Exploring Parameter Sharing in Multi-Agent Reinforcement Learning

Yongtao Qu

under the guidance of Professor Christopher Amato

Khoury College of Computer Science

Northeastern University

Boston, MA

qu.yo@northeastern.edu, c.amato@northeastern.edu

Dec 08, 2024

1 Objectives and Significance

Parameter sharing has long been a pivotal strategy in enhancing learning efficiency within multi-agent reinforcement learning(MARL) since 1993[18] or even earlier. This approach allows agents to leverage shared experiences and gradients, thereby significantly reducing the total number of policy parameters and improving learning efficiency. Many state-of-the-art deep MARL techniques, including value-based methods[17, 14, 6], policy gradients[11, 26], and communication learning algorithms[2, 23], have adopted parameter sharing, demonstrating remarkable performance in complex tasks such as StarCraft II[16].

Despite its widespread application, the performance of parameter sharing in multi-agent reinforcement learning remains underexplored in terms of systematic analysis. A thorough investigation of parameter sharing could illuminate its underlying properties and facilitate the development of more effective MARL algorithms. This project addresses this gap by analyzing several aspects

of parameter sharing, contributing to a deeper understanding of parameter sharing and its implications for advancing MARL methodologies. Several main findings include: (1) parameter sharing accelerates learning and facilitates coordination in homogeneous settings but struggles in environments requiring role specialization; (2) neural network architectures, such as RNNs, play a crucial role in the effectiveness of parameter sharing; (3) while agent indication and observation/action space padding offer solutions for specific challenges, they exhibit limited success in fully resolving issues like exploration in sparse reward environments. These insights provide valuable guidance for designing more effective MARL systems in the future.

2 Background

2.1 Reinforcement Learning

Reinforcement Learning (RL) is a machine learning paradigm in which an agent learns to make decisions by interacting with an environment. The agent takes actions in different situations, receives feedback in the form of rewards, and aims to maximize the cumulative reward over time. Unlike supervised learning, which uses labeled data, RL involves learning through trial and error, with the agent continuously improving its strategy based on the outcomes of its actions. The main goal of RL is to learn an optimal policy—a set of rules that defines the best action to take in each state to achieve the highest cumulative reward.

The RL process is often modeled as a Markov Decision Process (MDP), defined by the tuple (S, A, P, R, γ), where S represents the set of states, A the set of actions, $P : S \times A \times S \to [0, 1]$ the state transition probability function, $R : S \times A \to R$ the reward function, and $\gamma \in [0, 1]$ the discount factor that determines the importance of future rewards. The goal in RL is to learn a policy $\pi : S \to A$ that maximizes the expected cumulative reward, often represented by the value function $V(s)$, which estimates the expected return from state s.

A fundamental concept in RL is the Bellman equation, which describes the relationship between the value of a state and the values of subsequent states:

$$V(s) = \max_a \left[R(s, a) + \gamma \sum_{s'} P(s' \mid s, a) V(s') \right]$$

The Bellman equation forms the foundation for many RL algorithms, including both value-based methods (e.g., Q-learning) and policy-based methods (e.g., Policy Gradient), enabling agents to iteratively improve their decision-making strategies by evaluating and optimizing their actions.

Several classic RL algorithms have been developed based on these foundational principles. Q-learning is one of the most well-known value-based algorithms, which uses a Q-table to store the expected rewards for each action in every state. The agent updates the Q-values iteratively using the Bellman equation to converge towards the optimal policy. Deep Q-Networks (DQN)[13] extend Q-learning by utilizing deep neural networks to approximate Q-values, making it feasible for environments with large or continuous state spaces. Policy Gradient methods, in contrast, directly optimize the policy by adjusting the parameters to increase the expected reward, without relying explicitly on value functions.

2.2 Multi-Agent Reinforcement Learning

Multi-Agent Reinforcement Learning (MARL) extends traditional RL to environments involving multiple interacting agents, each with its own objectives. In MARL, agents must learn not only to optimize their individual rewards but also to adapt to the behaviors of other agents, making the problem significantly more complex compared to single-agent RL. These agents can either cooperate, compete, or form mixed relationships, depending on the nature of the task and their individual goals. One of the main challenges in MARL is the issue of non-stationarity. Since all agents are learning simultaneously, the environment from the perspective of each agent is constantly changing, making it difficult for agents to converge to optimal policies.

