rl_reach: Reproducible Reinforcement Learning Experiments for Robotic Reaching Tasks

4 authors, including:

Pierre Aumjaud
University College Dublin
6 PUBLICATIONS **44** CITATIONS

Philip Cardiff
University College Dublin
133 PUBLICATIONS **871** CITATIONS

Some of the authors of this publication are also working on these related projects:

Fracture behaviour of rubber nano-modified epoxy adhesives and adhesive joints View project

CIFIL: Characterization of Railway Infrastructure using Lidar View project

rl_reach: Reproducible Reinforcement Learning Experiments for Robotic Reaching Tasks

Pierre Aumjaud[*]
University College Dublin
Dublin, Ireland
pierre.aumjaud@ucd.ie

David McAuliffe
Resero Ltd
Dublin, Ireland
david.mcauliffe@resero.io

Francisco Javier Rodríguez Lera
Universidad de León
León, Spain
fjrodl@unileon.es

Philip Cardiff
University College Dublin
Dublin, Ireland
philip.cardiff@ucd.ie

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.

Current code version	v1.0
Permanent link to code/repository	https://github.com/PierreExeter/rl_reach
Permanent link to Reproducible Capsule	https://codeocean.com/capsule/4112840/tree/v1
Legal Code License	MIT License
Code versioning system used	git
Software code language used	Python 3
Compilation requirements & dependencies	Docker OR Python 3, Conda, CUDA (optional)
Link to developer documentation/manual	https://rl-reach.readthedocs.io/en/latest/index.html
Support email for questions	pierre.aumjaud@ucd.ie

Table 1: Code metadata

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

---

[*]Corresponding author

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 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. The code metadata are given in Table 1. `rl_reach`'s key features are:

- **Self-contained** : `rl_reach` packs together a widely-used RL framework – Stable Baselines 3 [16], training environments, evaluation and hyperparameter tuning scripts (Figure 1). 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 (Figure 2).

- **Debugging tools** : It is possible to produce a 2D or 3D live plot of the end-effector and goal positions during an evaluation episode (Figure 3), 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 (Figure 4). It is also possible to plot the training curves for each individual seed run (Figure 5). These plots have proven useful for debugging purposes, especially when testing a new training environment.
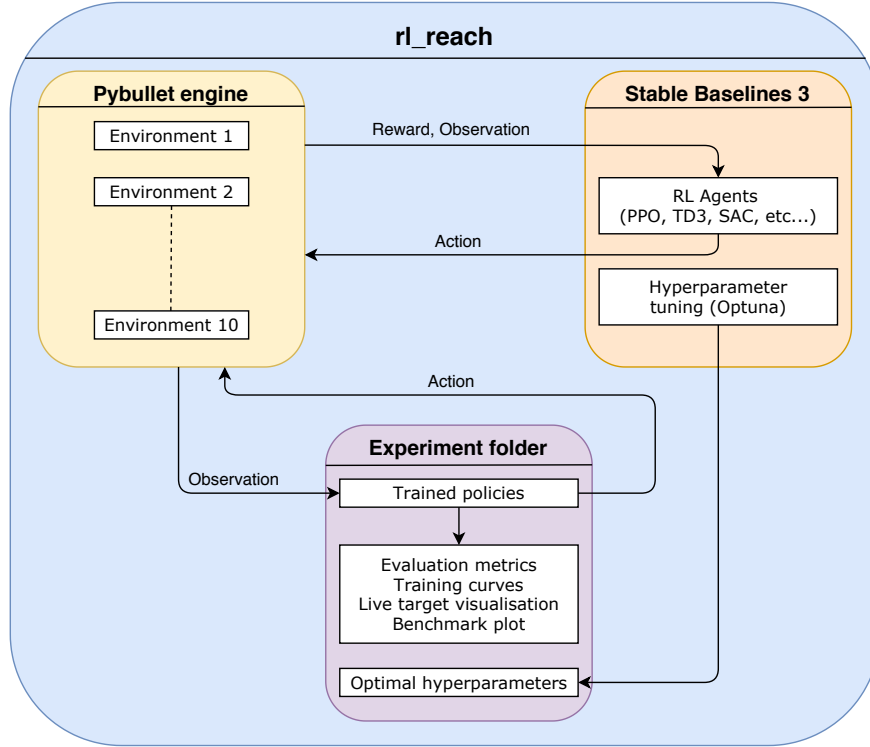

