MARLlib: Extending RLLib for Multi-Agent Reinforcement Learning

Siyi Hu¹, Yifan Zhong², Minquan Gao², Weixun Wang³, Hao Dong⁶, Zhihui Li¹, Xiaodan Liang², Xiaojun Chang¹, Yaodong Yang²,³
¹ReLER, AAI, University of Technology Sydney, ²Institute for AI, Peking University,
³Tianjin University, ⁴Qilu University of Technology (Shandong Academy of Sciences),
⁵Sun Yat-sen University, ⁶Center on Frontiers of Computing Studies, Peking University,
⁷Beijing Institute for General AI

ABSTRACT

Despite the fast development of multi-agent reinforcement learning (MARL) methods, there is a lack of commonly-acknowledged baseline implementation and evaluation platforms. As a result, an urgent need for MARL researchers is to develop an integrated library suite, similar to the role of RLLib in single-agent RL, that delivers reliable MARL implementation and replicable evaluation in various benchmarks. To fill such a research gap, in this paper, we propose Multi-Agent RLLib (MARLlib), a comprehensive MARL algorithm library that facilitates RLLib for solving multi-agent problems. With a novel design of agent-level distributed dataflow, MARLlib manages to unify tens of algorithms, including different types of independent learning, centralized critic, and value decomposition methods; this leads to a highly composable integration of MARL algorithms that are not possible to unify before. Furthermore, MARLlib goes beyond current work by integrating diverse environment interfaces and providing flexible parameter sharing strategies; this allows to create versatile solutions to cooperative, competitive, and mixed tasks with minimal code modifications for end users. A plethora of experiments are conducted to substantiate the correctness of our implementation, based on which we further derive new insights on the relationship between the performance and the design of algorithmic components. With MARLlib, we expect researchers to be able to tackle broader real-world multi-agent problems with trustworthy solutions. Our code and documentation are released for reference.

1 INTRODUCTION

Multi-Agent Reinforcement Learning (MARL) is a prosperous research field that has many real-world applications and holds revolutionary potential for advanced collective intelligence [6, 48, 44]. Despite its complexity [1], many existing work [2, 40, 5, 45, 24] has shown that agents are able to learn strategies that could outperform human experts and help guide human’s decision-making process in reverse. Significant as these outcomes are, the algorithm implementations are always task-specific, making it hard to compare algorithm performances, observe algorithm robustness across tasks, or use them off the shelf. Thus, developing a commonly-acknowledged baseline implementation and a unified tool suite for MARL research is in urgent demand.

While single-agent RL has witnessed successful unification for both algorithms (e.g. SpinningUp [1], Tianshou [43], RLLib [26], Dopamine [7] and Stable-Baselines series [12, 15, 33]) and environments (e.g. Gym [4]), multi-agent RL has unique challenges in building a comprehensive and high-quality library. Firstly, there exist diverse MARL algorithm pipelines. MARL algorithms diverge in learning targets such as working as a group and learning to cooperate, or competing with other agents and

https://github.com/Replicable-MARL/MARLlib
https://marllib.readthedocs.io/

Corresponding Authors. Contact: <siyi.hu@student.uts.edu.au>, <xiaojun.chang@uts.edu.au>, <yaodong.yang@pku.edu.cn>
finding a strategy that can maximize individual reward while minimizing others. Algorithms also have different restrictions on agent parameters sharing strategies, with HATRPO agents [20, 21, 42] forced to not share parameters and MAPPO capitalizing on sharing. Different styles of central information utilization such as mixing value functions (e.g. VDN [37]), sequential agent-by-agent rollout [42, 19, 29], or centralizing value function (e.g. MADDPG [27]) introduce extra challenge on algorithm learning style unification. Existing libraries such as EPyMARL [31] attempt to unify MARL algorithms under one framework by introducing independent learning, centralized critic, and value decomposition categorization, but are still lack of effort to address all the problems above. The diversity of MARL algorithms is still a huge challenge for unification.

Secondly, various multi-agent environment interfaces are mutually inconsistent, as they are originally designed to fit the task nature (e.g. asynchronous interaction are used in Hanabi, action mask are provided as additional information in SMAC [36], local observation and global state are mixed in MAgent [49]). The inconsistency hinders a directly unified agent-environment interaction processing and results in the issue of coupling between algorithm implementation and task environment; an algorithm implementation for one environment can not be directly applied to another due to interface changes. Furthermore, new MARL applications, such as robotics [8, 14], autonomous driving [50], and population biology [46], raise new challenges for implementation consistency. While PettingZoo [39] builds a collection of diverse multi-agent tasks, it is inconvenient for CTDE-based algorithm implementation as important information such as global state and action mask is not explicitly provided. Towards the inconsistency problem, other work, such as MAPPO benchmark [47], provides each environment with a unique runner script. Nevertheless, this solution creates hurdles for long-term maintenance as well as uneasiness for new task extensions.

