JaxMARL: Multi-Agent RL Environments and Algorithms in JAX

Extended Abstract

Alexander Rutherford*†
University of Oxford
arutherford@robots.ox.ac.uk

Benjamin Ellis*†
University of Oxford
benellis@robots.ox.ac.uk

Matteo Gallici*†
Universitat Politècnica de Catalunya
gallici@cs.upc.edu

Jonathan Cook*
University of Oxford
jcook@robots.ox.ac.uk

Andrei Lupu*
University of Oxford
andrei.lupu@mail.mcgill.ca

Garðar Ingvarsson*
University College London
gardarjuto@gmail.com

Timon Willi*
University of Oxford
timon.willi@gmail.com

Akbir Khan
University College London
akbir.94@gmail.com

Christian Schroeder de Witt
University of Oxford
cs@robots.ox.ac.uk

Alexandra Souly
University College London
alexandrasouly@gmail.com

Saptarashmi Bandyopadhyay
University of Maryland
saptab1@umd.edu

Mikayel Samvelyan
University College London
m.samvelyan@cs.ucl.ac.uk

Minqi Jiang
University College London
mnqjng@gmail.com

Robert Lange
Technical University Berlin
robert.t.lange@tu-berlin.de

Shimon Whiteson
University of Oxford
shimon.whiteson@cs.ox.ac.uk

Bruno Lacerda
University of Oxford
bruno@robots.ox.ac.uk

Nick Hawes
University of Oxford
nickh@robots.ox.ac.uk

Tim Rocktäschel
University College London
tim.rocktaschel@ucl.ac.uk

Chris Lu*†
University of Oxford
christopher.lu@exeter.ox.ac.uk

Jakob Foerster
University of Oxford
jakob@robots.ox.ac.uk

ABSTRACT

Benchmarks play an important role in the development of machine learning algorithms, with reinforcement learning (RL) research having been heavily influenced by the available environments. However, RL environments are traditionally run on the CPU, limiting their scalability with typical academic compute. Recent advancements in JAX have enabled the wider use of hardware acceleration to overcome these computational hurdles, enabling massively parallel RL training pipelines and environments. This is particularly useful for multi-agent reinforcement learning (MARL) research. First of all, multiple agents must be considered at each environment step, adding *computational burden*, and secondly, the *sample complexity* is increased due to non-stationarity, decentralised partial observability, or other MARL challenges. In this paper, we present JaxMARL, the first open-source code base that combines ease-of-use with GPU enabled efficiency, and supports a large number of commonly used MARL environments as well as popular baseline algorithms. When considering wall clock time, our experiments show that per-run our JAX-based training pipeline is up to 12500x faster than existing approaches. We also introduce and benchmark SMAX, a vectorised, simplified version of the popular StarCraft Multi-Agent Challenge, which removes the need to run the StarCraft II game engine. This not only enables GPU acceleration, but also provides a more flexible MARL environment, unlocking the potential for self-play, meta-learning, and other future applications in MARL. We provide code at https://github.com/flairox/jaxmarl.