Figure 1: `rl_reach`'s flowchart and components

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 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 (eg. 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 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.
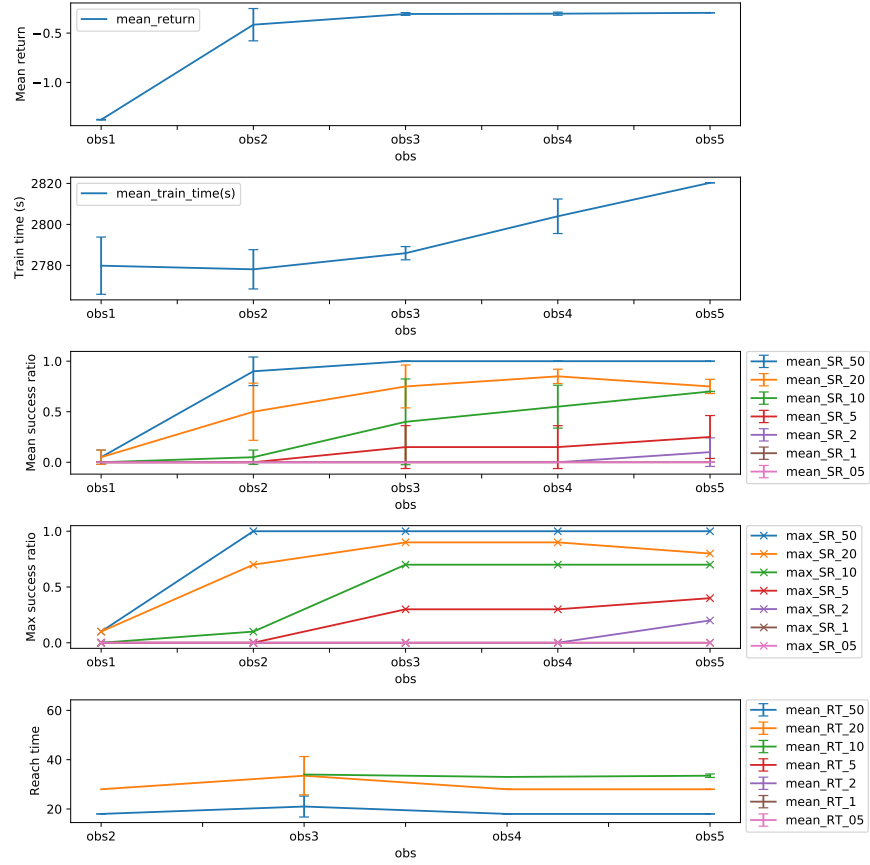
Figure 2: An example of visualisation plot that compares the performance of different RL experiments
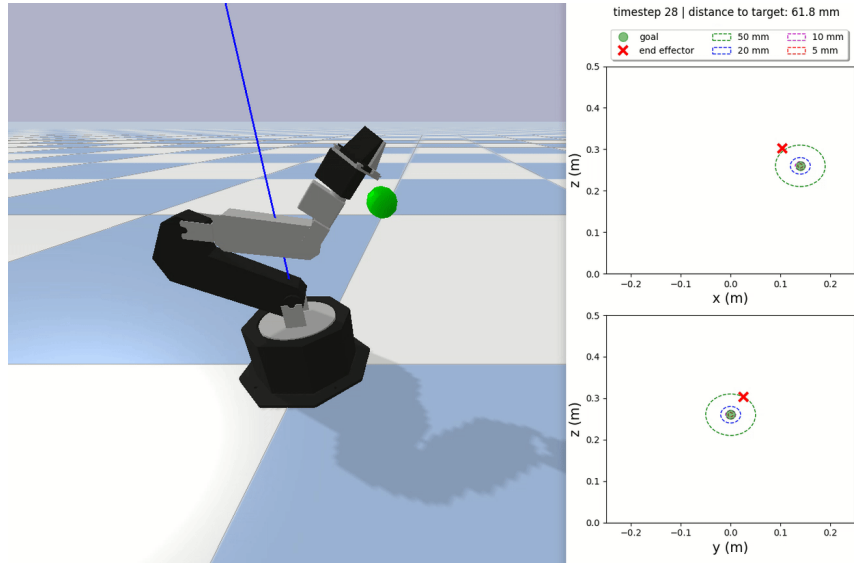