To address the above challenges in one work, we build a new library called MARLlib based on Ray [30] and RLlib. By inheriting core advantages from RLlib and providing the following four novel features, MARLlib serves as a comprehensive platform for MARL research community.

1. **Unified algorithm pipeline with a newly proposed agent-level distributed dataflow**: To unify algorithms under diverse MARL topics and enable them to share the same learning pipeline while preserving their unique optimization logics, we construct MARLlib under the guidance of a key observation: all multi-agent learning paradigms can be equivalently transformed to the combination of single-agent learning processes; thus each agent maintains its own dataflow and optimizes the policy regardless of other agents. With this philosophy, algorithms are implemented in a unified pipeline to tackle various types of tasks, including cooperative (team-reward-only cooperation), collaborative (individual-reward-accessible cooperation), competitive (individual competition), and mixed (teamwork-based competition) tasks. We further categorize algorithms based on how they utilize central information, thereby enabling module sharing and extensibility. As shown in Figure [1]
Table 1: A comparison between current MARL libraries and our MARLlib. (x) stands for the number of available algorithms. * denotes that the benchmark has a unique framework of its own.

| Library         | Task Mode       | Supported Env | Algorithm                                           | Parameter Sharing | Async Sampling | Framework |
|-----------------|-----------------|---------------|----------------------------------------------------|-------------------|---------------|-----------|
| PyMARL [13]     | cooperative      | 1             | Independent Learning (1) Centralized Critic (1) Value Decomposition (3) | full-sharing      |               | *         |
| PyMARL2 [16]    | cooperative      | 1             | Independent Learning (1) Centralized Critic (1) Value Decomposition (9) | full-sharing      | PyMARL        |           |
| MARL-Algorithms [28] | cooperative    | 1             | Communication (1) Graph (1) Multi-task (1)          | full-sharing      |               | *         |
| EPyMARL [31]    | cooperative      | 4             | Independent Learning (3) Centralized Critic (4) Value Decomposition (2) | full-sharing      | non-sharing   | PyMARL    |
| MAlib [51]      | self-play       | 2 + PettingZoo OpeneSpiel  | Population-based (9) | full-sharing group-sharing non-sharing | ✓             | *         |
| MAPPO benchmark | cooperative      | 4             | Multi-agent PPO (1)                                | full-sharing      | non-sharing   | ✓ pytorch-a2c-ppo-ackr-gail [13] |
| MARLlib         | cooperative      | 10 + PettingZoo | Independent Learning (6) Centralized Critic (7) Value Decomposition (5) | full-sharing      | group-sharing non-sharing | ✓ Ray [38] RLlib [60] |

MARLlib manages to unify tens of algorithms with the proposed agent-level distributed dataflow, validating its effectiveness.

2. Unified multi-agent environment interface: In order to fully decouple algorithms from environments, we propose a new interface following Gym standard, with a data structure design that is compatible with most of the existing multi-agent environments, supports asynchronous agent-environment interaction, and provides necessary information to algorithms. To show the advantage of our interface design, MARLlib supports ten environments (SMAC [36], MAMuJoCo [32], GRF [22], MPE [27], LBF [31], RWARE [31], MAgent [49], Pommerman [35], MetaDrive [25], and Hanabi [3]) picked from the zoo of multi-agent tasks because of their inter-diversity, covering various task settings in MARL, including differences of task mode, observation dimension, action space property, agent-environment interaction style, etc.

3. Effective policy mapping: Flexible parameter sharing is the key to enabling one algorithm to tackle different task modes. To reduce the manual effort on regulating policies assignment, MARLlib provides three parameter sharing strategies, namely full-sharing, non-sharing, and group-sharing, by implementing the policy mapping API of RLlib. By simply changing the configuration file, users can switch among these strategies regardless of algorithms or scenarios. Therefore, although policy mapping controls the correspondence between policies and agents throughout the pipeline, it is fully decoupled from algorithms and environments, enabling further customization of sharing strategies.

4. Exhaustive performance evaluation: We run all suitable algorithms on 23 selected scenarios of five most common and diverse environment suites under four random seeds on average, which sums up to over one thousand experiments in total. The empirical results not only substantiate the correctness of MARLlib, but they also serve as a useful and trustworthy reference for the MARL research community as the comprehensiveness and fairness of comparison are guaranteed by the unified implementation approach. In addition, hyper-parameter tables are provided to ensure reproducibility. Based on these results, we derive key observations and analyze them in depth in Section 5.

