Evaluation of Different Multi-Agent Reinforcement Learning Strategies in Cooperative Tasks

*Zhi Li (zli3037@gatech.edu)

Git Hash: 59028e1

Abstract: **Multi-agent deep reinforcement learning (MARL) has gained increasing attention for its applicability in cooperative multi-agent learning tasks. However, the lack of comparisons between different MARL approaches hinders progress in the field. This report collects data and results from existing research and conducts comparison of three distinct classes of MARL algorithms: decentralized strategy, centralised training and decentralized execution, value decomposition. The results are leveraging data from various research papers which were performed on a diverse range of cooperative multi-agent learning tasks. By comparing the performance of these algorithms across different learning tasks, we gain valuable insights into the effectiveness of the various learning approaches.**

I. INTRODUCTION

Multi-agent deep reinforcement learning has shown great potential in addressing complex cooperative multi-agent learning tasks. However, comparing the effectiveness of different MARL algorithms is important. To bridge this gap, this report aims to provide an evaluation and comparison of three major classes of MARL algorithms: decentralized strategy, centralized multi-agent policy gradient, and value decomposition. By conducting experiments on diverse cooperative multi-agent learning tasks and gathering data from reputable research papers, we seek to facilitate meaningful comparisons between these approaches.

II. COMPARISON OF ALGORITHM

A. Decentralised Training Decentralised Execution

Independent Q-Learning (IQL):

In IQL, each agent maintains its own decentralised state-action value function, which depends solely on the local history of observations and actions of that specific agent. The agents update the parameters of their Q-value networks using the standard Q-learning loss, as introduced by Watkins and Dayan in 1992. The updates are performed independently by each agent, making it a fully decentralised algorithm.

Independent synchronous Advantage Actor-Critic (IA2C):

IA2C is a variant of the commonly used A2C (Advantage Actor-Critic) algorithm adapted for decentralised training in multi-agent systems. Each agent has its own actor network to approximate its policy and a critic network to approximate the value function. Both the actor and critic networks are trained simultaneously based on the history of local observations, actions, and rewards perceived by the individual agent. The training is done synchronously, and each agent optimizes its policy and value function independently.

Independent Proximal Policy Optimisation (IPPO):

IPPO is a variant of the widely used PPO (Proximal Policy Optimisation) algorithm, specifically designed for decentralised training in multi-agent systems. Like IA2C, each agent in IPPO has its own actor network and critic network. The architecture is identical to IA2C. The main difference between PPO and A2C lies in the optimization process. PPO uses a surrogate objective that constrains the relative change of the policy during each update, allowing for more update epochs with the same batch of trajectories. In contrast, A2C can only perform one update epoch per batch of trajectories to ensure that the training batch remains on-policy. IPPO, being a variant of PPO, follows the same update process but with decentralised training for multi-agent settings.

B. Centralised Training Decentralised Execution

Multi-Agent Deep Deterministic Policy Gradients (MADDPG):

MADDPG is a variation of the DDPG (Deep Deterministic Policy Gradients) algorithm specifically designed for cooperative multi-agent settings. Each agent consists of a decentralised actor network, which conditions its policy on local observations, and a centralised critic network. The critic is trained on joint observations and actions to approximate the joint state-action value function, enabling information sharing during training. The actors minimize the deterministic policy gradient loss, and the critic is trained using the TD-lambda algorithm. MADDPG can handle continuous action spaces and discrete action spaces (using the Gumbel-Softmax trick).

Counterfactual Multi-Agent (COMA) Policy Gradient:

COMA introduces a modification to the actor's loss computation that performs counterfactual reasoning for credit assignment in cooperative MARL. The advantage is computed as the discrepancy between the state-action value of the followed joint action and a counterfactual baseline. This baseline is the expected value of each agent following its current policy while the actions of other agents are fixed. The actor is trained using the standard policy loss with the modified advantage, and the critic is trained using the TD-lambda algorithm.