Figure 3: The training environment with live visualisation of the end-effector and target position
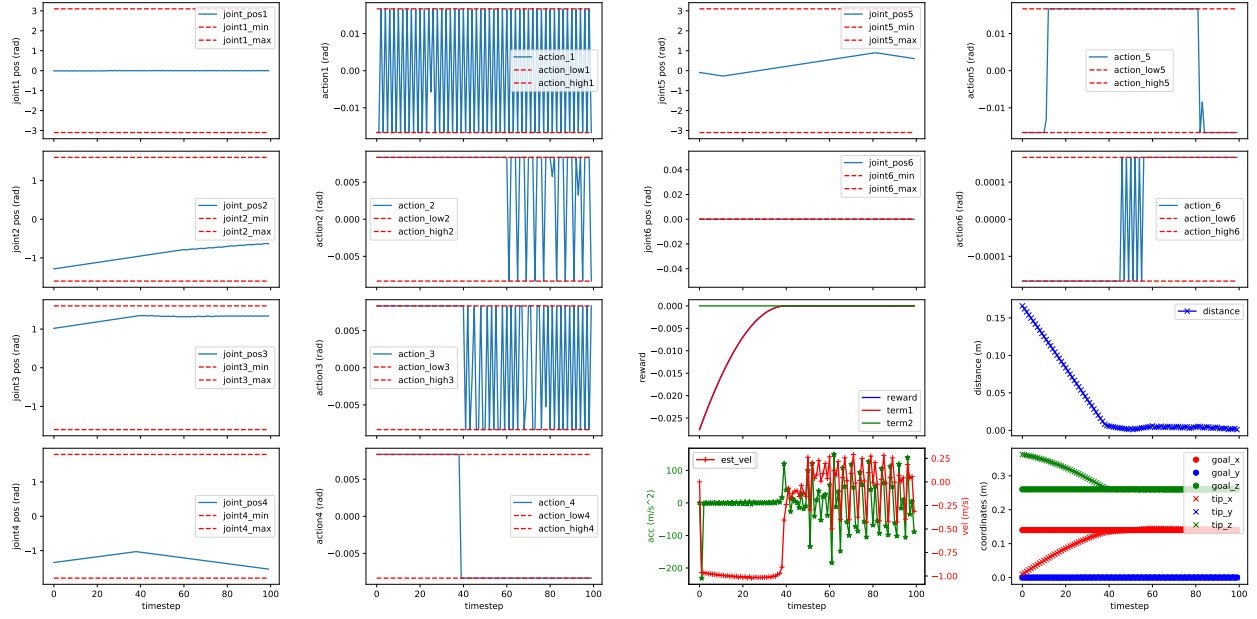
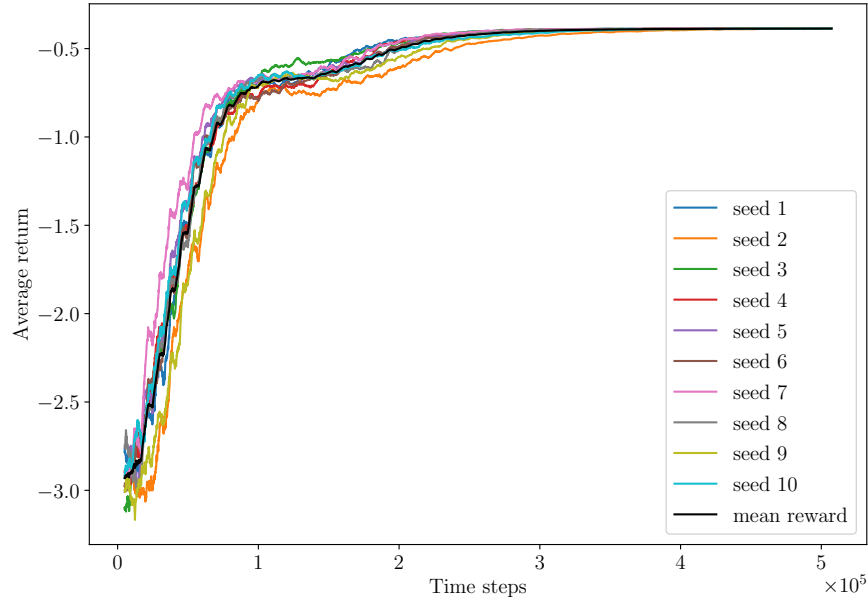Figure 4: An example of metadata plot after the evaluation of a trained policy

Figure 5: An example of training curve plot

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.