With the above characteristics for a new MARL benchmark, MARLlib becomes one of the most general platforms for building, training, and evaluating MARL algorithms.

2 RELATED WORK

Building a unified platform for MARL research is meaningful yet challenging. MARL research has witnessed the development of algorithm library starting from a single task with a limited number of algorithms to more enriched tools and APIs, covering diverse tasks and advanced algorithms.
PyMARL [13] is the first and most well-known MARL library. All algorithms in PyMARL is built for SMAC [36], where agents learn to cooperate for a higher team reward. However, PyMARL has not been updated for a long time, and can not catch up with the recent progress. To address this, the extension versions of PyMARL are presented including PyMARL2 [16] and EPyMARL [31]. PyMARL2 [16] focuses on credit assignment mechanism and provide a finetuned QMIX [34] with state-of-art-performance on SMAC. The number of available algorithms increases to ten, with more code-level tricks incorporated.

EPyMARL [31] is another extension for PyMARL that aims to present a comprehensive view on how to unify cooperative MARL algorithms. It first proposed the independent learning, value decomposition, and centralized critic categorization, but is restricted to cooperative algorithms. Nine algorithms are implemented in EPyMARL. Three more cooperative environments LBF [9], RWARE [9], and MPE [27] are incorporated to evaluate the generalization of the algorithms.

All PyMARL-based libraries follow the centralized training decentralized execution setting as PyMARL. There also exist other MARL libraries that are built in different styles and serve for their unique purposes.

MARL-Algorithms [28] is a library that covers broader topics compared to PyMARL including learning better credit assignment, communication-based learning, graph-based learning, and multi-task curriculum learning. Each topic has at least one algorithm, with nine implemented algorithms in total. The testing bed is limited to SMAC.

MAPPO benchmark [47] is the official code base of MAPPO [47]. It focuses on cooperative MARL and covers four environments. It aims at building a strong baseline and only contains MAPPO.

MAlib [51] is a recent library for population-based MARL which combines game-theory and MARL algorithm to solve multi-agent tasks in the scope of meta-game.

Existing libraries and benchmarks provide good platforms for developing and comparing MARL algorithms in different environments. However, there are essential limitations to the current work.

Firstly, these work are limited in task coverage. As shown in Table 1, most existing work only support one task mode. Moreover, the number of supported environments is insufficient for researching general MARL. Secondly, existing work pay little attention to the way algorithms are organized but only focus on implementing existing work. This results in poor extensibility and a bloated code structure. Our MARLlib, a comprehensive and unified library based on Ray and RLlib, effectively solves the dilemma by providing better algorithm unification and categorization, implementing more algorithms both in quantity and in diversity, covering four task modes, supporting ten environment suites, allowing flexible parameter sharing, and being friendly to different training demands.

3 MARLlib Architecture

In this section, we mainly explain how MARLlib addresses the two major challenges, namely the diversity of MARL algorithms and inconsistency of environment interfaces, with a newly proposed agent-level distributed dataflow, a unified agent-environment interface, and effective policy mapping.

3.1 A Novel Agent-Level Distributed Dataflow for Algorithm Unification

A common framework to solve multi-agent problems is Centralized Training Decentralized Execution (CTDE), where agents maintain their own policies for independent execution and optimization, and centralized information can be utilized to coordinate agents’ update directions during the training phase. Under this framework, existing libraries split the whole learning pipeline into two stages: data sampling and model optimization. In the model optimization stage, all data sampled in the data sampling stage are available to make the training centralized. However, in this way, choosing proper data and using these data to optimize the model are coupled in the same stage. As a result, extending an algorithm to fit other task modes (e.g. both cooperative and competitive) becomes more challenging and requires redesigning the whole learning pipeline.

MARLlib addresses this issue by equivalently decomposing the original grouped dataflow into agent-level distributed dataflow. Essentially, it takes every agent in multi-agent training as an independent
Figure 2: Agent-level distributed dataflow of MARLlib. Observed data refer to data sampled from environments, such as rewards or global states. Predicted data refer to data generated by agents, such as Q values or chosen actions. Postprocessing refers to the data sharing process. Each agent maintains its own learning pipeline where collected data are used to optimize the policy of agent itself. Therefore, the dataflow is agent-level distributed. There are three types of dataflow, namely independent learning, centralized critic, and value decomposition [31] that are categorized according to the central information utilization style. Independent learning algorithms (e.g. IQL [38]) is inherently a learning procedure with distributed dataflow for each agent and the information-sharing process is skipped, as shown in (a). For centralized critic algorithms (e.g. MAPPO [47]), central information, including both observed data and predicted data, is collected and shared in the postprocessing function before entering the training stage, as shown in (b). In the value decomposition category (e.g. FACMAC [32]), predicted data from all agents must be shared, whereas the observed data is optional, depending on the mixing function the algorithm adopts. The corresponding dataflow is shown in (c).