Multi-Agent A2C (MAA2C):

MAA2C extends the existing on-policy actor-critic algorithm A2C (Advantage Actor-Critic) by using centralised critics conditioned on the state of the environment rather than individual histories of observations. The critic learns a joint state value function. It is used as a baseline in MARL research and is sometimes referred to as Central-V due to its computation of a centralised state value function.

Multi-Agent PPO (MAPPO):

MAPPO is an extension of the widely-used PPO (Proximal Policy Optimization) algorithm for multi-agent settings. Similar to MAA2C, the critic in MAPPO learns a

joint state value function. The key difference is that MAPPO can perform several update epochs using the same training batch of trajectories, allowing for more efficient training compared to MAA2C, which can only perform one update epoch per batch.

C. Value Decomposition

Value Decomposition Networks (VDN):

VDN is a value decomposition algorithm in multi-agent reinforcement learning (MARL) that aims to learn a linear decomposition of the joint Q-value function. In VDN, each agent maintains a network to approximate its own state-action values. The algorithm decomposes the joint Q-value into the sum of individual Q-values for each agent. The joint state-action value function is trained using the standard DQN (Deep Q-Network) algorithm, and during training, gradients of the joint TD (Temporal Difference) loss flow backward to the network of each agent. By learning individual Q-values, VDN enables agents to focus on their local decision-making process without requiring direct communication or coordination with other agents.

QMIX (Q-value Mixing):

QMIX is an extension of VDN that addresses a broader class of environments. To represent a more complex decomposition of the joint Q-value, QMIX introduces a parameterized mixing network. The mixing network computes the joint Q-value based on each agent's individual state-action value function. A crucial requirement of the mixing function is that the optimal joint action, which maximizes the joint Q-value, should be the same as the combination of the individual actions maximizing the Q-values of each agent. By introducing the mixing network, QMIX can handle more complex cooperative multi-agent tasks compared to VDN. Like VDN, QMIX is trained to minimize the DQN loss, and the gradient is backpropagated to the individual Q-values.

III. Evaluation

A. Parameter Sharing

In deep multi-agent reinforcement learning (MARL), there are two common configurations for training algorithms: with and without parameter sharing.

Without Parameter Sharing:

In this configuration, each agent maintains its own set of parameters for its neural networks. These networks include the policy network responsible for decision-making and, if applicable, the critic network used for value estimation. Without parameter sharing, each agent's networks learn independently, and there is no exchange of information or shared knowledge between agents during training.

With Parameter Sharing:

With parameter sharing, all agents in the system share the same set of parameters for their neural networks. The policy and critic (if used) networks receive an additional input in the form of the identity of each agent as a one-hot vector. This allows each agent to develop a distinct behavior despite sharing parameters. During training, the loss is computed over all agents, and this collective loss is used to optimize the shared parameters, promoting coordinated learning among agents.

B. Performance Metrics

In deep multi-agent reinforcement learning (MARL), there are two common configurations for training algorithms: with and without parameter sharing.

Without Parameter Sharing:

In this configuration, each agent maintains its own set of parameters for its neural networks. These networks include the policy network responsible for decision-making and, if applicable, the critic network used for value estimation. Without parameter sharing, each agent's networks learn independently, and there is no exchange of information or shared knowledge between agents during training.

With Parameter Sharing:

With parameter sharing, all agents in the system share the same set of parameters for their neural networks. The policy and critic (if used) networks receive an additional input in the form of the identity of each agent as a one-hot vector. This allows each agent to develop a distinct behavior despite sharing parameters. During training, the loss is computed over all agents, and this collective loss is used to optimize the shared parameters, promoting coordinated learning among agents.

IV. Results

In this section, the compiled results are presented. Figure 1 illustrates the normalized evaluation returns for all algorithms in all environments, excluding matrix games. To compare and analyze the performance of different algorithms consistently across tasks, the returns are normalised using the formula: norm_Gat = (Gat - min(Gt)) / (max(Gt) - min(Gt)), where Gat represents the return of algorithm 'a' in task 't', and Gt represents the returns of all algorithms in task 't'.