In cooperative settings, agents work together to maximize a shared reward. Centralized Training with Decentralized Execution (CTDE) is a common approach in these scenarios, where agents are trained together in a centralized manner but act independently during execution. This allows agents to leverage shared information during training while maintaining independence in decision-making. While in competitive or mixed settings, where agents may have conflicting goals, learning becomes more dynamic as agents need to predict and counteract the actions of others. Algorithms like Multi-Agent Deep Deterministic Policy Gradient (MADDPG)[11] are designed to address such environments by allowing each agent to learn policies that consider the actions of other agents.

2.3 Parameter Sharing

Despite its wide use, multi-agent reinforcement learning (MARL) remains challenging due to the exponential growth of the joint action-observation space with the number of agents and the continuous evolution of agents' policies. Parameter sharing is a widely adopted approach in MARL to address the challenges of scalability and non-stationarity. In parameter sharing, multiple agents share a common set of parameters, typically in their policy or value networks, i.e. $\theta_1 = \cdots = \theta = \theta$, such that $\pi_1 = \cdots = \pi = \pi$ and $\boldsymbol{\pi} = (\pi, \cdots, \pi)$. This significantly reduces the number of required parameters, lowering computational complexity and speeding up the training process [21]. Furthermore, by sharing parameters, agents benefit from each other's experiences, accelerating learning and fostering cooperation.

Thanks to these advantages, parameter sharing has become a standard setting in MARL methods, dating back as early as 1993 [18]. Modern methods, such as value-based methods[17, 14, 6], policy gradients[11, 26], and communication learning algorithms[2, 23]. This approach has also been applied in practical domains, such as modeling driving behaviors [7].

However, parameter sharing introduces several challenges, as agents tend to learn similar policies. This can limit performance in scenarios where agents must exhibit diverse behaviors to achieve optimal outcomes. For example, Fu [4] demonstrates that policy gradient algorithms with parameter sharing fail to find the optimal solution in a simple two-agent XOR game. Notably, sacrificing the benefits of parameter sharing to achieve diversity can also be detrimental. Sharing necessary experiences or an understanding of tasks can broadly accelerate cooperative learning, as observed in human interactions. Without parameter sharing, agents must search within a much larger parameter space, which can be inefficient, as agents do not always need to behave differently.

The key challenge is thus to adaptively balance diversity and parameter sharing. As Li [9] suggests, this trade-off is critical for optimizing the effectiveness of MARL algorithms. These considerations highlight the need for a more detailed investigation into how parameter sharing should be utilized to best address the unique demands of multi-agent systems.

3 Methods

3.1 Recent Progresses on Parameter sharing

The use of agent indication was first introduced by RIAL [3], where each agent receives its own unique identifier as part of its input. This simple yet effective mechanism allows agents to specialize their behaviors despite sharing parameters. This idea has been widely adopted in subsequent works, such as [5], and remains a foundational method for enabling diversity within parameter-sharing frameworks.

As parameter sharing has become a default setting in many MARL methods, some recent research has shifted focus to critically examining its performance and addressing its limitations. Terry et al. [19] provided theoretical insights into parameter sharing for heterogeneous agents, proposing three key claims: (1) when agents have disjoint observation spaces, a shared model can still distinguish between agents and learn an optimal policy; (2) if observation spaces are not disjoint, agent indication can force distinct behavior; and (3) when agents have differing observation or action dimensions, padding them to the size of the largest allows training to proceed effectively. While these insights offer valuable guidelines, the lack of empirical validation leaves their practical applicability uncertain.

Recent advancements have sought to improve parameter sharing by balancing its strengths with the need for agent diversity. Christianos et al. [1] introduced selective parameter sharing, a framework where agents are grouped based on task similarity, allowing parameter sharing within groups but not across them. This approach reduces the number of parameters while achieving comparable or even superior performance to independent learning in certain scenarios. Similarly, Wang et al. [22] and Yang et al. [24] proposed task decomposition methods, where agents working on similar subtasks share parameters. These methods leverage regularizers in the loss function to encourage efficient sharing while maintaining necessary distinctions in behavior.

Li et al. [9] addressed the challenge of role differentiation within shared parameter networks by introducing agent-specific modules. These modules maximize mutual information between agents' identities and their trajectories, encouraging agents to develop distinct roles. However, this approach can lead to overfitting, as agents tend to revisit familiar trajectories rich in identity

information, limiting exploration and potentially resulting in suboptimal performance. Other recent efforts include [8], [20], and [10], each with focus on various mechanisms for improving the adaptability and efficiency of parameter sharing.

Despite these advancements, significant gaps remain in the theoretical and empirical understanding of parameter sharing. While individual studies tackle specific challenges, a comprehensive analyzing of its broader applicability is still lacking. Bridging these gaps is essential to fully harness the potential of parameter sharing in multi-agent reinforcement learning and enable its effective use across diverse scenarios.