Moreover, while all CTDE-based algorithms share similar agent-level dataflow in general, they still have unique data processing logic. Inspired by EPyMARL [31], we further classify algorithms into independent learning, centralized critic, value decomposition categories to enable module sharing and extensibility. Independent learning algorithms let agents learn independently; centralized critic algorithms optimize the critic with shared information, which then guides the optimization of decentralized actors; value decomposition algorithms learn a joint value function as well as its decomposition into individual value functions, which agents then employ to select actions during execution. According to their algorithmic properties, we implement suitable data sharing strategies in postprocessing phase, as illustrated in Figure 2.

Therefore, still preserving the unique properties of all algorithms, the agent-level distributed dataflow unifies diverse learning paradigms. Our implementation approach shows its value in unifying algorithms following CTDE in a single pipeline, which is capable of handling all task modes and achieving similar performance to the original implementation.
3.2 A Universal Interface for Agent-environment Interaction

In RL, Gym interface, \texttt{obs, reward, done, info}, is universally used. However, this standardized interface cannot be trivially extended to MARL. In MARL, multiple agents coexist, each having its own experience data. Extra information is sometimes available, such as action mask, global state. Depending on the task mode, the reward may be a scalar or a dictionary. Moreover, the agents may not interact with the environments synchronously, posing another challenge. Towards these issues, MARLlib unifies the multi-agent interface from two aspects.

Firstly, MARLlib unifies the interface data structure. MARLlib proposes a new interface following Gym standard, \texttt{obs, reward, done, info}, that is compatible with diverse multi-agent task settings in practice, covering both data structure and task modes. As illustrated in Figure 3c in MARLlib, the observation returned from the environment is a dictionary with three keys: \texttt{observation, action_mask, global_state}. This design satisfies most circumstances and is compatible with RLLib’s data processing logic. Other observation-related information is included in \texttt{info}. \texttt{reward} is a dictionary with agent id as the key. To accommodate cooperative tasks, scalar team reward is transformed to dictionary structure by copying it for agent number times. \texttt{done} is a dictionary containing a single key "\texttt{__all__}”, which is true only when all agents are terminated.

Secondly, MARLlib supports both synchronous and asynchronous agent-environment interaction. Existing MARL libraries before MARLlib, such as PyMARL (Figure 3b), do not support asynchronous sampling and mainly focus on the synchronous cases. However, asynchronous agent-environment interaction is common in multi-agent tasks like Go and Hanabi. MARLlib supports synchronous and asynchronous agent-environment interaction, thanks to Ray/RLLib’s flexible data collection mechanism: the data are collected and stored with agent id. Only when we receive the terminal signal \texttt{done} will all data be returned for subsequent usage. This sampling process is illustrated in Figure 3c.

3.3 Effective Policy Mapping

In multi-agent scenarios, a proper parameter sharing strategy can improve the algorithm’s performance. Important as it is, most of existing work support insufficient sharing modes and the implementation is repetitious — MAPPO benchmark rewrites everything for shared and separated settings, while EPyMARL repeats model structures to support both. In MARLlib, we support full-sharing (all agents share parameters), non-sharing (no agents share parameters), and group-sharing (agents within the same group share parameters) of parameters by implementing the policy mapping API of RLLib. Intuitively, it maps the virtual policies of agents to physical policies that are actually maintained, used, and optimized. Agents mapped to the same physical policy share parameters. As policy mapping is transparent to agents, they actually sample data and do optimization with the physical policies. Therefore, different types of parameter sharing can take place without affecting algorithm implementation. In practice, we only need to maintain a policy mapping dictionary for every environment with all the relevant information to support multiple sharing modes. More customized parameter sharing strategy can be realized by revising the policy mapping API to suit the needs.
MARLlib allows users to regulate the whole training pipeline by customizing configuration files, which is clean for usage and convenient for experimental report. For a complete MARL pipeline, configurations of four different aspects are supported, including task configuration, algorithm configuration, agent model configuration, and basic training settings for Ray and RLlib. The contents and locations of them are shown in Figure 4. Task configuration maintains the key parameters of the task scenario when generating from environment engine. For instance, the level of the enemy in SMAC can be decided by users. Algorithm configuration is in charge of the hyper-parameters of the algorithm’s learning procedure. In addition, tricks can also be turned on or off by simply changing the corresponding key values. As MARLlib aims at building a general benchmark, tricks that are only applicable for one task but not for others are not incorporated, which guarantees that MARLlib’s algorithm configuration is valid for diverse multi-agent tasks. Agent model configuration is responsible for constructing the agent learning unit: a neural network. Basic training settings for Ray and RLlib control the computation resource allocation. By configuring the four aspects, the whole pipeline is determined and the training can be launched by simply running the entrance script main.py. After the training begins, all configurations will be recorded. Referring to the record file can help reproduce exactly the same performance and conduct a fair experiment.