Table 1 displays the maximum returns achieved by the nine algorithms in all 25 tasks with parameter sharing. In Table 1, the highest mean return for each task is highlighted in bold. Two-sided t-tests are conducted with a significance threshold of 0.05 to compare the highest-performing algorithm against each other algorithm in each task. If an algorithm's performance is not statistically significantly different from the best algorithm, the corresponding value is annotated with an asterisk. The bold or asterisks in the table indicate the best-performing algorithms per task.

A. Decentralised Training Decentralised Excution

It is observed that Independent Learning (IL) algorithms perform adequately in most tasks, despite their simplicity. However, their performance is limited in partially observable SMAC and RWARE tasks when compared to their Centralised Training Decentralised Execution (CTDE) counterparts. This limitation arises from the IL algorithms' inability to reason over joint information of agents, which becomes crucial in these specific environments.

Below are the performance highlights of the specific IL algorithms:

IQL (Independent Q-Learning):

- IQL performs significantly worse than the other IL algorithms in the partially observable Speaker-Listener task and all RWARE tasks.

- It is particularly effective in all but three LBF tasks, where relatively larger grid-worlds are used.

- Interestingly, IQL achieves the best performance among all algorithms in the "3s_vs_5z" task and performs competitively in the rest of the SMAC tasks.

IA2C (Independent Advantage Actor-Critic):

- The stochastic policy of IA2C proves to be particularly effective in most environments, except for a few SMAC tasks where its performance is slightly lower.

- In the majority of tasks, it performs similarly to IPPO, except for RWARE and some SMAC tasks where it outperforms IPPO.

- IA2C performs competitively compared to all CTDE algorithms and significantly outperforms COMA and MADDPG in most tasks.

IPPO (Independent Proximal Policy Optimization):

- IPPO generally performs competitively in all tasks across different environments.

- On average, it achieves higher returns than IA2C in MPE, SMAC, and RWARE tasks, but slightly lower returns in the LBF tasks.

- IPPO outperforms MAA2C in the partially observable RWARE tasks, but in general, it performs worse compared to its centralised MAPPO version.

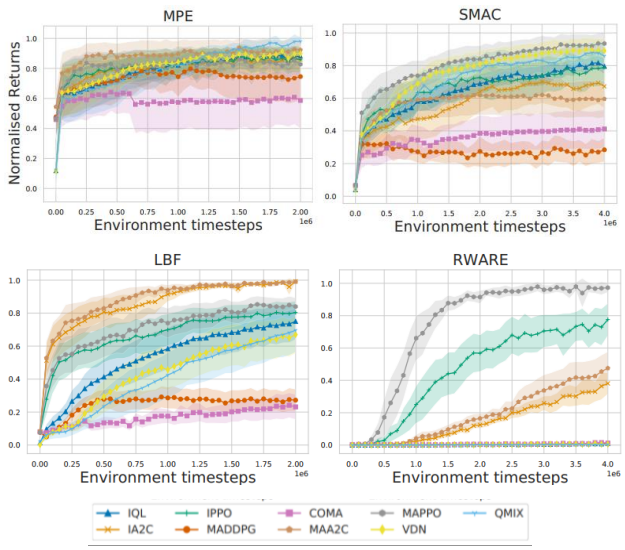

Figure 1: Normalised evaluation returns averaged over the tasks in the all environments. Shadowed part represents the 95% confidence interval.

B. Centralised Training Decentralised Excution

In the evaluation, it is found that centralised training, which focuses on learning powerful critics over joint observations and actions, proves to be valuable in tasks requiring significant coordination under partial observability, such as the MPE Speaker-Listener and harder SMAC tasks. However, Independent Learning (IL) algorithms remain competitive compared to Centralised Training Decentralised Execution (CTDE) algorithms in fully observable tasks like MPE and LBF.

Below are the performance highlights for each category of algorithms:

Centralised Multi-Agent Policy Gradient:

- MADDPG performs worse than most other algorithms, except COMA, in the majority of tasks, with only some competitive performance in certain MPE tasks.

