

Improving Single and Multi-Agent Deep Reinforcement Learning Methods

Tarun Gupta

Exeter College
University of Oxford

*A thesis submitted for the degree of
Doctor of Philosophy*

Michaelmas Term 2023

Abstract

Reinforcement Learning (RL) is a framework where an agent learns to make decisions using data-driven feedback from interactions with the environment in the form of rewards or penalties for actions. Deep RL integrates deep learning with RL, harnessing the power of deep neural networks to process complex, high-dimensional data. Using the framework of deep RL, our machine learning research community has achieved tremendous progress in enabling machines to make sequential decisions over long time horizons. These advances include attaining super-human performance in Atari [Mnih et al., 2015], mastering the game of Go, beating the human world champion [Silver et al., 2017], providing robust recommendation systems [Gomez-Uribe and Hunt, 2015, Singh et al., 2021]. This thesis focuses on identifying some key challenges that impede the learning of RL agents within their specific environments and improving the methods leading to better performance of agents, improved sample efficiency, and generalizability of learned agent policies.

In Part I of the thesis, we focus on exploration in single-agent RL settings where an agent must interact with a complex environment to pursue a goal. An agent that fails to explore its environment is unlikely to achieve high performance, as it will miss critical rewards and, as a result, cannot learn optimal behavior. One key challenge arises from sparse reward environments where the agent only receives feedback once the task is completed, making exploration more challenging. We propose a novel method that enables semantic exploration, resulting in higher sample efficiency and performance on sparse reward tasks.

In Part II of the thesis, we focus on cooperative Multi-Agent Reinforcement Learning (MARL), an extension of the usual RL setting, where we consider multiple agents interacting in the same environment toward a shared task. In multi-agent tasks requiring significant coordination among agents with strict penalties for miscoordination, state-of-the-art MARL methods often fail to learn useful behaviors as agents get stuck in a sub-optimal equilibrium. Another challenge is exploration in the joint action space of all agents, which grows exponentially with the number of agents. To address these challenges, we propose innovative approaches like universal value exploration and scalable role-based learning. These methods facilitate improved coordination among agents, faster exploration, and enhance the agents' ability to adapt to new environments and tasks, showcasing zero-shot generalization capabilities and resulting in higher sample efficiency. Lastly, we investigate independent policy-based methods in cooperative MARL, where each agent considers other agents as part of the environment. We show that such methods can perform better than state-of-the-art joint learning approaches on a popular multi-agent benchmark.

In summary, the contributions of this thesis significantly improve the state-of-the-art in deep (multi-agent) reinforcement learning. The agents developed in this thesis can explore their environments efficiently to improve sample efficiency, learn tasks that require significant multi-agent coordination, and enable zero-shot generalization across various tasks.

Improving Single and Multi-Agent Deep Reinforcement Learning Methods



Tarun Gupta
Exeter College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Michaelmas Term 2023

Dedicated to my family

Acknowledgements

I am grateful to my supervisor, Shimon Whiteson for his unwavering support, guidance, and encouragement throughout my PhD journey. His insights and expertise have been invaluable, and I feel fortunate to have had the opportunity to work under his mentorship. I extend my heartfelt thanks to my thesis examiners, Michael Osborne and Christopher Amato, for their thorough review, constructive feedback, and the stimulating discussions that greatly enhanced the quality of this thesis. The financial support from the Clarendon Scholarship has been instrumental in allowing me to pursue my research without financial concerns. My heartfelt thanks go to Exeter College for providing a supportive and enriching social environment during my time at Oxford.

My gratitude also goes to all my collaborators and colleagues at the Whiteson Research Lab: Kristian Hartikainen, Risto Vuorio, Jacob Beck, Benjamin Ellis, Zheng Xiong, Nasma Dasser, Matthew Jackson, Mingfei Sun, Mattie Fellows, Bei Peng, Wendelin Boehmer, Jelena Luketina, Matt Smith, Charline Le Lan, Vitaly Kurin, Anuj Mahajan, Luisa Zintgraf, Shangtong Zhang, Tabish Rashid, Christian Schroeder de Witt, Maximilian Igl, Greg Farquhar. Working alongside such brilliant and inspiring individuals has been a privilege and they have been an integral part of this journey.

I would like to acknowledge the mentorship and support I received during my internships. At Google Waymo, I am grateful to Maximilian Igl, Shimon Whiteson and the rest of the team for their guidance and for providing an invaluable learning experience. My time at NVIDIA Research was profoundly impactful, and I extend my thanks to Marco Pavone, Peter Karkus, Tong Che, Danfei Xu, and the rest of the team for their mentorship and support. At Microsoft Research, I am deeply thankful to Anssi Kanervisto, David Bignell, Gunshi Gupta, Raluca Georgescu, Sam Devlin, Sergio Valcarcel Macua, Shan Zheng Tan, Tabish Rashid, Tim Pearce, Yuhao Cao, Udit Arora, Katja Hofmann, Cheng Zhang for their encouragement and the opportunity to grow as a researcher.

