M2RL: A Multi-player Multi-agent Reinforcement Learning Framework for Complex Games

Tongtong Yu[1] , **Chenghua He**[1] and **Qiyue Yin**[1,2*]
[1]Institute of Automation, Chinese Academy of Sciences, China
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences, China
xtina988@hotmail.com, hechenghua23@mails.ucas.ac.cn, qyyin@nlpr.ia.ac.cn

Abstract

Distributed deep reinforcement learning (DDRL) has gained increasing attention due to the emerging requirements for addressing complex games like Go and StarCraft. However, how to effectively and stably train bots with asynchronous and heterogeneous agents cooperation and competition for multiple players under multiple machines (with multiple CPUs and GPUs) using DDRL is still an open problem. We propose and open M2RL, a Multi-player and Multi-agent Reinforcement Learning framework, to make training bots for complex games an easy-to-use warehouse. Experiments involving training a two-player multi-agent Wargame AI, and a sixteen-player multi-agent community game Neural MMO AI, demonstrate the effectiveness of the proposed framework by winning a silver award and beating high-level AI bots designed by professional players.

1 Introduction

Distributed deep reinforcement learning has gained significant attention in human-computer gaming since the breakthrough of AlphaGo series [Silver *et al.*, 2016]. Later, it has been tested for other kinds of complex games, such as AlphaStar for StarCraft [Vinyals *et al.*, 2019], OpenAI Five for Dota2 [Berner *et al.*, 2019], and JueWu for King of Glory [Ye *et al.*, 2020]. These large-scale complex games provide scenarios that are more aligned with real-life applications compared with simple academic environments like classical gym games, which have proved the necessity and significance of DDRL for complex games.

Currently, DDRL has ushered in an explosive growth, and many excellent DDRL frameworks have been developed and opened [Yin *et al.*, 2024]. Specially, SeedRL [Espeholt *et al.*, 2019] is an efficient DDRL framework, which designs a structure to connect inference model and optimizer model for improving resource utilization efficiency and data transmission speed. MAVA [Pretorius *et al.*, 2021], designed for building scalable multi-agent reinforcement learning system, is built on top of DeepMind's Acme [Hoffman *et al.*, 2020], and several MARL baseline systems have been implemented. MALib [Zhou *et al.*, 2023] is an advanced computing framework for population-based multi-agent reinforcement learning, with a centralized task dispatching model and a programming architecture named Actor-Learner-Evaluator.

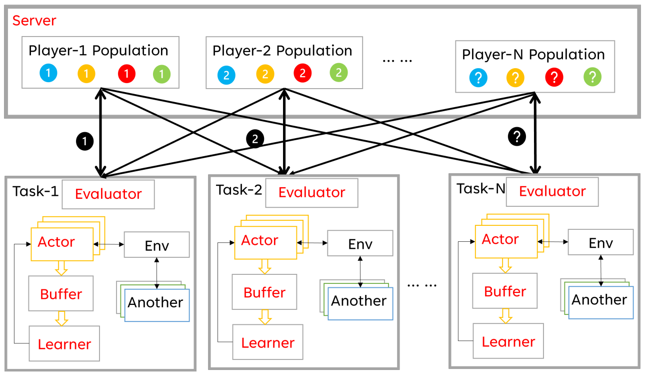

Figure 1: The overall architecture of M2RL.

Towards developing an open-sourced and easy-to-use DDRL system for complex games, we identity **multi-agent decision engineering**, **multi-player populations evolution**, and **multi-machine distributed training** as three primary elements, which are used for supporting asynchronous and heterogeneous agents cooperation and competition under imperfect information games, performing evolution of asymmetrical players, and utilizing multiple hardware devices, respectively. However, none of previous frameworks can support the three elements simultaneously, and have been tested in the training and evaluation of bots for complex game environments. Accordingly, we develop M2RL, which encodes the three elements into an **Actor-Learner-Buffer-Evaluator-Server** framework, and use a two-player multi-agent Wargame [Yin *et al.*, 2023] and a sixteen-player multi-agent community game Neural MMO [Suarez *et al.*, 2019] as complex game instances to validate the proposed system.