5 Benchmarking Results and Analysis

In this section, we evaluate 17 algorithms on 23 tasks of five MARL testing beds including SMAC [36], MPE [27], GRF [22], MAMuJoCo [32], and MAgent [49], which are chosen for their popularity in MARL research and their diversity in task modes, observation shape, additional information, action space, sparse or dense reward, and homogeneous or heterogeneous agent types. We report the mean reward of experiments under four random seeds, which sums up to over one thousands experiments in total. Experiments results are shown in Table 2 and Figure 5. Based on these results, we substantiate the quality of implementation and provide insightful analysis.

5.1 Quality of Implementation

To show the correctness of MARLlib, we compare the performances of MARLlib implementation on SMAC to those reported by EPyMARL with the important hyper-parameters kept the same. Results of EPyMARL consume 40 million steps on on-policy algorithms and four million on off-policy algorithms. MARLlib only consumes half of them respectively, as we find that it is enough for training to converge. Even with fewer training steps, we match most of the performances reported by
Centralized critic is better at learning diverse yet coordinated behaviors. In a multi-agent task, agents can take different roles, and their behaviors are expected to be role-specific [17, 41]. Centralized critic is suitable for these tasks since local observations and global information are both well utilized.

Table 2: Algorithm performances (in reward) for cooperative tasks. Both discrete control tasks (MPE, SMAC, GRF) and continuous control tasks (MAMuJoCo) are covered. Among four environment suites, SMAC has two rows for each scenario. The first row is the performances reported by EPyMARL, and the second row is performances with MARLlib. For other environments, only MARLlib performances are included. * represents no data reported. Dark cells indicate the top two performances on each scenario.

| Env | Scenario | Independent Learning | Value Decomposition |
|-----|----------|----------------------|---------------------|
|     |          | IQL | IPG | IAC2 | ITIPPO | IPPO | MAAC2 | COMA | MATPPO | MAPPO | VDN | QMIX | VDA2C | VDPPO |
| SMAC | 2x vs 1x | 15.18 | 9.72 | 10.34 | 10.18 | 16.14 | 13.94 | 13.94 | 14.32 | 14.32 | 15.18 | 14.78 | 9.48 | 9.48 |
|      | 3x5r     | 15.18 | 9.72 | 10.34 | 10.18 | 16.14 | 13.94 | 13.94 | 14.32 | 14.32 | 15.18 | 14.78 | 9.48 | 9.48 |
|      | 13x5r    | -75.36 | -30.72 | -30.72 | -30.72 | -47.57 | -47.57 | -47.57 | -47.57 | -47.57 | -75.36 | -47.57 | -30.72 | -30.72 |
|      | MMM2     | 15.18 | 9.72 | 10.34 | 10.18 | 16.14 | 13.94 | 13.94 | 14.32 | 14.32 | 15.18 | 14.78 | 9.48 | 9.48 |
|      | 3x vs 5x | 15.18 | 9.72 | 10.34 | 10.18 | 16.14 | 13.94 | 13.94 | 14.32 | 14.32 | 15.18 | 14.78 | 9.48 | 9.48 |

EPyMARL, as shown in Table 2. For all performance pairs available to compare, MARLlib attains similar results on 63% of them (the reward difference is less than 1.0), achieves superior results on 25% of them, and appears inferior on the rest 12%. Since every algorithm exhibits expected performances and for generality and stability we do not rely on task-specific tricks, these experimental results substantiate the correctness of implementation. In this table, we also report for the first time the performances of five algorithms on SMAC and MPE, twelve on GRF, and ten on MAMuJoCo for community reference. All the experiments are reproducible, as we provide learning curves and complete training configurations in our code repository.