On a personal note, I am immensely grateful to my family for their unconditional love, support, and belief in me. Their encouragement has been the bedrock of my academic and personal achievements. Finally, a special thanks to my close friends who have provided emotional support and understanding throughout this journey. Your encouragement and love have been my motivation and strength. To everyone who has played a part in this journey, your contributions have been invaluable, and I am eternally grateful.

Abstract

Reinforcement Learning (RL) is a framework where an agent learns to make decisions using data-driven feedback from interactions with the environment in the form of rewards or penalties for actions. Deep RL integrates deep learning with RL, harnessing the power of deep neural networks to process complex, high-dimensional data. Using the framework of deep RL, our machine learning research community has achieved tremendous progress in enabling machines to make sequential decisions over long time horizons. These advances include attaining super-human performance in Atari [Mnih et al., 2015], mastering the game of Go, beating the human world champion [Silver et al., 2017], providing robust recommendation systems [Gomez-Uribe and Hunt, 2015, Singh et al., 2021]. This thesis focuses on identifying some key challenges that impede the learning of RL agents within their specific environments and improving the methods leading to better performance of agents, improved sample efficiency, and generalizability of learned agent policies.

In Part I of the thesis, we focus on exploration in single-agent RL settings where an agent must interact with a complex environment to pursue a goal. An agent that fails to explore its environment is unlikely to achieve high performance, as it will miss critical rewards and, as a result, cannot learn optimal behavior. One key challenge arises from sparse reward environments where the agent only receives feedback once the task is completed, making exploration more challenging. We propose a novel method that enables semantic exploration, resulting in higher sample efficiency and performance on sparse reward tasks.

In Part II of the thesis, we focus on cooperative Multi-Agent Reinforcement Learning (MARL), an extension of the usual RL setting, where we consider multiple agents interacting in the same environment toward a shared task. In multi-agent tasks requiring significant coordination among agents with strict penalties for miscoordination, state-of-the-art MARL methods often fail to learn useful behaviors as agents get stuck in a sub-optimal equilibrium. Another challenge is exploration in the joint action space of all agents, which grows exponentially with the number of agents. To address these challenges, we propose innovative

approaches like universal value exploration and scalable role-based learning. These methods facilitate improved coordination among agents, faster exploration, and enhance the agents' ability to adapt to new environments and tasks, showcasing zero-shot generalization capabilities and resulting in higher sample efficiency. Lastly, we investigate independent policy-based methods in cooperative MARL, where each agent considers other agents as part of the environment. We show that such methods can perform better than state-of-the-art joint learning approaches on a popular multi-agent benchmark.

In summary, the contributions of this thesis significantly improve the state-of-the-art in deep (multi-agent) reinforcement learning. The agents developed in this thesis can explore their environments efficiently to improve sample efficiency, learn tasks that require significant multi-agent coordination, and enable zero-shot generalization across various tasks.

Contents

List of Figures	xv
List of Tables	xix
1 Introduction	1
1.1 Motivation	1
1.1.1 Challenges in RL	4
1.1.2 Challenges in Multi-Agent RL	5
1.2 Thesis Structure and Contributions	7
2 Background	11
2.1 Reinforcement Learning	11
2.1.1 Q-Learning	14
2.1.2 Policy Gradient	15
2.2 Cooperative Multi-Agent RL	16
Part I : Semantic Exploration in RL	18
3 Language Guided Exploration	21
3.1 Introduction	22
3.2 Background	25
3.2.1 Exploration in Deep RL	25
3.2.2 Foundation Models in RL	30
3.2.3 IMPALA	31
3.2.4 CLIP	32
3.2.5 Problem Formulation	33
3.3 FoMoRL	33
3.3.1 Semantic Intrinsic Motivation	33
3.3.2 Training Setup	35
3.4 Experiments	36