2 System Overview

The overall architecture of M2RL is shown in Figure 1, where actor-buffer-learner embedded with multi-agent decision engineering, is used for RL agents training, and evaluator-server embedded with multi-player populations evolution, is used to

*Corresponding author

Algorithm 1: Multi-Agent decision making.

```
input  : Initial observation $Obs$ from environment
output: $Trajectories$ for training
1  while not done do
2      Update $feature$ with $Obs$;
3      $validagents \leftarrow AgentMasking(Obs)$;
4      for $agent \leftarrow validagents$ do
5          $validactions \leftarrow ActionMasking(Obs)$;
6          if $validactions$ then
7              $feature \leftarrow StateEngineer(Obs)$;
8              $Output \leftarrow Model(feature)$;
9              $RecordCollect(feature, Output)$;
10             $action \leftarrow ActionEngineer(output)$;
11             $action$ add to $actionlist$;
12         $finishedaction \leftarrow Update(Obs)$;
13     if $finishedaction$ then
14         $reward \leftarrow RewardEngineer(Obs)$;
15         $Update(finishedaction, reward)$;
16     $Obs' \leftarrow step(actionlist)$;
17     $Obs \leftarrow Obs'$;
18 $Trajectories \leftarrow RecordFilter$
```

make evolution for different players. All the modules run in parallel in multiple machines with multiple GPUs and CPUs.

2.1 Multi-agent Decision Engineering

Usually, the perception stage and decision stage are required for a multi-agent decision. To cope with imperfect information processing and asynchronous and heterogeneous agents decision, which occur frequently for complex games, state engineering, action engineering and reward engineering should be specialized designed. The overall process is shown in Algorithm 1, where *AgentMasking* and *ActionMasking* are functions used to select agents and validate actions in the current time step.

State Engineering. M2RL encodes perfect observation as oracle features and a transforming function to obtain imperfect inference features, and embeds several typical encoding methods to extract the different kinds of observations such as one-hot for unit attributes.

Action Engineering. To cope with asynchronous decision-making, the action space is divided into continuous actions and momentary actions. Momentary actions refer to actions where the environment state is immediately updated in the next time step after issuing the action instruction. In the contrast, continuous actions are actions where the environment state update cannot be obtained immediately until the continuous actions are completed. Besides, action translators are introduced to transform actions from neural networks to actions suitable for game environment.

Reward Engineering. The sparse reward environment is not conducive to the learning and exploration of the agents. So, M2RL provides reward shaping, which includes expert knowledge templates and algorithms like intrinsic reward to estimate reward for each step.

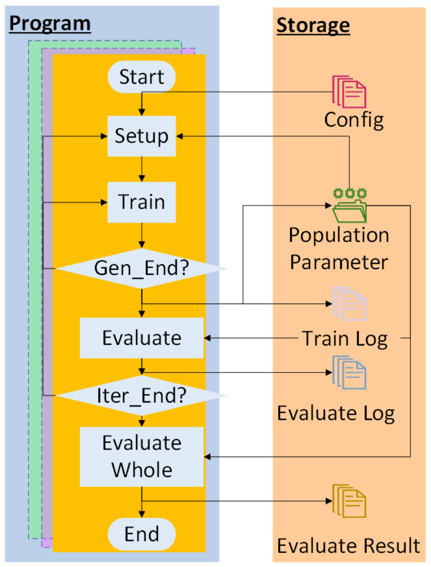

Figure 2: Process of population iterative training.

2.2 Multi-player Populations Evolution

In complex multi-player games, the asymmetry of players pose challenges to training, requiring the trained bots to have stronger robustness and diversity. M2RL provides a population strategy iterative training method that can be widely applied to various complex multi-player games, shown in Figure 2. Training of a player is wrapped as a task, which is independent and does not affect other players. Accordingly, each player can choose its own training approach based on its preferences and objectives without waiting for other players to complete their training. Besides, a player can make a copy of itself (a different task) by choosing different evolutionary approaches (e.g., self-play and fictitious self-play [Heinrich and Silver, 2016]), which largely enhances the flexibility.

During the populations evolution, there are two stages that require strategies evaluation. The first stage is in the iterative training process, where typical evaluation rules are used to generate the next confrontation population, such as self-play, fictitious self-play and prioritized fictitious self-play. The second evaluation step is to assess the final population strategy generated for the player after the iteration ends. This assessment involves evaluating the Nash equilibrium of the strategy population, as well as the rankings based on methods such as Elo and TrueSkill [Herbrich *et al.*, 2006].

2.3 Multi-machine Distributed Training

M2RL implements parallel processing of the five modules (see Figure 1) to improve data throughput, which bases on Ray [Malekmohammadi, 2021] as the scheduling layer, as shown in Figure 3. Specifically, we set neural network model on GPUs for inference and optimizing, and set the interaction with the environment, sampling storage on CPUs. The server is designed to collect and dispatch parameters between different machines, which is also deployed on CPUs.