3.2 Environments

This project explores reinforcement learning, where data are generated by agents interacting within environments, therefore a static dataset is not required. Instead, agents will run in various environments to comprehensively evaluate experimental outcomes. The majority of experiments in this project are conducted on the Multi-Agent Particle Environments [12], which consist of communication-oriented scenarios where particle agents can move, communicate, observe each other, push one another, and interact with fixed landmarks. Three standard cooperative environments from the library are utilized: Simple-Spread, Simple-Reference, and Simple-Speaker-Listener. Besides the standard environments given, two custom-designed environments, Multi-Color-Spread and 4vs5-Spread, are introduced to further investigate specific challenges in multi-agent reinforcement learning. These environments are described below. The complete list of environments used in this project is summarized in Table 1.

Multi-Color-Spread In this environment, there are four smaller agents initialized in the center and four larger landmarks placed in the corners. Each agent and landmark is assigned a unique color, and their positions are randomized in every episode. The objective of the agents is to minimize the distance between themselves and their corresponding colored landmarks.

4vs5-Spread In this environment, there are four agents (colored green) and five landmarks, with four positioned in the corners and one in the center. Agents are initialized at random positions, and their goal is to get as close as possible to any landmark. A landmark becomes ”occupied” when

an agent gets sufficiently close to it, and will result the team a +10 reward per occupied landmark. Ideally, the four agents should coordinate to occupy the four corner landmarks to maximize rewards. However, challenges arise as a ”lazy” agent might occupy the center landmark while the other three agents spread to the corners, still yielding +40 rewards. This scenario is difficult to learn because it introduces misleading feedback during training, often causing agents to converge on suboptimal strategies or become stuck at the center landmark.

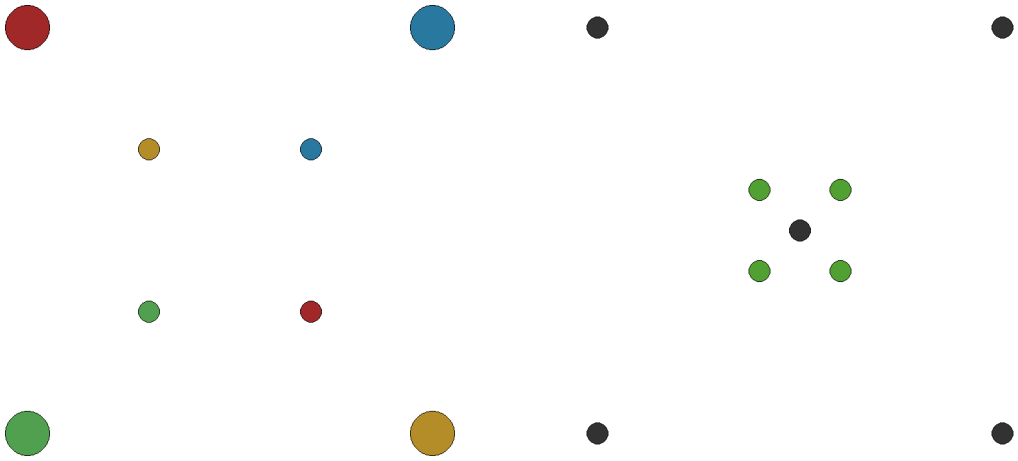

Figure 1: multi-color-spread

Figure 2: 4vs5-spread

Table 1: MARL Environments and Key Attributes Used in the Project

MPE Environments	**Disjoint Obs.**	**Different Sizes (Actions/Obs.)**	**Agents**	**Relationships**
simple_spread	No	No	Homogeneous	Cooperative
simple_spread_discrete	No	No	Homogeneous	Cooperative
simple_reference	No	No	Homogeneous	Cooperative
simple_speaker_listener	Yes	Yes	Heterogeneous	Cooperative
multi_color_spread	No	No	Homogeneous	Cooperative
4vs5_spread	No	No	Homogeneous	Cooperative

3.3 Algorithms

In this project, we utilize commonly used algorithms, MAPPO and IPPO, as baselines for our experiments. For MAPPO, we consider two variations: RMAPPO, where the actor and critic networks use recurrent neural networks , and MAPPO, where multi-layer perceptrons are used

instead. IPPO, on the other hand, uses RNNs by default. These three algorithms are employed to evaluate the impact of parameter sharing and to showcase its performance across various scenarios.

