Hierarchical and Non-Hierarchical Multi-Agent InteractionsBased on Unity Reinforcement Learning

Zehong Cao
University of Tasmania,
Australia
zhcaonctu@gmail.com
zehong.cao@utas.edu.au

Kaichiu Wong
University of Tasmania, Australia
kaichiu.wong@utas.edu.au

Quan Bai
University of Tasmania,
Australia
quan.bai@utas.edu.au

Chin-Teng Lin
University of Technology Sydney,
Australiachin-teng.lin@uts.edu.au

OpenAI Gym [1] and DeepMind Lab [6], which were developed to

ABSTRAC

The open-source Unity platform, where agents can be trained using hierarchical or non-hierarchical reinforcement learning, supports the use of games and simulations as environments for multiple- agent interactions. In this demonstration, we present hierarchical and non-hierarchical multi-agent interactions based on Unity rein- forcement learning, specifically, hierarchical reinforcement learn- ing that sets different levels of agent's observations to achieve the goal. We created four multi-agent scenarios in the Unity environment, namely, Crawler, Tennis, Banana Collector, and Soc- cer, to test the interaction performances of hierarchical and non-hierarchical reinforcement learning. The simulation-interaction performances show that hierarchical reinforcement learning can be applied to multi-agent environments and can compete with agents trained via non-hierarchical reinforcement learning. **The demonstration video can be viewed at the following link: https://youtu.be/YQYQwLPXaL4**

KEYWORDS

Unity, Multi-Agent Interactions, Hierarchical, Reinforcement Learning

1 INTRODUCTION

Reinforcement learning (RL) typically refers to a goal-oriented al- gorithm that learns how to achieve complex tasks with mimicing human performance. In the agent training process, an agent ob- serves the environment and takes actions to receive rewards for accomplishing tasks in the process of achieving a goal. The agent is punished for making incorrect decisions and rewarded for making the right decisions, which makes this approach one of the most reliable training methods [5]. Many platforms that enable users to develop and test RL

algorithms are currently available, including

investigate how agents learn complex tasks. However, the above RL platforms, such as OpenAI Gym, lack the ability to flexibly con- figure the simulation for multiple agents; therefore, the simulation environment is an unmodifiable black box from the perspective of the learning system. Recently, the Unity platform released a new open-source toolkit [4] developed for creating and interacting with RL simulation environments. The toolkit enables games and simu- lations to serve as environments for training and testing intelligent RL agents, and these trained agents can be used for multiple pur- poses, including testing game builds and evaluating different game designs in multi-agent interactions.

To coordinate and test agent-agent interactions, we use RL to train agents in a developed environment to achieve an optimised policy. Some state-of-art RL algorithms have been developed to optimise the training performance, such as proximal policy optimi- sation (PPO) [7], which simplifies the training implementation to handle complex scenarios. Furthermore, to accelerate the learning process and improve generalisation in a multi-agent environment, hierarchical reinforcement learning (HRL) was proposed to learn a policy composed of multiple layers, each of which is responsible for control at a different level of temporal abstraction [9] [3]. One recent example of an HRL framework is feudal networks (FuNs)

[8] proposed by the DeepMind group, which employ a manager module and a worker module for hierarchical training. This frame- work was extended by [2] to a method called hierarchical critics assignment (HCA), which assigns a virtual manager that can be added on top of all worker agents in

the environment to observe the global environment and provide global critic signal to push worker agents towards the goal. Each worker agent must observe the local environment and take actions based on local and global critics. In our investigation of existing studies, we did not find any methods that support different RL types for multi-agent interaction in flexible environments. This lack of research motivated us to use RL (non-hierarchical approach) and HRL (hierarchical approach) for two agent teams separately to demonstrate the testing performance with real-time interactions.

In this demonstration, to support interaction among multiple agents, we train agents separately via RL (using the PPO algorithm) and HRL (using the HCA algorithm) approaches while playing a series of game scenarios in the Unity environment. We developed four multi-agent simulation scenarios in the Unity platform, namely, Crawler, Tennis, Banana Collector and Soccer. The original versions of these scenarios were designed by Unity: we modified them to use RL and HRL algorithms for multi-agent interaction tasks. Our demonstration makes the following contributions: 1) we develop new game environments to assist with RL and HRL designs and interactions for testing multi-agent systems; 2) the agents trained by HRL achieve better performance with high scores in competition- based interaction games; and 3) we consider the potential impact on applications in multi-agent competitions.

2 DEMONSTRATION SCENARIOS

In the Unity platform with a new open-source toolkit [4], we de- veloped four multi-agent scenarios as shown in Fig. 1, namely, Crawler, Tennis, Banana Collector and Soccer, to simulate multi- agent competitive interactions [1] . The experiments in each scenario are defined in terms of two agent teams: the blue team and the red team. Agents from the blue team are trained via HRL (hierarchical approach), whereas agents from the red team are trained via RL (non-hierarchical approach). Each agent targets a specific scenario goal to receive the maximum game score, and the game scores are recorded for the pre-trained RL and HRL agents during the interaction stage.

