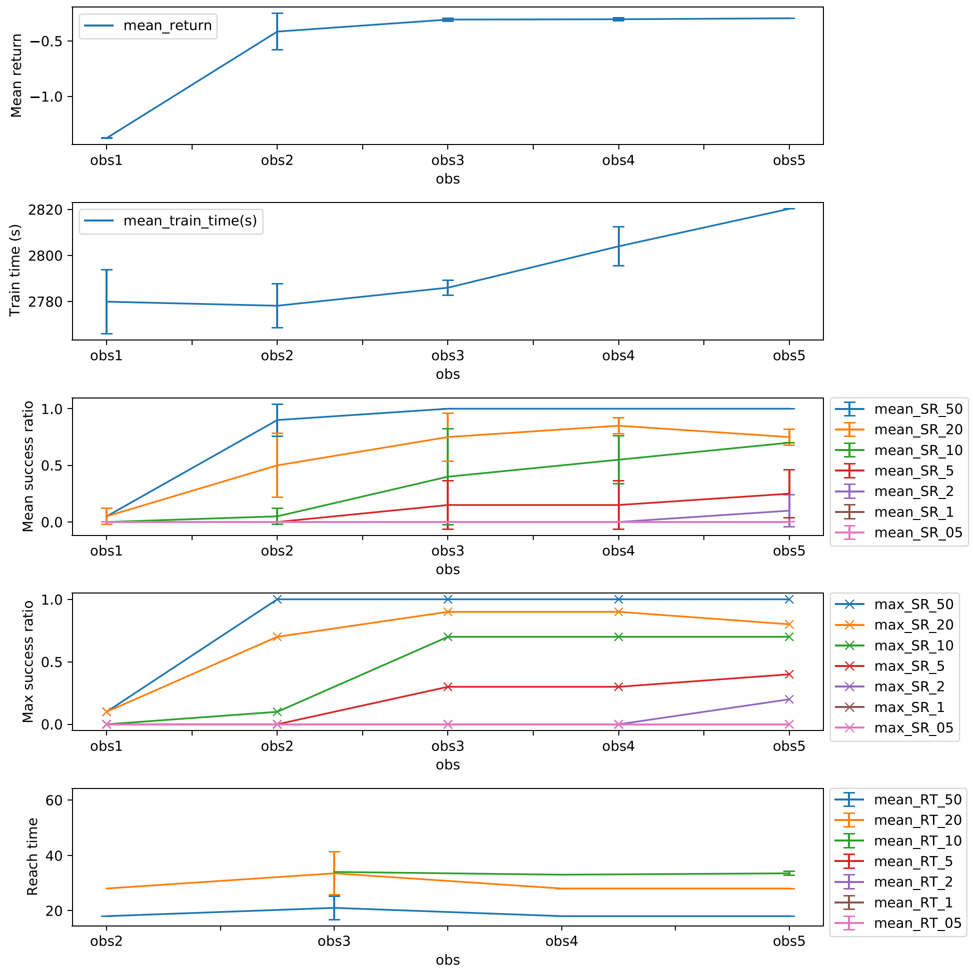

**Fig. 4.** An example of visualisation plot that compares the performance of different RL experiments.

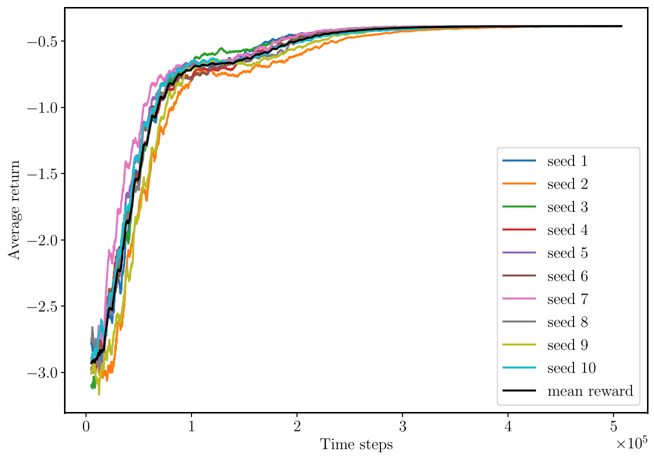

**Fig. 5.** An example of training curve plot.

more industrially-relevant than many of the toy problems considered in other benchmark packages, thus allowing straightforward transfer of RL applications from academic research to industry.

A peer-reviewed article [21] has emanated from this software where the performance of robotics RL agents trained to reach target positions is compared. The trained policies were successfully transferred from the simulated to the physical robot environment.

## 4. Conclusion and potential improvements

We chose to focus on the reaching task as it is one of the simplest tasks to solve with a robotic arm, which allows users to run experiments with relatively low computing resources, while still being industrially relevant. Moreover, the reaching task allows the user to shape the reward easily and to implement training environments with both dense and sparse rewards. However, `rl_reach` would benefit from supporting more complex and diverse manipulation tasks such as stacking, assembly, pushing or inserting. It also does not include the classic toy problems used traditionally for benchmarking RL agents. Finally, an implementation of the training environments for the physical WidowX arm would help validate the performance of policies trained in simulation.