3.4 Hyperparameters and Evaluations

For nearly all experiments, we used the same hyperparameters as suggested by the authors of MAPPO [25]. One exception is in the 4vs5-Spread environment, where the clip epsilon for MAPPO was set to 0.4 to encourage exploration due to the environment's sparse reward structure. Each run was executed three to five times due to the long training time and limited time of the project. Since all the environments used in this project are fully cooperative with agents sharing the same reward, we evaluated performance based on the reward of one agent in each run. The shaded regions in the performance curves represent the variance across these runs.

4 Results

In our experiments, we compare the performance of parameter sharing across various dimensions and environments. The results for each environment are presented. Below, we analyze the outcomes from different perspectives to highlight key findings and insights.

4.1 Factors in Parameter Sharing Performance

4.1.1 Agent indication

Study description. Agent indication, a technique that incorporates the unique index of each agent into its observation or state, has been widely used in parameter sharing to enable the specialization of policies. This study aims to validate the extent to which agent indication can enhance the effectiveness of parameter sharing and support the specialization of agents' behavior.

Interpretation. Fig. 4.1.1 illustrates the performance of agent indication across different environments. The experiments compare three variations of RMAPPO: standard RMAPPO with full parameter sharing (rmappo-shared), RMAPPO without parameter sharing (rmappo-ind), and RMAPPO with full parameter sharing and agent IDs appended to observations (rmappo-shared+id).

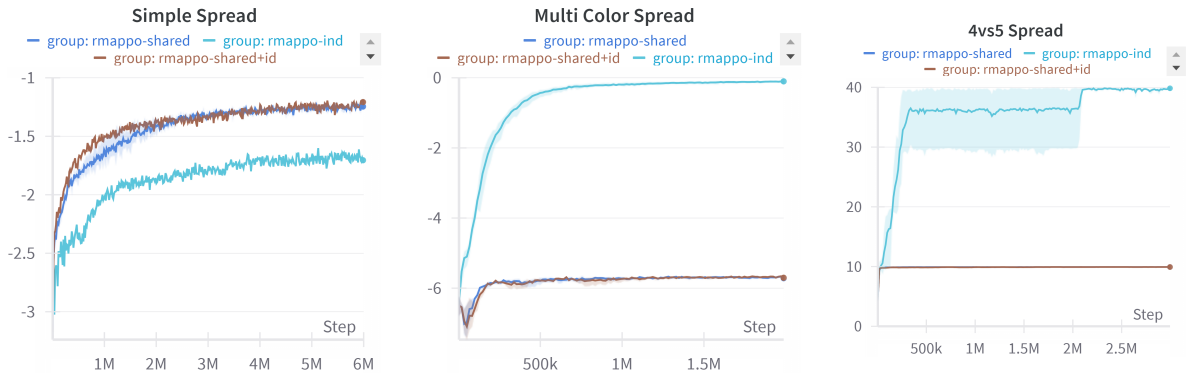

Figure 3: The performance of adding agent indication in different environments.

The results show that, in all three environments, RMAPPO-shared+id exhibits no noticeable improvement over rmappo-shared, whether or not parameter sharing is useful in this scenario. This finding is in slight contrast to previous work by [1], which demonstrated in their experiments that agent indication could enhance performance in other environments. The discrepancy suggests that the effectiveness of agent indication is environment-dependent and may not generalize to all settings.

Conclusion. These results indicate that agent indication has limited utility in improving the performance of parameter sharing for specialization. While it may work well in certain environments, it should not be treated as a universal solution for enabling agent specialization.

4.1.2 Different observation/action sizes

Study description. Parameter sharing becomes challenging to implement when dealing with heterogeneous agents that have different observation or action spaces. One straightforward approach to address this issue is by padding zeros to align all observations and actions to the maximum size. While this method theoretically allows for optimal policies, which is shown by [19], no empirical results were provided to validate its effectiveness. In this experiment, we use the Simple Speaker-Listener environment, where two heterogeneous agents with completely different observation and action spaces learn to cooperate, to empirically test the padding approach.