3.4.1	MiniGrid Domain	36
3.4.2	Baseline	37
3.4.3	Results	38
3.4.4	CLIP Analysis on MiniGrid	41
3.5	Further Discussions and Limitations	44
3.5.1	Promoting Novelty in Pixel Space vs CLIP Representations	44
3.5.2	Discussion on the choice of foundation model	45
3.5.3	Limitations	46
3.6	Conclusion and Future Work	47
Part II : Cooperative Multi-Agent RL		48
4	Brief Survey of Cooperative Multi-Agent RL	51
4.1	Independent Learners	53
4.2	Centralized Execution	55
4.3	Communication	56
4.4	Centralized Training with Decentralized Execution (CTDE)	57
4.4.1	Value Function Factorization	57
4.4.2	Actor-Critic Methods	62
4.5	Diversity in MARL	64
Part II(A) : Cooperative Value-Based Multi-Agent RL		66
5	Universal Value Exploration	69
5.1	Introduction	70
5.2	Relative Overgeneralization	71
5.3	Background on Successor Features	72
5.3.1	Task based Universal Value Functions	72
5.3.2	Successor Features	73
5.3.3	Universal Successor Features and Generalized Policy Improvement	73
5.4	Illustrative Example	75
5.5	Multi-Agent Universal Successor Features	76
5.5.1	Decentralized Execution	78
5.5.2	Decentralized Local GPI	78
5.5.3	Training	79
5.6	UneVEn	80

5.6.1	Action-Selection Schemes	81
5.7	Related Work on Multi-Agent RL	83
5.7.1	Exploration	83
5.8	Related Work on Successor Features	84
5.9	Experiments	85
5.9.1	Domain 1 : m-Step Matrix Game	85
5.9.2	Domain 2 : Cooperative Predator-Prey	86
5.9.3	Domain 3 : Starcraft Multi-Agent Challenge (SMAC)	93
5.10	Conclusion, Limitations and Future Work	99
6	Role Based Exploration	101
6.1	Introduction	102
6.2	RODE Learning Framework	104
6.2.1	Determining Role Action Spaces by Learning Action Repre- sentations	105
6.2.2	Learning the Role Selector and Role Policies	107
6.3	Related Work	110
6.3.1	Hierarchical MARL	110
6.3.2	Dealing with Large Discrete Action Spaces	111
6.3.3	Role-Based Learning	112
6.4	Experiments	113
6.4.1	Action Representations	114
6.4.2	Performance and Ablation Study	115
6.4.3	Case Study : Role Dynamics	118
6.4.4	Effect of Role Interval	120
6.4.5	Effect of Role Selector Architecture	121
6.4.6	Policy Transfer	122
6.4.7	Role Representations	124
6.5	Analysis on Clustering Algorithms	124
6.5.1	Influence of Number of Clusters k	125
6.5.2	Other Clustering Algorithms	126
6.6	Conclusion and Future Work	127
	Part II(B) : Cooperative Policy-Based Multi-Agent RL	129

7	Independent PPO for SMAC	133
7.1	Introduction	133
7.2	Related Work	136
7.3	Background	137
7.3.1	Trust Region Policy Optimization	138
7.3.2	Independent Actor-Critic	138
7.4	Independent PPO	139
7.4.1	Value Clipping	140
7.4.2	Learning Architecture	141
7.5	Empirical Results	141
7.5.1	IPPO Performance	142
7.5.2	Role of Centralised Value Functions	143
7.5.3	Ablation Studies	143
7.6	Discussion	145
7.7	Conclusion and Future Work	148
8	Afterword	151
8.1	Broader Impact	152
Appendices		
A	Language Guided Exploration	157
A.1	Additional Results	157
B	Universal Value Exploration	159
B.1	Formal derivation of the illustrative example in Section 5.4	159
B.2	Training algorithm	161
B.3	Implementation Details	161
B.3.1	Hyper parameters	161
B.3.2	NN Architecture	163
B.4	Additional Results	164
B.4.1	Predator-Prey	164
B.4.2	StarCraft II Micromanagement	165

C Role Based Exploration	167
C.1 Architecture, Hyperparameters, and Infrastructure	167
C.2 Experimental Details	169
C.2.1 Benchmarking on StarCraft II Micromanagement Tasks . . .	169
C.2.2 Policy Transfer	170
D Independent PPO for SMAC	173
D.1 Additional Results	173
Bibliography	177

List of Figures

2.1	Graphical model representation of MDP.	13
3.1	Example of a fully observable view of an agent (marked with red triangle) in an obstructed maze from MiniGrid Environment [Chevalier-Boisvert et al., 2018] learning to navigate to the goal location (marked with the blue ball). The shaded part shows the partially observable image that the agent can see. Note that the agent cannot see past locked doors.	22
3.2	Training Architecture for CLIP’s visual and language encoder with a contrastive loss directly taken from Radford et al. [2021]. The visual representations of Image I_1 are trained to be <i>close</i> to its corresponding caption representations T_1 but <i>far</i> from other caption representations $T_2, \dots, T_N$ and vice-versa.	32
3.3	An illustration of different components of FoMoRL.	35
3.4	Representative examples of different MiniGrid environments. The highlighted area showcases the partial observations the agent receives.	36
3.5	A few examples of the fully observable view (top) vs corresponding partially observable view (bottom) available to the agent.	38
3.6	An example rollout of an agent in very hard exploration task MultiRoom-N10-S10 where agent has learnt to generate diversity within the partially observable restricted view by changing directions in a cell and then moving to the next cell. In the full rollout, the agent is able to move to the next room but soon runs out of time to solve the task.	40
3.7	RGB Input Image for CLIP Embedding Analysis.	42