5.2 Performance Inheritance

Empirically, we find that developing MARL algorithm based on strong RL algorithm is a wise choice. For instance, PPO is primarily used in single-agent RL because of its better empirical performance than vanilla PG and A2C. This superiority affects the performance of their multi-agent counterparts — MAPPO and VDPPO surpass MAA2C and VDA2C in most scenarios. Another evidence that corroborates this conclusion is the robustness of value iteration methods. Compared to policy-gradient methods, value iteration-based algorithms are less hyperparameter-sensitive and more sample efficient. The multi-agent version of Q learning like IQL, VDN, and QMIX also inherits this advantage and shows robust performance in most scenarios like SMAC and MPE.

5.3 Suitable Scenarios of MARL Algorithms

From Table 2 we find algorithms of different categorization show superiority on specific tasks that share similar task patterns.

Independent learning is effective when the central information is not necessary. While coordination among agents is essential for MARL algorithms and independent learning is theoretically suboptimal, existing work [10] has pointed out that independent learning can surpass other algorithms. In Table 2, we find that independent learning algorithms are better than its centralized critic counterparts in scenarios like simple_spread and pass_and_shoot, where agents are expected to behave similarly and central information is not necessary for policy optimization. By the same logic, without a global view, independent learning fails to solve coordination tasks such as simple_spread and simple_reference.

Centralized critic is better at learning diverse yet coordinated behaviors. In a multi-agent task, agents can take different roles, and their behaviors are expected to be role-specific [17, 41]. Centralized critic is suitable for these tasks since local observations and global information are both well utilized.

https://github.com/Replicable-MARL/MARLlib/tree/main/results
Figure 5: Reward curves of eight mixed scenarios (agents compete in group) in MAgent (a-d) and MPE (e-h). Different styles of curves stand for different agent groups. The reward curves of competing groups show a dynamic balance during the learning procedure, while the balance point depends on both algorithms and tasks.

Good examples are MAPPO and MATRPO. MAPPO is a representative algorithm of centralized critic that is robust on most cooperative tasks in SMAC, MPE, and GRF. MATRPO is the centralized version of ITRPO that outperforms other algorithms in MAMuJoCo. MAPPO and MATRPO are strong baselines on cooperative tasks where agents have diverse behaviors.

Value decomposition dominates the popular cooperative benchmarks, except for two cases. The first case is continuous control. Well-known algorithms of value decomposition like VDN and QMIX are not suitable for continuous control tasks, and VDA2C and VDPPO are inferior compared to other algorithms. The second case is a long-term planning problem with a sparse reward function like in GRF. Empirically, the performances of value decomposition methods are significantly worse than algorithms of other categories such as ITRPO and MAPPO. We identify two primary reasons: 1) value iteration used by VDN and QMIX prefers a dense reward function; 2) the mixer can hardly decompose a Q function close to zero. Except for these two cases, value decomposition algorithms achieve robust performance with the best sample efficiency.

5.4 Algorithm Evaluation in Mixed Scenarios

Benchmarking algorithms in mixed tasks is challenging. Agents in mixed tasks behave both cooperatively (with teammates) and competitively (to their opponents). It is hard to justify which algorithm is better based on the reward gained as the policies are always in dynamic balance: when one policy is better optimized, the performances of its opponents’ policies are degraded.

Under mixed task mode, algorithms can be evaluated by the summed reward of all different policies. One policy optimization forces competitive policies to get a higher reward. Therefore, the higher the summed reward, the better the algorithms (Figure 5[a-d]). However, there are exceptions (Figure 5[e-h]). The summed reward is a constant value or around a constant value and policies quickly reach equilibrium with mirrored learning curves between competitive policies as a significant pattern. A fair and general criterion to evaluate algorithms on constant-sum tasks is still an active research direction.

6 Conclusion

In this paper, we introduce a new MARL library, MARLlib, that covers broad topics and tasks in multi-agent research. MARLlib unprecedentedly unifies diverse algorithm learning paradigms and multi-agent environment interfaces with newly proposed agent-level distributed dataflow, interface unification methods, and flexible parameter sharing strategies thanks to policy mapping. Thousands of experiments are conducted to validate the correctness of implementation and derive new insights on the relationship between performance and algorithmic components. Overall, it serves as a comprehensive and general platform for algorithm training, evaluation, and comparison, and can benefit applications.
such as large-scale multi-agent systems or MARL education. We strongly recommend readers to have a look at our code and documentation, where more details are provided for reference. Future work includes extending MARLlib to algorithms that do not conform to CTDE framework, such as communication-based methods. We hope MARLlib can benefit the MARL research community and facilitate future research.