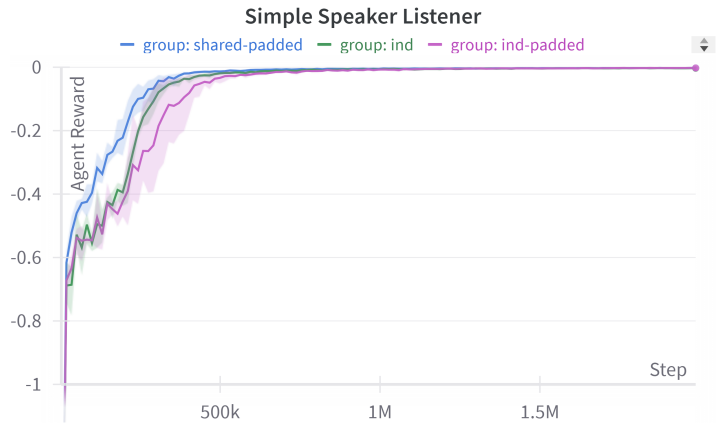

Figure 4: Agent Reward on Simple Speaker Listenr

Interpretation. We use the standard RMAPPO as the baseline and extend its implementation to support parameter sharing. Three configurations are compared: (1) RMAPPO without parameter sharing (ind), (2) RMAPPO without parameter sharing but with padded observation and action spaces (ind-padded), and (3) RMAPPO with full parameter sharing and padded spaces (shared-padded).

As shown in Fig. 4, while padding slightly hinders the performance of independent learning (ind-padded compared to ind), it enables the use of parameter sharing (shared-padded), which learns significantly faster without compromising overall performance. This suggests that padding makes it feasible to apply parameter sharing in heterogeneous environments, providing a practical and efficient solution for multi-agent reinforcement learning with different observation and action spaces.

Conclusion. The results demonstrate that padding observation and action spaces is a viable approach to enable parameter sharing for heterogeneous agents. While it may introduce minor obstacles for independent learning, the substantial efficiency and learning speed gains from parameter sharing outweigh these drawbacks. Padding thus offers a convenient and effective means to generalize parameter sharing to scenarios with diverse agent configurations.

4.1.3 Neural Networks

Study description. Little has been explored about the impact of the choice of neural network previously. However, our experiments reveal that this choice plays a significant role when using parameter sharing, suggesting it is a factor worthy of further investigation.

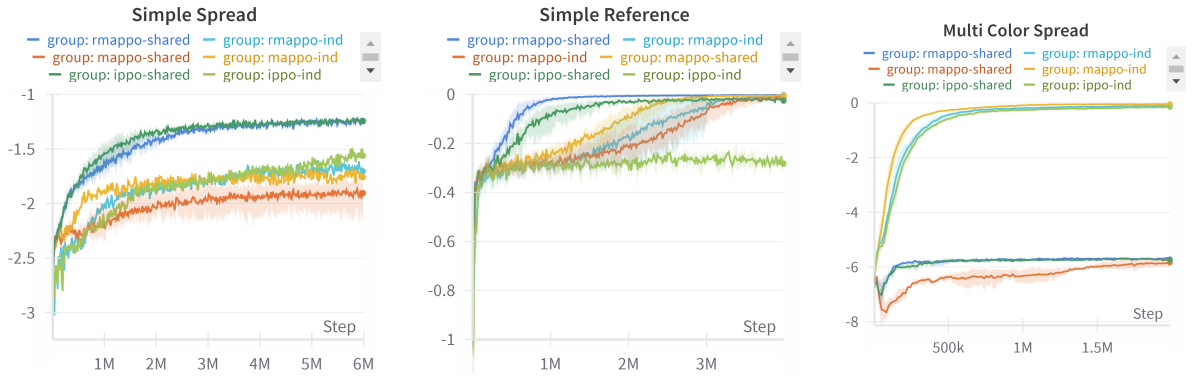

Figure 5: The performance of different neural network in different environments.

Interpretation. In these experiments, three pairs of algorithms were evaluated, comparing cases with parameter sharing (shared) and without parameter sharing (ind). Surprisingly, we found that the use of RNNs is a critical factor for the success of parameter sharing. In the Simple Spread environment, MAPPO with parameter sharing (MAPPO-shared) not only failed to outperform other parameter-sharing algorithms like RMAPPO-shared and IPPO-shared but also exhibited the worst performance of all algorithms. Similarly, in the Simple Reference environment, MAPPO-shared displayed significantly slower convergence compared to RMAPPO-shared and IPPO-shared. Finally, in the Multi-Color Spread environment, MAPPO-shared once again showed the worst performance among all tested configurations.

This unexpected outcome highlights a previously underexplored issue: RNNs appear to be a key factor for effective parameter sharing in MARL. One possible explanation is that RNNs, by maintaining a hidden state across time steps, are better suited for handling the temporal dependencies and non-stationarity inherent in multi-agent environments. This capability may allow them to better generalize across agents sharing the same network parameters, promoting coordination and synchronization. Conversely, MLPs, lacking this temporal context, may struggle to achieve the same level of performance under parameter sharing.

Conclusion. These findings indicate that the choice of neural network architecture has a profound impact on the effectiveness of parameter sharing. While RNNs significantly enhance the performance of shared policies, MLPs can hinder convergence and overall outcomes. This insight emphasizes the need for further research into the interaction between network architecture and parameter sharing, potentially uncovering new ways to optimize MARL systems.

