JaxMARL: Multi-Agent RL Environments and Algorithms in JAX

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

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

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 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:

*Core Contributor
†Equal Contribution
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