Acknowledgements

References

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

[2] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: OpenAI Gym. arXiv **1606.01540** (2016)

[3] Bellemare, M.G., Veness, J.: The Arcade Learning Environment : An Evaluation Platform for General Agents. Journal of Artificial Intelligence Research **47**, 253–279 (2013)

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

[5] Nichol, A., Pfau, V., Hesse, C., Klimov, O., Schulman, J.: Gotta Learn Fast: A New Benchmark for Generalization in RL. arXiv **1804.03720** (2018)

[6] Cobbe, K., Hesse, C., Hilton, J., Schulman, J.: Leveraging Procedural Generation to Benchmark Reinforcement Learning. In: Proceedings of the 37th International Conference on Machine Learning, pp. 2048 – 2056 (2020)

[7] Osband, I., Doron, Y., Hessel, M., Aslanides, J., Sezener, E., Saraiva, A., McKinney, K., Lattimore, T., Szepezvari, C., Singh, S., van Roy, B., Sutton, R., Silver, D., van Hasselt, H.: Behaviour Suite for Reinforcement Learning. In: International Conference on Learning Representations (2020)

[8] Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D.D.L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T., Riedmiller, M., Benchmarking, F.: DeepMind Control Suite. arXiv **1801.00690** (2018)

[9] Tassa, Y., Tunyasuvunakool, S., Muldal, A., Doron, Y., Trochim, P., Liu, S., Bohez, S., Merel, J., Erez, T., Lillicrap, T., Heess, N.: dm_control : Software and Tasks for Continuous Control. Software Impacts **6**(100022) (2020)

[10] Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., Levine, S.: Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning. In: Conference on Robot Learning (CoRL) (2019)

[11] Fan, L., Zhu, Y., Zhu, J., Liu, Z., Zeng, O., Gupta, A., Creus-Costa, J., Savarese, S., Fei-Fei, L.: SURREAL: Open-Source Reinforcement Learning Framework and Robot Manipulation Benchmark. In: 2nd Conference on Robot Learning, *Proceedings of Machine Learning Research*, vol. 87, pp. 767–782 (2018)

[12] James, S., Ma, Z., Arrojo, D.R., Davison, A.J.: RLBench: The Robot Learning Benchmark & Learning Environment. IEEE Robotics and Automation Letters **5**(2), 3019–3026 (2020)

[13] Fu, J., Kumar, A., Nachum, O., Tucker, G., Levine, S.: D4RL: Datasets for Deep Data-Driven Reinforcement Learning. arXiv **2004.07219** (2020)

[14] Zhu, Y., Wong, J., Mandlekar, A., Martín-Martín, R.: robosuite: A Modular Simulation Framework and Benchmark for Robot Learning. arXiv **2009.12293** (2020)

[15] Lucchi, M., Zindler, F., Mühlbacher-Karrer, S., Pichler, H.: robo-gym – An Open Source Toolkit for Distributed Deep Reinforcement Learning on Real and Simulated Robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2020)

[16] Raffin, A., Hill, A., Ernestus, M., Gleave, A., Kanervisto, A., Dormann, N.: Stable Baselines3 (2019). URL https://github.com/DLR-RM/stable-baselines3

[17] Yang, B., Zhang, J., Pong, V., Levine, S., Jayaraman, D.: REPLAB: A Reproducible Low-Cost Arm Benchmark Platform for Robotic Learning. In: International Conference on Robotics and Automation (ICRA) (2019)

[18] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An imperative style, high-performance deep learning library. Advances in Neural Information Processing Systems **32**, 8026–8037 (2019)

[19] Coumans, E., Bai, Y.: PyBullet, a Python Module for Physics Simulation for Games, Robotics and Machine Learning (2019). URL https://pybullet.org/

[20] Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A Next-generation Hyperparameter Optimization Framework. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 2623–2631 (2019). doi:10.1145/3292500.3330701

[21] Aumjaud, P., McAuliffe, D., Rodríguez-Lera, F.J., Cardiff, P.: Reinforcement Learning Experiments and Benchmark for Solving Robotic Reaching Tasks. In: Advances in Physical Agents II, pp. 318–331 (2021)