4.1.4 Fully Identical Agents

Study description. For fully identical agents, it is widely accepted that using parameter sharing could help accelerate the training process. However, sometimes it could have even better performances.

Interpretation. The results are shown in the first two environments, as depicted in Fig. 4.1.3. It is surprising that, in the Simple Spread environment, using parameter sharing with rnn not only accelerates the training process but also leads to a higher final performance compared to independent policies. This indicates that parameter sharing helps synchronize the learning process among agents, potentially allowing them to better coordinate their actions and achieve improved overall outcomes.Similarly, in the Simple Reference environment, the shared parameter approach demonstrates stable convergence and higher efficiency during training. This reinforces the idea that parameter sharing is particularly effective when agents are fully identical, as it simplifies the optimization landscape and promotes collective learning.

Conclusion. The experiments in these environments clearly demonstrate the advantages of parameter sharing for fully identical agents. Beyond accelerating training, parameter sharing enables agents to better synchronize their strategies, even possibly resulting in improved performance in cooperative scenarios. These findings suggest that parameter sharing is a highly effective method for environments with homogeneous agents and with same goals and could be a strong baseline for similar setups.

4.1.5 Specialized Roles

Study description. While parameter sharing has been widely used in complex multi-agent environments such as StarCraft Multi-Agent Challenge (SMAC)[15], its ability to differentiate roles among agents remains doubted. To investigate this, we designed a manual scenario called Multi-Color Spread, where agents are rewarded only if they locate the corresponding color landmark. This setup explicitly requires agents to specialize in their roles and cooperate effectively to maximize the group reward.

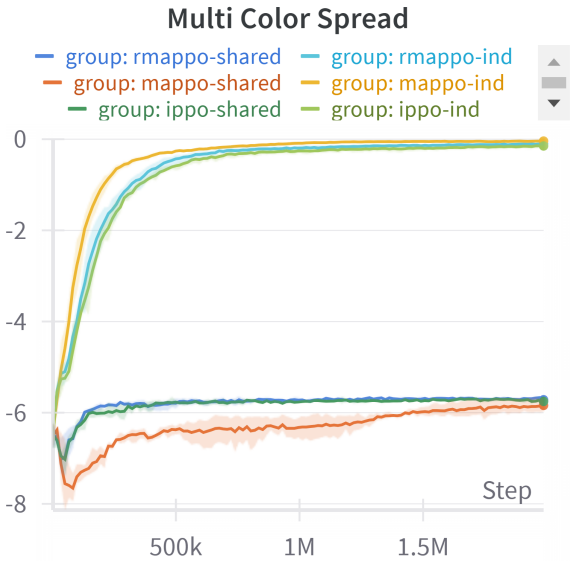

Figure 6: Agent Reward on Multi Color Spread Environment

Interpretation. As shown in Fig. 6, the performance of all parameter-sharing algorithms consistently lagged behind their independent counterparts in the Multi-Color Spread environment. While the independent methods learned to effectively locate the correct landmarks and achieve higher rewards, the parameter-sharing algorithms failed to develop the specialized behaviors necessary for task success. This indicates that the shared agents struggled to differentiate their roles, resulting in suboptimal cooperative strategies. Even if the addition of agent indication (as shown in Fig. 4.1.1) did not improve this outcome.

Conclusion. These findings highlight a key limitation of parameter sharing: its limited ability to promote role differentiation in environments that require agents to specialize in their tasks. While parameter sharing offers benefits such as faster learning and coordination, it is insufficient for scenarios that demand some distinct, role-specific behaviors. This limitation arises because agents using parameter sharing often exhibit similar behaviors. This challenge is consistent with findings from [9], where in Google Football environments, players on the same team may competed for the ball instead of adopting complementary roles.

4.1.6 Sparse Reward

Study description. Exploration has been a pivotal factor in multi-agent reinforcement learning, especially in environments with sparse rewards. The intuition is that parameter sharing can hinder diversity among agents' policies, thus making effective exploration more challenging. To investigate this, we use two environments: 4vs5 Spread and Simple-Spread-Discrete, where each landmark provides an immediate +10 reward if it is "occupied" by an agent. This sparse reward setting makes exploration a critical component for achieving success.