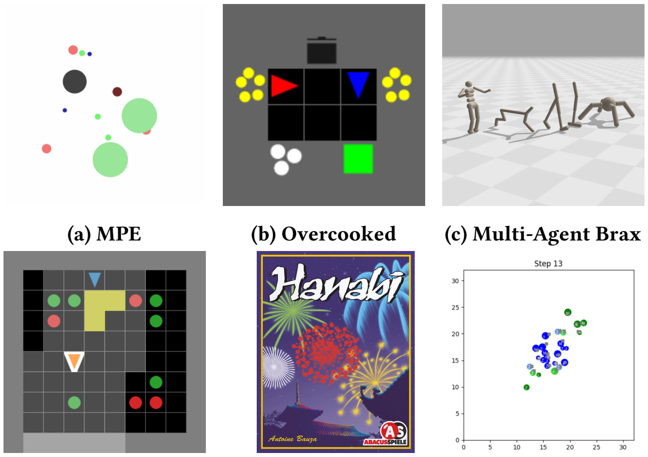
KEYWORDS

Multi-Agent Reinforcement Learning, JAX, Benchmarks

ACM Reference Format:
Alexander Rutherford[1][2], Benjamin Ellis[1][2], Matteo Gallici[1][2], Jonathan Cook[1], Andrei Lupu[1], Garðar Ingvarsson[1], Timon Willi[1], Akbir Khan, Christian Schroeder de Witt, Alexandra Souly, Saptarashmi Bandyopadhyay, Mikayel Samvelyan, Minqi Jiang, Robert Lange, Shimon Whiteson, Bruno Lacerda, Nick Hawes, Tim Rocktäschel, Chris Lu[1][2], and Jakob Foerster. 2024. JaxMARL: Multi-Agent RL Environments and Algorithms in JAX: Extended Abstract. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 3 pages.

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand.

*Core Contributor
†Equal Contribution



(a) MPE (b) Overcooked (c) Multi-Agent Brax

(d) STORM (e) Hanabi (f) SMAX

Figure 1: A selection of JaxMARL environments. We provide vectorised implementations of a wide range of environments from different MARL settings.

1 INTRODUCTION

Benchmarks play a pivotal role in the development of new single and multi-agent reinforcement learning (MARL) algorithms by defining problems, enabling comparisons, and focusing efforts. Often, data transfer between the CPU (where the environment is simulated) and the GPU (where the agents are evaluated) is a crucial bottleneck for simulation speed. Simulation speed in turn is vital for progress in reinforcement learning (RL) because RL algorithms often require a large number of environment interactions.

This problem is compounded in MARL, where non-stationarity and decentralised partial observability greatly worsen the sample complexity [1]. Hardware acceleration and parallelisation are crucial to alleviating this, but current acceleration and parallelisation methods are typically not implemented in Python, reducing their accessibility for most machine learning researchers [14, 16]. For example, the extremely efficient Hanabi library [6] from Meta-AI research is implemented in C++ and has seen relatively little adoption by the community. However, recent advances in JAX [2] have opened up new possibilities for using Python code directly with hardware accelerators, enabling the wider use of massively parallel RL training pipelines and environments.

The JAX [2] library provides composable function transformations, allowing for automatic vectorisation, device parallelisation, automatic differentiation and just-in-time (JIT) compilation, for device-agnostic optimisation. Using JAX, both the environment rollouts and model training can happen on a hardware accelerator (such as a GPU or TPU), removing the cost of data transfer between devices and allowing for significant parallelisation. Recently, PureJaxRL [8, 9] has demonstrated the power of this end-to-end JAX-based approach; running both the environment and the model training on a GPU yields a 4000x speedup over a "traditional" pipeline with a GPU-trained policy but a CPU-based environment.

Alongside the current computational issues faced by MARL researchers, recent work also highlights issues with the evaluation standards and use of benchmarks in the MARL community. In particular, MARL papers typically only test on a few domains. Of the 75 recent MARL papers analysed by [5], 50% used only one evaluation environment and a further 30% used only two. While the StarCraft Multi-Agent Challenge [SMAC, 12] and MPE [7], the two most used environments, have various tasks or maps, the lack of a standard set raises the risk of biased comparisons and incorrect conclusions. This leads to environment overfitting and unclear progress markers.

Instead, novel MARL methods should be tested on a wide range of domains to accurately evaluate their limits and enable better comparisons. The likely issue preventing this is the lack of a unified codebase and the computational burden of further evaluation.

2 JAXMARL

We present JaxMARL, a Python library that for the first time brings together JAX implementations of eight common MARL environments under one API. We additionally provide JAX implementations for five state-of-the-art algorithms, allowing for end-to-end JAX-based training pipelines in a similar fashion to PureJaxRL. By alleviating computational constraints, JaxMARL allows rapid evaluation of novel methods across a broad set of domains, and hence has the potential to be a powerful tool to address MARL's evaluation crisis. Specifically, we find that JaxMARL achieves over 12500x speedup compared to "conventional" approaches.

We also create SMAX, a JAX-based simplification of the centralised training with decentralised execution (CTDE) benchmarks SMAC [12] and SMACv2 [4]. SMAX features simplified dynamics, greater flexibility and a more sophisticated but fully-decentralised heuristic AI, while retaining the high-dimensional observation space, complex unit type interactions and procedural scenario generation that lend SMAC and SMACv2 much of their difficulty.