REFERENCES

[1] Joshua Achiam. Spinning Up in Deep Reinforcement Learning. 2018.
[2] Bowen Baker, Ingrid Kanitscheider, Todor M. Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. In ICLR, 2020.
[3] Nolan Bard, Jakob N Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, et al. The hanabi challenge: A new frontier for ai research. Artificial Intelligence, 2020.
[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.
[5] Noam Brown and Tuomas Sandholm. Superhuman ai for multiplayer poker. Science, 365 (6456):885–890, 2019.
[6] Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. IEEE Transactions on Systems, Man, and Cybernetics, 38(2):156–172, 2008.
[7] Pablo Samuel Castro, Subhodeep Moitra, Carles Gelada, Saurabh Kumar, and Marc G. Bellemare. Dopamine: A Research Framework for Deep Reinforcement Learning. 2018.
[8] Yuanpei Chen, Yaodong Yang, Tianhao Wu, Shengjie Wang, Xidong Feng, Jiechuang Jiang, Stephen Marcus McAleer, Hao Dong, Zongqing Lu, and Song-Chun Zhu. Towards human-level bimanual dexterous manipulation with reinforcement learning. arXiv preprint arXiv:2206.08686, 2022.
[9] Filippos Christianos, Lukas Schäfer, and Stefano Albrecht. Shared experience actor-critic for multi-agent reinforcement learning. In NIPS, 2020.
[10] Christian Schroeder de Witt, Tarun Gupta, Denys Makoviychuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? arXiv preprint arXiv:2011.09533, 2020.
[11] Xiaotie Deng, Yuhao Li, David Henry Mguni, Jun Wang, and Yaodong Yang. On the complexity of computing markov perfect equilibrium in general-sum stochastic games. arXiv preprint arXiv:2109.01795, 2021.
[12] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines, 2017.
[13] Python MARL Framework. URL [https://github.com/oxwhirl/pymarl].
[14] Yiran Geng, Boshi An, Haoran Geng, Yuanpei Chen, Yaodong Yang, and Hao Dong. End-to-end affordance learning for robotic manipulation. arXiv preprint arXiv:2209.12941, 2022.
[15] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable baselines, 2018.
[16] Jian Hu, Siyang Jiang, Seth Austin Harding, Haibin Wu, and Shih wei Liao. Rethinking the implementation tricks and monotonicity constraint in cooperative multi-agent reinforcement learning. 2021.
[17] Siyi Hu, Chuanlong Xie, Xiaodan Liang, and Xiaojun Chang. Policy diagnosis via measuring role diversity in cooperative multi-agent rl. In ICML, 2022.

[18] Ilya Kostrikovk. URL https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail

[19] Jakub Grudzien Kuba, Muning Wen, Linghui Meng, Haifeng Zhang, David Mguni, Jun Wang, Yaodong Yang, et al. Settling the variance of multi-agent policy gradients. Advances in Neural Information Processing Systems, 34:13458–13470, 2021.

[20] Jakub Grudzien Kuba, Ruqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning. In ICLR, 2022.

[21] Jakub Grudzien Kuba, Xidong Feng, Shiyao Ding, Hao Dong, Jun Wang, and Yaodong Yang. Heterogeneous-agent mirror learning: A continuum of solutions to cooperative marl. arXiv preprint arXiv:2208.01682, 2022.

[22] Karol Kurach, Anton Raichuk, Piotr Stanczyk, Michał Zajac, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, and Sylvain Gelly. Google research football: A novel reinforcement learning environment. In AAAI, 2020.

[23] Marc Lanctot, Edward Lockhart, Jean-Baptiste Lespiau, Vinicius Zambaldi, Satyaki Upadhyay, Julien Pérolat, Sriram Srinivasan, Finbarr Timbers, Karl Tuyls, Shayegan Omidshafiei, et al. Openspiel: A framework for reinforcement learning in games. arXiv preprint arXiv:1908.09453, 2019.

[24] Minne Li, Zhiwei Qin, Yan Jiao, Yaodong Yang, Jun Wang, Chenxi Wang, Guobin Wu, and Jieping Ye. Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning. In The World Wide Web Conference, pp. 983–994, 2019.

[25] Quanyi Li, Zhenghao Peng, Lan Feng, Qihang Zhang, Zhenghai Xue, and Bolei Zhou. Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2022.

[26] 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 ICML, 2018.

[27] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In NIPS, 2017.

[28] MARL-Algorithms. URL https://github.com/starry-sky6688/MARL-Algorithms

[29] Linghui Meng, Muning Wen, Yaodong Yang, Chenyang Le, Xiyun Li, Weinan Zhang, Ying Wen, Haifeng Zhang, Jun Wang, and Bo Xu. Offline pre-trained multi-agent decision transformer: One big sequence model conquers all starcraftii tasks. arXiv preprint arXiv:2112.02845, 2021.