As for the software selected for ease of adjustments and hardware compatibility, we select the PyTorch framework,

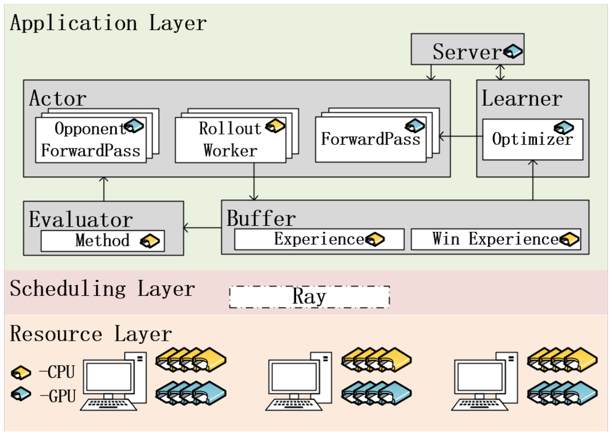

Figure 3: Multi-machine distributed training.

which is widely used for the newest hardware and most of popular games' environment software versions due to its better scalability. Additionally, we provide a simplified initial version of the training framework using TensorFlow 1.0. The network models built with TensorFlow 1.0 are capable of utilizing fewer GPU resources, thereby providing a convenient choice for users with limited resources.

3 Use Cases

We demonstrate the DDRL training capability of the proposed M2RL system through two experiments covering two different kinds of complex games, i.e., the IJCAI 2022 version Neural MMO with sixteen symmetrical players and Wargame with unbalanced abilities for the red and blue players. For all experiments, we use two machines equipped with 4 * RTX2080 Ti GPUs and 2 * 48-core CPUs.

Results on Neural MMO is shown in Figure 4. Since different players are symmetrical, one policy net can be used for all the players. We use self-play and fictitious self-play to train bots for each player, and compete against all the generated decision models and the built-in AI RandomTeam. Accordingly, we have self-play bots (SP) and fictitious self-play bots (FSP) with different generations. The average top1 ratio for these players in 40 testing maps are recorded, and each map is tested for 100 rounds. From the results, it can be seen that the ability of the agent is iteratively improved, and all trained models can clearly prevail in the face of unseen Team. Finally, based on our framework, our team AlphaMMO win silver tier award for IJCAI 2022 Neural MMO Challenge.

Results on Wargame is shown in Figure 5 (data used from [Yin *et al.*, 2024]). We use prioritized fictitious self-play to train the red and blue players for several generations, and compete against three professional-level AI bots. Besides, the asymmetric replicator dynamic method [Accinelli and Carrera, 2011] is used to show the dynamic equilibrium of strategies for both players. From Figure 5 (a), we can see that the learned bots for red and blue players can beat never seen high-performance AI bots, and from Figure 5 (b), we can see that the strategies iterated are becoming stronger.

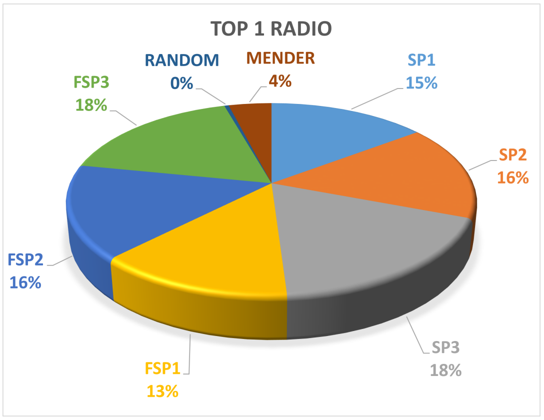

Figure 4: Top1 ratio on Neural MMO.

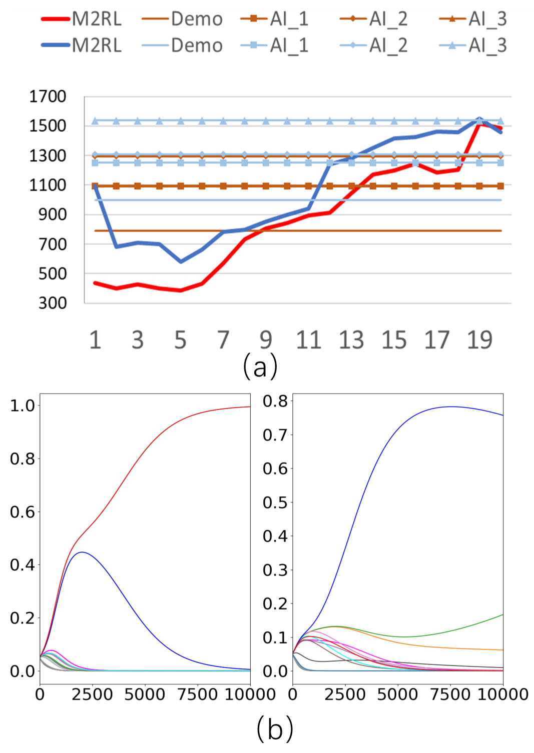

Figure 5: Results on Wargame (from [Yin *et al.*, 2024]).