As illustrated in Figure 1, in addition to SMAX, our library includes the most popular environments from several MARL settings. For CTDE, we include the Multi-Agent Particle Environments (MPE) [7], and Multi-Agent Brax (MABrax). Meanwhile, for zero-shot coordination (ZSC) and ad-hoc teamplay, we include Hanabi and Overcooked. Lastly, from the general-sum literature, we include the CoinGame and Spatial-Temporal Representations of Matrix Games (STORM), a representation of matrix games as grid-world scenarios with temporally extended actions.

We additionally provide JAX implementations of Independent PPO (IPPO) [3, 13], MAPPO [17], QMIX [11], VDN [15] and Independent Q-Learning (IQL) [10], five of the most common MARL algorithms, allowing new techniques to be easily benchmarked.

3 RESULTS AND CONCLUSIONS

All of our implementations provide orders-of-magnitude speed-ups compared to their current non-JAX implementations, with specific results provided within our GitHub repository. We additionally demonstrate correspondence between our implementations and existing ones. Hardware acceleration offers important opportunities for MARL research by lowering computational barriers, increasing the speed at which ideas can be iterated, and allowing for more thorough evaluation. We hope that JaxMARL will help advance MARL by improving the ability of academic labs to conduct research with thorough, fast, and effective evaluations.

REFERENCES

[1] Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. 2002. The complexity of decentralized control of Markov decision processes. *Mathematics of operations research* 27, 4 (2002), 819–840.

[2] 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. 2018. *JAX: composable transformations of Python+NumPy programs*. http://github.com/google/jax

[3] Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip H. S. Torr, Mingfei Sun, and Shimon Whiteson. 2020. Is Independent Learning All You Need in the StarCraft Multi-Agent Challenge? https://doi.org/10.48550/arXiv.2011.09533 arXiv:2011.09533 [cs].

[4] Benjamin Ellis, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob N Foerster, and Shimon Whiteson. 2022. SMACv2: An improved benchmark for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2212.07489* (2022).

[5] Rihab Gorsane, Omayma Mahjoub, Ruan de Kock, Roland Dubb, Siddarth Singh, and Arnu Pretorius. 2022. Towards a Standardised Performance Evaluation Protocol for Cooperative MARL. *arXiv preprint arXiv:2209.10485* (2022).

[6] Hengyuan Hu and Jakob N Foerster. 2020. Simplified Action Decoder for Deep Multi-Agent Reinforcement Learning. In *International Conference on Learning Representations*. https://openreview.net/forum?id=B1xm3RVtwB

[7] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *Neural Information Processing Systems (NIPS)* (2017).

[8] Chris Lu, Jakub Kuba, Alistair Letcher, Luke Metz, Christian Schroeder de Witt, and Jakob Foerster. 2022. Discovered policy optimisation. *Advances in Neural Information Processing Systems* 35 (2022), 16455–16468.

[9] Chris Lu, Timon Willi, Alistair Letcher, and Jakob Nicolaus Foerster. 2023. Adversarial cheap talk. In *International Conference on Machine Learning*. PMLR, 22917–22941.

[10] 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. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

[11] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2020. Monotonic value function factorisation for deep multi-agent reinforcement learning. *The Journal of Machine Learning Research* 21, 1 (2020), 7234–7284.

[12] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. 2019. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043* (2019).

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

[14] Brennan Shacklett, Luc Guy Rosenzweig, Zhiqiang Xie, Bidipta Sarkar, Andrew Szot, Erik Wijmans, Vladlen Koltun, Dhruv Batra, and Kayvon Fatahalian. 2023. An Extensible, Data-Oriented Architecture for High-Performance, Many-World Simulation. *ACM Trans. Graph.* 42, 4 (2023).

[15] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. 2017. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296* (2017).

[16] 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. 2022. EnvPool: A Highly Parallel Reinforcement Learning Environment Execution Engine. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 22409–22421. https://proceedings.neurips.cc/paper_files/paper/2022/file/8caaf08e49ddbad6694fae067442ee21-Paper-Datasets_and_Benchmarks.pdf

[17] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. 2022. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems* 35 (2022), 24611–24624.