- COMA exhibits low performance in most tasks and only performs competitively in one SMAC task, likely due to high variance in the computation of counterfactual advantage and difficulty in finding solutions in the Speaker-Listener and RWARE tasks.

- MAA2C generally performs competitively in most tasks but achieves slightly lower returns compared to MAPPO in SMAC and RWARE tasks.

MAPPO (Multi-Agent Proximal Policy Optimization):

- MAPPO achieves high returns in most tasks and demonstrates the benefits of on-policy optimization with its surrogate objective, leading to improved sample efficiency compared to MAA2C. It outperforms other algorithms in the RWARE tasks.

Value Decomposition (VDN and QMIX):

- VDN and QMIX prove to be effective approaches in most environments, outperforming or matching the highest returns of any other algorithm, except in the RWARE tasks. However, in the RWARE tasks, they, along with other algorithms like IQL, COMA, and MADDPG, struggle due to sparse rewards, indicating the need for sufficiently dense rewards to enable value function decomposition.

- VDN's assumption of linear value function decomposition is violated in some MPE tasks, whereas QMIX performs consistently well across most tasks. QMIX's value function decomposition allows it to achieve slightly higher returns in more complicated tasks where linear decomposition is insufficient.

In conclusion, centralised training shows its advantage in tasks requiring coordination under partial observability, while IL algorithms perform competitively in fully observable tasks. Value decomposition methods like VDN and QMIX exhibit strong performance in most environments, showcasing the advantages of combining centralised training and decentralised execution. However, the success of value decomposition methods also depends on the availability of sufficiently dense rewards, as seen in the challenging RWARE tasks. Overall, each category of algorithms demonstrates specific strengths and limitations, underscoring the importance of choosing the appropriate approach based on the characteristics of the multi-agent environment.

Table 1: Maximum Returns and 95% Confidence Interval for All Nine Algorithms with Parameter Sharing in 25