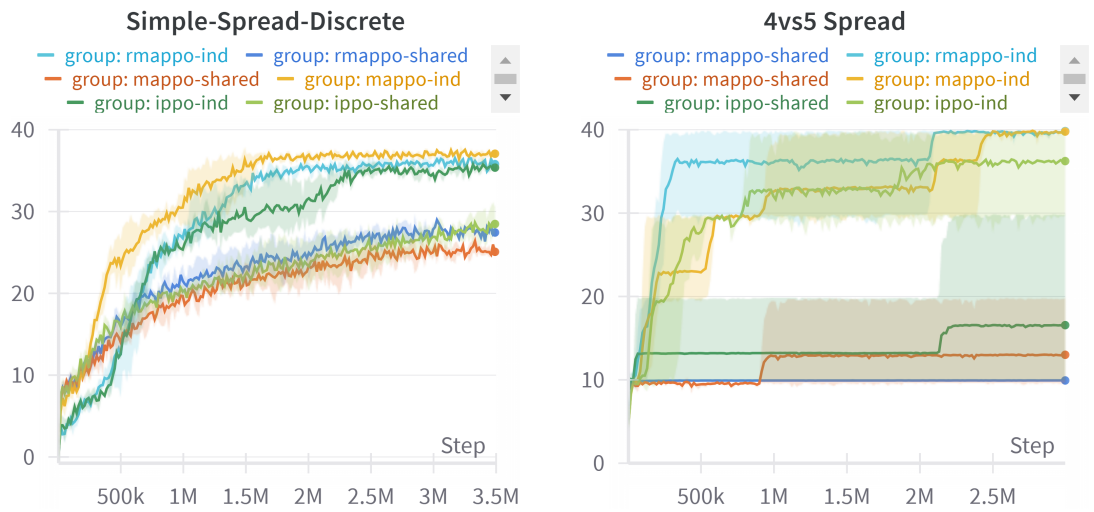

Figure 7: The agent rewards in sparse reward environments.

Interpretation. As shown in Fig.4.1.6, parameter sharing hinders exploration in both environments. In the 4vs5 Spread environment , RMAPPO-ind and IPPO-ind significantly outperform

their shared counterparts. The independent versions exhibit better exploration capabilities, allowing agents to discover optimal strategies for occupying landmarks despite the sparse reward setting. Similarly, in the Simple-Spread-Discrete environment, RMAPPO-ind demonstrates a clear advantage over RMAPPO-shared, achieving faster convergence and higher final rewards. The shared policy structure appears to restrict agents from independently exploring the environment, further validating the hypothesis that parameter sharing can impede exploration in sparse reward settings.

Conclusion. The results highlight a significant drawback of parameter sharing in environments with sparse rewards: its tendency to limit policy diversity, which is crucial for effective exploration. Independent policies, by contrast, allow agents to explore more freely and discover optimal strategies in challenging settings. These findings suggest that while parameter sharing is beneficial for coordination, it may require complementary mechanisms to mitigate its negative impact on exploration in sparse reward scenarios.

4.2 Overall Performance

The overall average reward performance across all environments is summarized in Table 2. Note that rmappo-ai in the table represents rmappo with agent indication. For a more detailed analysis, including insights into convergence steps and standard deviations, we encourage readers to refer to the earlier learning curve plots.

	rmappo-shared	**rmappo-ind**	**rmappo-ai**	**mappo-shared**	**mappo-ind**	**ippo-shared**	**ippo-ind**
simple-spread	**-1.25**	-1.69	**-1.25**	-1.91	-1.75	**-1.25**	-1.61
simple-spread-discrete	27.33	**35.78**	23.57	24.96	**36.94**	28.62	**35.65**
simple-reference	**-0.00**	-0.02	**-0.00**	-0.01	-0.03	-0.02	-0.27
simple-speaker-listener	-0.00	-0.00	-	-	-	-	-
multi-color-spread	-5.69	**-0.12**	-5.68	-5.91	**-0.05**	**-0.17**	-5.72
4vs5-spread	9.92	**39.84**	9.92	13.00	**39.77**	16.56	**36.23**

Table 2: Comparison of different methods across scenarios.

5 Conclusions

This project provides a comprehensive analysis of parameter sharing in multi-agent reinforcement learning, highlighting its effectiveness in cooperative environments while addressing its limitations in tasks requiring role differentiation and exploration. A key finding is the critical role of neural network architectures, with RNNs significantly outperforming MLPs in shared policy settings due to their ability to handle temporal dependencies. Additionally, mechanisms like agent indication and observation/action space padding were evaluated, showing mixed effectiveness depending on the environment's complexity. These insights underline the importance of aligning parameter-sharing strategies with the specific demands of a given task or environment. Parameter sharing should not be treated as a one-size-fits-all approach. Instead, it must be tailored to the unique characteristics of the environment and task requirements.