4 Conclusion

This demo paper presents M2RL, a multi-player multi-agent distributed deep reinforcement learning framework and system for complex games. Our system is highly modularized and consists of essentially designed parts to deal with complex games facing asynchronous and heterogeneous agents cooperation and competition for multiple asymmetrical players, which also maximally utilizes the computing resources including CPUs and GPUs. M2RL has been tested on two complex games Wargame and Neural MMO with encouraging training results (a silver tier award and a professional level performance), showing an important infrastructure for solving complex games.

Acknowledgements

This work was supported by the National Science and Technology Major Project, Grant No.2022ZD0116403, and the Strategic Priority Research Program of Chinese Academy of Sciences, Grant No. XDA27000000.

References

[Accinelli and Carrera, 2011] Elvio Accinelli and Edgar J Sánchez Carrera. Evolutionarily stable strategies and replicator dynamics in asymmetric two-population games. In *Dynamics, Games and Science I: DYNA 2008, in Honor of Maurício Peixoto and David Rand, University of Minho, Braga, Portugal, September 8-12, 2008*, pages 25–35. Springer, 2011.

[Berner *et al.*, 2019] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[Espeholt *et al.*, 2019] Lasse Espeholt, Raphaël Marinier, Piotr Stanczyk, Ke Wang, and Marcin Michalski. Seed rl: Scalable and efficient deep-rl with accelerated central inference. *arXiv preprint arXiv:1910.06591*, 2019.

[Heinrich and Silver, 2016] Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016.

[Herbrich *et al.*, 2006] Ralf Herbrich, Tom Minka, and Thore Graepel. Trueskill™: a bayesian skill rating system. *Advances in neural information processing systems*, 19, 2006.

[Hoffman *et al.*, 2020] Matthew W Hoffman, Bobak Shahriari, John Aslanides, Gabriel Barth-Maron, Nikola Momchev, Danila Sinopalnikov, Piotr Stańczyk, Sabela Ramos, Anton Raichuk, Damien Vincent, et al. Acme: A research framework for distributed reinforcement learning. *arXiv preprint arXiv:2006.00979*, 2020.

[Malekmohammadi, 2021] Saber Malekmohammadi. Ray: A distributed framework for emerging ai applications. 2021.

[Pretorius *et al.*, 2021] Arnu Pretorius, Kale-ab Tessera, Andries P Smit, Claude Formanek, St John Grimbly, Kevin Eloff, Siphelele Danisa, Lawrence Francis, Jonathan Shock, Herman Kamper, et al. Mava: A research framework for distributed multi-agent reinforcement learning. *arXiv preprint arXiv:2107.01460*, 2021.

[Silver *et al.*, 2016] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[Suarez *et al.*, 2019] Joseph Suarez, Yilun Du, Phillip Isola, and Igor Mordatch. Neural mmo: A massively multiagent game environment for training and evaluating intelligent agents. *arXiv preprint arXiv:1903.00784*, 2019.

[Vinyals *et al.*, 2019] 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.

[Ye *et al.*, 2020] Deheng Ye, Zhao Liu, Mingfei Sun, Bei Shi, Peilin Zhao, Hao Wu, Hongsheng Yu, Shaojie Yang, Xipeng Wu, Qingwei Guo, et al. Mastering complex control in moba games with deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6672–6679, 2020.

[Yin *et al.*, 2023] Qiyue Yin, Meijing Zhao, Wancheng Ni, Junge Zhang, and Kaiqi Huang. Intelligent decision making technology and challenge of wargame. *Acta Automatica Sinica*, 49(5):913–928, 2023.

[Yin *et al.*, 2024] Qiyue Yin, Tongtong Yu, Shengqi Shen, Jun Yang, Meijing Zhao, Kaiqi Huang, Bin Liang, and Liang Wang. Distributed deep reinforcement learning: A survey and a multi-player multi-agent learning toolbox. *Machine Intelligence Research*, 21:411–430, 2024.

[Zhou *et al.*, 2023] Ming Zhou, Ziyu Wan, Hanjing Wang, Muning Wen, Runzhe Wu, Ying Wen, Yaodong Yang, Yong Yu, Jun Wang, and Weinan Zhang. Malib: A parallel framework for population-based multi-agent reinforcement learning. *J. Mach. Learn. Res.*, 24:150–1, 2023.