Tasks

	Tasks \Algs.	IQL	IA2C	IPPO	MADDPG	COMA	MAA2C	MAPPO	VDN	QMIX
Matrix Games	Climbing	**195.00 ± 67.82**	175.00 ± 0.00	175.00 ± 0.00	170.00 ± 10.00	185.00 ± 48.99	175.00 ± 0.00	175.00 ± 0.00	175.00 ± 54.77	175.00 ± 54.77
	Penalty k=0	**250.00 ± 0.00**	**250.00 ± 0.00**	**250.00 ± 0.00**	249.98 ± 0.04	**250.00 ± 0.00**	**250.00 ± 0.00**	**250.00 ± 0.00**	**250.00 ± 0.00**	**250.00 ± 0.00**
	Penalty k=-25	**50.00 ± 0.00**	**50.00 ± 0.00**	**50.00 ± 0.00**	49.97 ± 0.02	**50.00 ± 0.00**	**50.00 ± 0.00**	**50.00 ± 0.00**	**50.00 ± 0.00**	**50.00 ± 0.00**
	Penalty k=-50	**50.00 ± 0.00**	**50.00 ± 0.00**	**50.00 ± 0.00**	49.98 ± 0.02	**50.00 ± 0.00**	**50.00 ± 0.00**	**50.00 ± 0.00**	**50.00 ± 0.00**	**50.00 ± 0.00**
	Penalty k=-75	**50.00 ± 0.00**	**50.00 ± 0.00**	**50.00 ± 0.00**	49.97 ± 0.02	**50.00 ± 0.00**	**50.00 ± 0.00**	**50.00 ± 0.00**	**50.00 ± 0.00**	**50.00 ± 0.00**
	Penalty k=-100	**50.00 ± 0.00**	**50.00 ± 0.00**	**50.00 ± 0.00**	49.97 ± 0.03	**50.00 ± 0.00**	**50.00 ± 0.00**	**50.00 ± 0.00**	**50.00 ± 0.00**	**50.00 ± 0.00**
MPE	Speaker-Listener	−18.36 ± 4.67	−12.60 ± 3.62 *	−13.10 ± 3.50	−13.56 ± 1.73	−30.40 ± 5.18	−10.71 ± 0.38 *	**−10.68 ± 0.30**	−15.95 ± 2.48	−11.56 ± 0.53
	Spread	−132.63 ± 2.22	−134.43 ± 1.15	−133.86 ± 3.67	−141.70 ± 1.74	−204.31 ± 6.30	−129.90 ± 1.63 *	−133.54 ± 3.08	−131.03 ± 1.85	**−126.62 ± 2.96**
	Adversary	9.38 ± 0.91	12.12 ± 0.44 *	**12.17 ± 0.32**	8.97 ± 0.89	8.05 ± 0.89	12.06 ± 0.45 *	11.30 ± 0.38	9.28 ± 0.90	9.67 ± 0.66
	Tag	22.18 ± 2.83	17.44 ± 1.31	19.44 ± 2.94	12.50 ± 6.30	8.72 ± 4.42	19.95 ± 7.15 *	18.52 ± 5.64	24.50 ± 2.19	**31.18 ± 3.81**
SMAC	2s_vs_1sc	16.72 ± 0.38	20.24 ± 0.00	20.24 ± 0.01	13.14 ± 2.01	11.04 ± 7.21	20.20 ± 0.05 *	**20.25 ± 0.00**	18.04 ± 0.33	19.01 ± 0.40
	3s5z	16.44 ± 0.15	18.56 ± 1.31 *	13.36 ± 2.08	12.04 ± 0.82	18.90 ± 1.01 *	19.95 ± 0.05 *	**20.39 ± 1.14**	19.57 ± 0.20 *	19.66 ± 0.14 *
	corridor	15.72 ± 1.77	**18.59 ± 0.62**	17.97 ± 3.44 *	5.85 ± 0.58	7.75 ± 0.19	8.97 ± 0.29	17.14 ± 4.39 *	15.25 ± 4.18 *	16.45 ± 3.54 *
	MMM2	13.69 ± 1.02	10.70 ± 2.77	11.37 ± 1.15	3.96 ± 0.32	6.95 ± 0.27	10.37 ± 1.95	17.78 ± 0.44	**18.49 ± 0.31**	18.40 ± 0.24 *
	3s_vs_5z	**21.15 ± 0.41**	4.42 ± 0.02	19.36 ± 6.15 *	5.99 ± 0.58	3.23 ± 0.05	6.68 ± 0.55	18.17 ± 4.17 *	19.03 ± 5.77 *	16.04 ± 2.87
LBF	8x8-2p-2f-c	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	0.46 ± 0.02	0.61 ± 0.30	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	0.96 ± 0.07 *
	8x8-2p-2f-2s-c	**1.00 ± 0.00**	**1.00 ± 0.00**	0.78 ± 0.05	0.70 ± 0.04	0.45 ± 0.15	**1.00 ± 0.00**	0.95 ± 0.06	**1.00 ± 0.00**	**1.00 ± 0.00**
	10x10-3p-3f	0.93 ± 0.02	**1.00 ± 0.00**	0.98 ± 0.01	0.24 ± 0.04	0.19 ± 0.06	**1.00 ± 0.00**	0.99 ± 0.01	0.84 ± 0.08	0.84 ± 0.08
	10x10-3p-3f-2s	0.86 ± 0.01	0.94 ± 0.03 *	0.70 ± 0.03	0.41 ± 0.03	0.29 ± 0.12	**0.96 ± 0.02**	0.72 ± 0.03	0.90 ± 0.03	0.90 ± 0.01
	15x15-3p-5f	0.17 ± 0.08	**0.89 ± 0.04**	0.77 ± 0.08	0.10 ± 0.02	0.08 ± 0.04	0.87 ± 0.06 *	0.77 ± 0.02	0.15 ± 0.02	0.09 ± 0.04
	15x15-4p-3f	0.54 ± 0.18	0.99 ± 0.01 *	0.98 ± 0.01	0.17 ± 0.03	0.17 ± 0.04	**1.00 ± 0.00**	0.96 ± 0.02	0.38 ± 0.13	0.15 ± 0.06
	15x15-4p-5f	0.22 ± 0.04	0.93 ± 0.03 *	0.67 ± 0.22	0.12 ± 0.06	0.12 ± 0.06	**0.95 ± 0.01**	0.70 ± 0.25 *	0.30 ± 0.04	0.25 ± 0.09
RWARE	Tiny 4p	0.72 ± 0.37	26.34 ± 4.60	31.82 ± 10.71	0.54 ± 0.10	1.16 ± 0.15	32.50 ± 9.79	**49.42 ± 1.22**	0.80 ± 0.28	0.30 ± 0.19
	Small 4p	0.14 ± 0.28	6.54 ± 1.15	19.78 ± 3.12	0.18 ± 0.12	0.16 ± 0.16	10.30 ± 1.48	**27.00 ± 1.80**	0.18 ± 0.27	0.06 ± 0.08
	Tiny 2p	0.28 ± 0.38	8.18 ± 1.25	20.22 ± 1.76 *	0.44 ± 0.34	0.48 ± 0.34	8.38 ± 2.59	**21.16 ± 1.50**	0.12 ± 0.07	0.14 ± 0.19