To build on the findings of this project, future work will focus on validating the results across a broader range of environments to ensure their generalizability. And in addition to policy gradient methods, value-based approaches such as QMIX should be incorporated to investigate whether similar trends hold under different algorithmic paradigms. Moreover, the impact of communication mechanisms on the effectiveness of parameter sharing needs to be systematically explored, as communication could play a critical role in enhancing coordination and diversity. Finally, a deeper investigation into why RNNs play such a pivotal role in parameter sharing is essential, potentially uncovering insights into how temporal dependencies and non-stationarity affect shared policies. These efforts could provide a more comprehensive understanding of parameter sharing in multi-agent reinforcement learning and its broader applicability.

6 Individual tasks

This project represents a significant portion of my ongoing research under the guidance of Professor Christopher Amato. All aspects of the project, including the modification of the algorithm code, the design of custom environments, and the execution of experiments, were carried out during the timeframe since the release of Homework 3. Specifically, the environments Multi-Color-Spread and 4vs5-Spread were designed from the ground up to address challenges such as role special-

ization and sparse rewards. Additionally, the analysis of parameter sharing, including the study of agent indication, neural network architectures, and handling heterogeneous observation/action spaces, was conducted entirely during this project.

References

[1] Filippos Christianos et al. Scaling multi-agent reinforcement learning with selective parameter sharing. In *International Conference on Machine Learning*, 2021.

[2] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, 2016.

[3] Jakob Foerster et al. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

[4] Wei Fu, Chao Yu, Zelai Xu, Jiaqi Yang, and Yi Wu. Revisiting some common practices in cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2206.07505*, 2022.

[5] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops, Best Papers, São Paulo, Brazil, May 8-12, 2017, Revised Selected Papers 16*, pages 66–83. Springer, 2017.

[6] Shariq Iqbal, Christian A Schroeder De Witt, Bei Peng, Wendelin Böhmer, Shimon Whiteson, and Fei Sha. Randomized entity-wise factorization for multi-agent reinforcement learning. In *International Conference on Machine Learning*, 2021.

[7] Fabian Konstantinidis, Moritz Sackmann, Oliver De Candido, Ulrich Hofmann, Jörn Thielecke, and Wolfgang Utschick. Parameter sharing reinforcement learning for modeling multi-agent driving behavior in roundabout scenarios. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 1974–1981. IEEE, 2021.

[8] Chenghao Li et al. Celebrating diversity in shared multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

[9] Chenghao Li, Tonghan Wang, Chengjie Wu, Qianchuan Zhao, Jun Yang, and Chongjie Zhang. Celebrating diversity in shared multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34:3991–4002, 2021.

[10] Dapeng Li et al. Adaptive parameter sharing for multi-agent reinforcement learning. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024.

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

[12] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Neural Information Processing Systems (NIPS)*, 2017.

[13] Volodymyr Mnih. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[14] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, 2018.

[15] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft Multi-Agent Challenge. *CoRR*, abs/1902.04043, 2019.

[16] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

[17] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 2018.

[18] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, 1993.

[19] Justin K. Terry et al. Revisiting parameter sharing in multi-agent deep reinforcement learning. *arXiv preprint arXiv:2005.13625*, 2020.

[20] Jingwei Wang, Qianyue Hao, Wenzhen Huang, Xiaochen Fan, Zhentao Tang, Bin Wang, Jianye Hao, and Yong Li. Dyps: Dynamic parameter sharing in multi-agent reinforcement learning for spatio-temporal resource allocation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3128–3139, 2024.

[21] Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. Roma: Multi-agent reinforcement learning with emergent roles. *arXiv preprint arXiv:2003.08039*, 2020.

[22] Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. Rode: Learning roles to decompose multi-agent tasks. *arXiv preprint arXiv:2010.01523*, 2020.

[23] Tonghan Wang, Jianhao Wang, Chongyi Zheng, and Chongjie Zhang. Learning nearly decomposable value functions with communication minimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[24] Mingyu Yang, Jian Zhao, Xunhan Hu, Wengang Zhou, Jiangcheng Zhu, and Houqiang Li. Ldsa: Learning dynamic subtask assignment in cooperative multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 35:1698–1710, 2022.

[25] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. In *Advances in Neural Information Processing Systems*, volume 35, pages 24611–24624, 2022.

[26] Tianhao Zhang, Yueheng Li, Chen Wang, Guangming Xie, and Zongqing Lu. Fop: Factorizing optimal joint policy of maximum-entropy multi-agent reinforcement learning. In *International Conference on Machine Learning*, 2021.