[30] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, et al. Ray: A distributed framework for emerging {AI} applications. In 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18), 2018.

[31] Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In NIPS Track on Datasets and Benchmarks, 2021.

[32] Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienney, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. Facmac: Factored multi-agent centralised policy gradients. NIPS, 2021.

[33] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. JMLR, 2021.
[34] Tabish Rashid, Mikayel Samvelyan, Christian Schröder de Witt, Gregory Farquhar, Jakob N. Foerster, and Shimon Whiteson. QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning. In ICML, 2018.

[35] Cinjon Resnick, Wes Eldridge, David Ha, Denny Britz, Jakob N. Foerster, Julian Togelian, Kyunghyun Cho, and Joan Bruna. Pommerman: A multi-agent playground. In Jichen Zhu (ed.), Joint Proceedings of the AIIDE 2018 Workshops co-located with 14th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2018), 2018.

[36] Mikayel Samvelyan, Tabish Rashid, Christian Schröder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob N. Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. In AAMAS, 2019.

[37] Jianyu Su, Stephen Adams, and Peter A Beling. Value-decomposition multi-agent actor-critics. In AAAI, 2021.

[38] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In ICML, 1993.

[39] Justin K. Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis S. Santos, Clemens Dieffendahl, Caroline Horsch, Rodrigo Perez-Vicente, Niall L. Williams, Yashas Lokesh, and Praveen Ravi. Pettingzoo: Gym for multi-agent reinforcement learning. In Marc’ Aurelio Ranzato, Alina Beygelzimer, NIPS, 2021.

[40] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. Nature, 575(7782):350–354, 2019.

[41] Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. RODE: learning roles to decompose multi-agent tasks. In ICLR, 2021.

[42] Muning Wen, Jakub Grudzien Kuba, Runji Lin, Weinan Zhang, Ying Wen, Jun Wang, and Yaodong Yang. Multi-agent reinforcement learning is a sequence modeling problem. arXiv preprint arXiv:2205.14953, 2022.

[43] Jiayi Weng, Huayu Chen, Dong Yan, Kaichao You, Alexis Duburcq, Minghao Zhang, Hang Su, and Jun Zhu. Tianshou: a highly modularized deep reinforcement learning library. CoRR, 2021.

[44] Yaodong Yang and Jun Wang. An overview of multi-agent reinforcement learning from game theoretical perspective. CoRR, abs/2011.00583, 2020.

[45] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. Mean field multi-agent reinforcement learning. In International Conference on Machine Learning, pp. 5571–5580. PMLR, 2018.

[46] Yaodong Yang, Lantao Yu, Yiwei Bai, Ying Wen, Weinan Zhang, and Jun Wang. A study of AI population dynamics with million-agent reinforcement learning. In AAMAS, 2018.

[47] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of mappo in cooperative, multi-agent games. arXiv preprint arXiv:2103.01955, 2021.

[48] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. Handbook of Reinforcement Learning and Control, pp. 321–384, 2021.

[49] Lianmin Zheng, Jiacheng Yang, Han Cai, Ming Zhou, Weinan Zhang, Jun Wang, and Yong Yu. Magent: A many-agent reinforcement learning platform for artificial collective intelligence. In AAAI, 2018.
[50] Ming Zhou, Jun Luo, Julian Villella, Yaodong Yang, David Rusu, Jiayu Miao, Weinan Zhang, Montgomery Alban, Iman Fadakar, Zheng Chen, Aurora Chongxi Huang, Ying Wen, Kimia Hassanzadeh, Daniel Graves, Dong Chen, Zhengbang Zhu, Nhat Nguyen, Mohamed Elsayed, Kun Shao, Sanjeevan Ahilan, Baokuan Zhang, Jiannan Wu, Zhengang Fu, Kasra Rezaee, Peyman Yadmeall, Mohsen Rohani, Nicolas Perez Nieves, Yihan Ni, Seyedershad Banijamali, Alexander Cowen Rivers, Zheng Tian, Daniel Palenicek, Haitham bou Ammar, Hongbo Zhang, Wulong Liu, Jianye Hao, and Jun Wang. Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving, 2020.

[51] Ming Zhou, Ziyu Wan, Hanjing Wang, Muning Wen, Runzhe Wu, Ying Wen, Yaodong Yang, Weinan Zhang, and Jun Wang. Malib: A parallel framework for population-based multi-agent reinforcement learning. *arXiv preprint arXiv:2106.07551*, 2021.