`rl_reach` has been designed as a self-contained tool, packaging both the training environments and the RL framework Stable Baselines 3 for convenience purposes. However this does not offer the flexibility to experiment with RL algorithms that are not supported by this framework. A potential future improvement would consist in producing a modular implementation of `rl_reach` where both the training environments and the RL agents could be easily interchangeable.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgements

This Career-FIT project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 713654.

## References

[1] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, D. Meger, Deep reinforcement learning that matters, in: 32nd AAAI Conference on Artificial Intelligence, AAAI 2018, 2018, pp. 3207–3214, arXiv:1709.06560.

[2] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, OpenAI Gym, CoRR abs/1606.0. arXiv:1606.01540.

[3] M.G. Bellemare, J. Veness, The Arcade Learning Environment : An Evaluation Platform for General Agents, vol. 47, 2013, pp. 253–279, arXiv:1207.4708v2.

[4] C. Beattie, J.Z. Leibo, D. Teplyashin, T. Ward, M. Wainwright, A. Lefrancq, S. Green, A. Sadik, J. Schrittwieser, K. Anderson, S. York, M. Cant, A. Cain, A. Bolton, S. Gaffney, H. King, D. Hassabis, S. Legg, S. Petersen, DeepMind Lab, 2016, pp. 1–11. arXiv:1612.03801v2.

[5] A. Nichol, V. Pfau, C. Hesse, O. Klimov, J. Schulman, Gotta learn fast: A new benchmark for generalization in RL, arxiv, 2018, pp. 1–21. arXiv:1804.03720.

[6] K. Cobbe, C. Hesse, J. Hilton, J. Schulman, Leveraging procedural generation to benchmark reinforcement learning, arXiv arXiv:1912.01588.

[7] I. Osband, Y. Doron, M. Hessel, J. Aslanides, E. Sezener, A. Saraiva, K. McKinney, T. Lattimore, C. Szepezvari, S. Singh, B. van Roy, R. Sutton, D. Silver, H. van Hasselt, Behaviour suite for reinforcement learning, arxiv, 2019, pp. 1–19. arXiv:1908.03568.

[8] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D.D.L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, T. Lillicrap, M. Riedmiller, F. Benchmarking, DeepMind Control Suite, arXiv:1801.00690v1.

[9] Y. Tassa, S. Tunyasuvunakool, A. Muldal, Y. Doron, P. Trochim, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, Dm_control : Software and Tasks for Continuous Control, vol. 6, 2020, pp. 1–34. arXiv:2006.12983v2.

[10] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, S. Levine, Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning, arXiv (CoRL), 2019, pp. 1–18. arXiv:1910.10897.

[11] L. Fan, Y. Zhu, J. Zhu, Z. Liu, O. Zeng, A. Gupta, J. Creus-Costa, S. Savarese, L. Fei-Fei, SURREAL: Open-Source reinforcement learning framework and robot manipulation benchmark, in: A. Billard, A. Dragan, J. Peters, J. Morimoto (Eds.), Proceedings of the 2nd Conference on Robot Learning, Vol. 87 of Proceedings of Machine Learning Research, PMLR, 2018, pp. 767–782, URL http://proceedings.mlr.press/v87/fan18a.html.

[12] S. James, Z. Ma, D.R. Arrojo, A.J. Daviso, Davison, RLBench: The robot learning benchmark & learning environment, arXiv, vol. 5(2), 2019, pp. 3019–3026, arXiv:1909.12271.

[13] J. Fu, A. Kumar, O. Nachum, G. Tucker, S. Levine, D4RL: Datasets for deep data-driven reinforcement learning, arxiv, 2020, pp. 1–15. arXiv:2004.07219.

[14] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, robosuite: A Modular Simulation Framework and Benchmark for Robot Learning arXiv:2009.12293.

[15] M. Lucchi, F. Zindler, S. Mühlbacher-Karrer, H. Pichler, Robo-gym – An open source toolkit for distributed deep reinforcement learning on real and simulated robots, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2020), 2020, arXiv:2007.02753.

[16] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, Stable Baselines, GitHub repository. URL https://github.com/hill-a/stable-baselines.

[17] B. Yang, J. Zhang, V. Pong, S. Levine, D. Jayaraman, REPLAB: A Reproducible Low-Cost Arm Benchmark Platform for Robotic Learning, International Conference on Robotics and Automation (ICRA) arXiv:1905.07447.

[18] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: An imperative style, high-performance deep learning library, arXiv (NeurIPS). arXiv:1912.01703.

[19] E. Coumans, Y. Bai, PyBullet, a Python Module for Physics Simulation for Games, Robotics and Machine Learning, URL https://pybullet.org/.

[20] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: A next-generation hyperparameter optimization framework, 2019, pp. 2623–2631, http://dx.doi.org/10.1145/3292500.3330701, arXiv:1907.10902.

[21] P. Aumjaud, D. McAuliffe, F.J. Rodríguez-Lera, P. Cardiff, Reinforcement learning experiments and benchmark for solving robotic reaching tasks, 2021, pp. 318–331, arXiv:2011.05782, 10.1007/978-3-030-62579-5_22.