C. *Parameter Sharing*

Figure 2 provides a visual representation of the normalized maximum returns averaged over all nine algorithms and tasks with and without parameter sharing in all environments. The observations from the figure show that, except for matrix games, parameter sharing generally leads to improved returns compared to no parameter sharing.

The improvements with parameter sharing can be attributed to its ability to use a larger number of trajectories to train the same shared architecture, leading to enhanced sample efficiency. By sharing parameters, agents can learn from each other's experiences, enabling better coordination and more effective decision-making in complex multi-agent environments. This sharing of information and knowledge allows the algorithms to leverage joint observations and actions, resulting in improved overall performance.

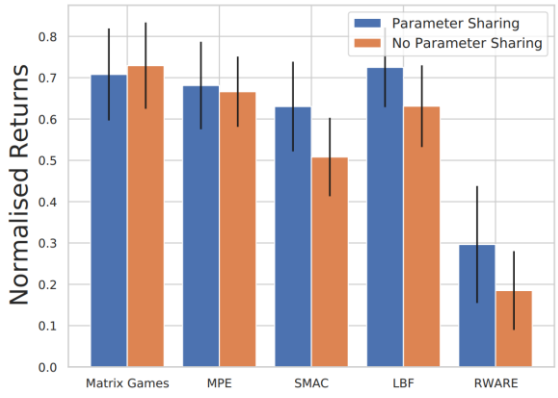

Figure 2: Normalised maximum returns averaged over all algorithms with/without parameter sharing (with standard error).

V. Analysis

Independent Learning (IL) can be effective in multi-agent systems under certain conditions. IL algorithms are capable of achieving competitive performance in tasks that do not require extensive coordination among agents and where each agent can independently learn a policy that achieves relatively high returns based solely on its local observation history. For example, in smaller tasks with simpler coordination requirements, agents can learn effective policies without relying heavily on joint information or extensive collaboration with other agents.

However, the effectiveness of IL algorithms diminishes in tasks that demand significant coordination among agents and where the environment is partially observable. In such cases, centralised information becomes crucial for successful decision-making and effective coordination. IL algorithms that rely on off-policy training from experience replay buffers face challenges in such environments, as the individual agents lack the necessary joint information to choose well-coordinated actions. On the other hand, on-policy IL algorithms like IA2C and IPPO can leverage the currently demonstrated behavior of other agents to learn more effective policies, and they are not as affected by the non-stationarity issue observed in off-policy IL.

Centralised training methods, such as MAA2C and MAPPO, provide an advantage in tasks with partial observability, as they have access to joint information (observations and actions) over all agents, enabling them to optimize individual policies more effectively. This advantage is particularly significant in tasks where agent observations lack important information about other agents or parts of the environment due to partial observability. In fully observable tasks, where all agents have access to the same information, both centralised and non-centralised algorithms perform similarly.