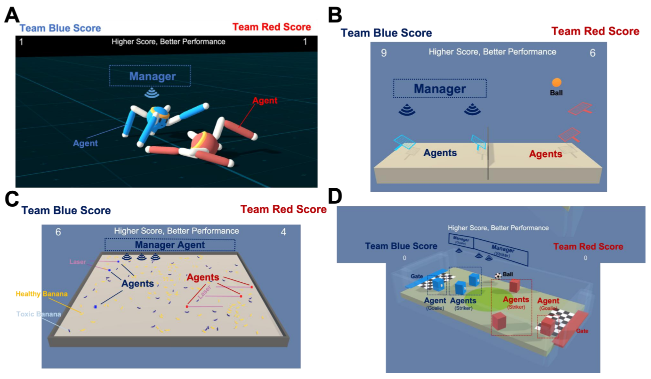

Figure 1: Multi-agent interactions in four scenarios

2.1 Crawler

The Crawler scenario is a modified scenario that originally allows a single agent to learn to walk in Unity. As shown in Fig. 1-A, we initiate a crawler agent with two arms and two forearms and create the logic for agents to learn to fight and compete with each other during the training progress. The agents are required to learn to maintain their body balance and to not touch the ground while walking to the opponent's position and then to fight against the opponent and cause the challenger to lose their balance to obtain a reward. In addition, as shown in the GitHub repository, the task goal, agent reward function, and behaviour parameters, including action and observation spaces, are defined for the crawler agents.

2.2 Tennis

Tennis competition is used as an example to simulate a sports game of bouncing a ball to an opponent's area in a multi-agent

environment. The agents are required to control the movement of a racket to ensure that the ball does not drop or fall outside of the boundaries on their side of the field. As an extension of the original scenario, as shown in Fig. 1-B, we increased the number of agents to two each on the blue and red teams, where the blue team, assigned to the HRL scenario, has a virtual manager on top of the agents to observe the global environment. Furthermore, as shown in the GitHub repository, the task goal, agent reward function, and behaviour parameters, including action and observation spaces, are defined for the tennis agents.

2.3 Banana Collector

The Banana Collector scenario involves multiple agents competing to collect target bananas. The environment consists of two different types of banana, healthy bananas and toxic bananas. Each agent must learn how to move and collect as many healthy (yellow) ba- nanas as possible while avoiding toxic (purple) bananas. When an agent touches and collects a toxic banana, the agent is frozen for 20 s and then continues to collect healthy bananas. As shown in Fig. 1-C, we create a manager on top of the blue-team agents trained by HRL to observe the global environment. We also define the task goal, agent reward function, and behaviour parameters, including action and observation spaces for the collector agents, as shown in the GitHub repository.

2.4 Soccer

As shown in Fig. 1-D, we create a two agent teams Soccer scenario in which the agents aim to attack the other team's gate and defences without the ball being kicked into their own gate. Each team has two types of agents, a goalie and a striker, who aim to defend their own gate and score by attacking the opponent's gate, respectively. For the blue-team agents, we add a virtual manager to obtain global observations and give critics of the agents' actions. As presented in the GitHub repository, the task goal, agent reward function, and behaviour parameters, including action and observation spaces, are defined for the soccer agents.

3 CONCLUSION

In summary, our demonstration shows that the Unity platform can support the development of new games and simulations for RL and HRL environments with multi-agent interactions. We created four scenarios with multiple agents in the Unity environment, namely, Crawler, Tennis, Banana Collector, and Soccer. We also presented hi- erarchical and non-hierarchical multi-agent interactions by means of RL and HRL algorithms and showed that the HRL-trained agents with a virtual manager that can observe global information achieve better performance with higher game scores as demonstrating in the video. We believe our demonstration has a potential impact on high attentions to HRL and the relevant applications in multi-agent competitions.

ACKNOWLEDGMENTS

This research was sponsored in part by the Office of Naval Research Global, US, and was accomplished under Cooperative Agreement Number ONRG - NICOP - N62909-19-1-2058.

REFERENCES

[1] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schul- man, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).

[2] Zehong Cao and Chin-Teng Lin. 2019. Reinforcement Learning from Hierarchical Critics. *arXiv preprint arXiv:1902.03079* (2019).

[3] Dongge Han, Wendelin Boehmer, Michael Wooldridge, and Alex Rogers. 2019. Multi-Agent Hierarchical Reinforcement Learning with Dynamic Termination. In *Proceedings of the 18th International Conference on Autonomous Agents and Multi- Agent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2006–2008.

[4] Arthur Juliani, Vincent-Pierre Berges, Esh Vckay, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. 2018. Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627* (2018).

[5] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. 1996. Rein- forcement learning: A survey. *Journal of artificial intelligence research* 4 (1996), 237–285.

[6] Joel Z Leibo, Cyprien de Masson d'Autume, Daniel Zoran, David Amos, Charles Beattie, Keith Anderson, Antonio García Castañeda, Manuel Sanchez, Simon Green, Audrunas Gruslys, et al. 2018. Psychlab: a psychology laboratory for deep reinforcement learning agents. *arXiv preprint arXiv:1801.08116* (2018).

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

[8] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. 2017. Feudal networks for hierar- chical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 3540–3549.

[9] Fangkai Yang, Daoming Lyu, Bo Liu, and Steven Gustafson. 2018. Peorl: Inte- grating symbolic planning and hierarchical reinforcement learning for robust decision-making. *arXiv preprint arXiv:1804.07779* (2018).