Value decomposition is a promising approach to improve the performance of IL algorithms. VDN and QMIX demonstrate better performance compared to the otherwise similar IQL algorithm in most tasks. QMIX's flexible approach of using a trainable mixing network for joint Q-value computation proves to be beneficial, particularly in harder tasks that involve complicated interactions between agents and result in non-stationarity. However, this flexibility comes with additional computational expense. In simpler environments, where decomposition does not have to be complex, VDN and QMIX perform similarly, with both outperforming policy gradient methods (except COMA). However, value decomposition methods, including VDN and QMIX, struggle in tasks with sparse rewards, as seen in the RWARE environment, where learning individual utilities becomes challenging due to rarely achieving positive rewards.

In summary, the effectiveness of Independent Learning in multi-agent systems depends on the complexity of the coordination required, the observability of the environment, and the availability of joint information. Centralised training methods offer advantages in tasks with partial observability, while value decomposition methods like VDN and QMIX show promise in enhancing the performance of IL algorithms in various cooperative multi-agent learning tasks.

VI. Conclusions

In this report, author collected a comprehensive evaluation of nine MARL algorithms in a diverse set of 25 cooperative learning tasks. These tasks varied in terms of observability, reward density, and the number of agents involved, providing a thorough analysis of the algorithms' performance in different scenarios. The analysis revealed that independent learning can be effective in simpler tasks where extensive coordination is not required, but it becomes limited in tasks with partial observability and extensive coordination demands. Centralised training proved beneficial in tasks with partial observability, as it leveraged joint information for more effective decision-making. Value decomposition methods, like VDN and QMIX, demonstrated promising results, especially in tasks involving complex interactions between agents.

However, our study does have limitations. We focused solely on cooperative environments and commonly-used MARL algorithms. Future research should explore competitive environments and tackle other MARL challenges, such as exploration, communication, and opponent modeling. These additional studies will further contribute to our understanding of the strengths and limitations of existing MARL algorithms and provide valuable insights for practical considerations and future research in the field. Overall, we hope that our work provides valuable guidance for researchers and practitioners in multi-agent deep reinforcement learning.

VII. Declaration

Due to a series of technical difficulties and data issues encountered during Project 3, I have decided to change my research topic. Unfortunately, none of the environment setups for Project 3 were functioning as expected on my end, and I faced various issues at different stages of the setup and simulation process. Additionally, obtaining the correct baseline log data proved to be problematic, as the downloaded data was only a fraction of the expected size.

The data and results are collected from the reference.

References

[1] Sutton, R. S., & Barto, A. (2018). Reinforcement learning: An introduction. Cambridge, MA: The MIT Press.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] https://gymnasium.farama.org/environments/box2d/lunar_lander/

[3] https://github.com/PacktPublishing/Hands-on-Reinforcement-Learning-with-PyTorch

[4] Georgios Papoudakis, Filippos Christianos, Arrasy Rahman, and Stefano V. Albrecht. Dealing with non-stationarity in multi-agent deep reinforcement learning. arXiv preprint arXiv:1906.04737,2019.

[5] Tian, Y., Cao, M., & Zhang, J. (2020). Benchmarking Multi-Agent Deep Reinforcement Learning Algorithms in Cooperative Tasks. arXiv preprint arXiv:2006.07869.

[6] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. International Conference on Machine Learning, 1993.

[7] 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. Human-level control through deep reinforcement learning. Nature, pages 529–533, 2015.

[8] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. International Conference on Machine Learning, 2016.

[9] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. OpenAI baselines, 2017.

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

[11] Richard S. Sutton. Learning to predict by the methods of temporal differences. Machine learning, 1988.

[12] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. International Conference on Machine Learning, 1993.

[13] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. Advances in Neural Information Processing Systems, 2017.

[14] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. arXivpreprint arXiv:1509.02971, 2015.

[15] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax.International Conference on Learning Representations, 2017.

[16] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson.

[17] Counterfactual multi-agent policy gradients. AAAI Conference on Artificial Intelligence, 2018.

[18] 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.