3.8	Similarity scores for proposed language descriptions of the input test image. (a) Test to check whether CLIP representations prefer a description with the presence of a green square (incorrect) or a grey square (correct) in the test image. (b) Test to check the count of blue squares in the test image. (c) Test to check if CLIP representations prefer the description with spatial information of the key in the the test image or not. (d) Test to check more fine-grained understanding of the test image by comparing against the shapes present within the yellow square in the test image.	43
5.1	Simplified predator-prey environment (left), where two agents are only rewarded when they both stand next to prey (right).	75
5.2	Task space of Figure 5.1. Tasks solvable under uniform exploration ($\pi^2(C s) = \frac{1}{3}$) are green, shades of red represent tasks that on average exhibit RO for different $\pi^2(C s)$	76
5.3	UneVEn works by evaluating random tasks (blue dots), and exploring already solved (magenta dots). This increases the fraction of observed $u^2 = C$, and thus the set of solvable tasks, which eventually reaches the <i>target task</i> (cross).	77
5.4	Schematic illustration of the MAUSFs training and UneVEn exploration with GPI policy.	79
5.5	m -step matrix game from Mahajan et al. [2019] for $m = 10$. The red cross means that selecting that joint action will lead to termination of the episode.	85
5.6	SOTA comparison on the m -step matrix game ($m = 10$).	86
5.7	Baseline predator-prey results without RO ($p = 0$).	89
5.8	Comparison between UneVEn and SOTA MARL baselines with $p \in \{0.004, 0.008, 0.012, 0.016\}$	90
5.9	Ablation results. Comparison between different action selection schemes of UneVEn for $p \in \{0.012, 0.016\}$	92
5.10	Zero-shot generalization comparison between UneVEn and SOTA MARL baselines; training on $p = 0.004$, testing on $p \in \{0.2, 0.5\}$	92
5.11	Representative images of a few maps from the SMAC environment taken from Samvelyan et al. [2019].	94
5.12	Comparison between VDN and QMIX baselines with original positive reward function and modified reward function with $p \in \{0.0, 0.5\}$ on SMAC maps.	96

5.13	Comparison between VDN and QMIX baselines with original positive reward function and modified reward function with $p \in \{0.0, 0.5\}$ on the remaining SMAC maps.	97
5.14	Comparison between UneVEn and SOTA MARL baselines on SMAC maps with $p = 1.0$	98
6.1	RODE framework. (a) The forward model for learning action representations. (b) Role selector architecture. (c) Role action spaces and role policy structure.	106
6.2	Action representations learned by our predictive model on corridor (left) and 3s5z_vs_3s6z (right). Blue arrows indicate the directions moving towards enemies while red ones show the directions moving away from enemies.	113
6.3	Performance comparison with baselines on <i>all</i> super hard maps and one hard map (2c_vs_64zg). In Appendix C.2.1, we show results on the whole benchmark.	115
6.4	The number of scenarios (out of all 14 scenarios) in which the algorithm’s median test win % is the highest by at least 1/32. . . .	116
6.5	Ablation studies regarding each component of RODE. The best performance that can be achieved by QMIX is shown as horizontal dashed lines.	117
6.6	Visualization of roles in one episode on corridor . The first row shows game snapshots. In the second row, black circles are enemies and circles in other colors are ally agents, with different colors indicating different assigned roles. The size of circles is proportional to the remaining health points of units. The remaining and maximum health points are also shown in the lower right corner of the figures.	119
6.7	Frequencies of taking each role during training on corridor . Correspond to Fig. 6.6. The frequencies of two roles which help exploration (<i>Role 0</i> and <i>Role 2</i>) first increase and then decrease.	120
6.8	Influence of the role interval (the number of timesteps between two consecutive role selections).	121
6.9	Comparisons between RODE with a recurrent role selector (RODE) and a fully-connected role selector conditioned on local observations (RODE (Fully-Connected Role Selector)).	122

6.10	Win rates on unseen maps of the policy learned on corridor , where 6 ally Zealots face 24 Zerglings. We do not train policies on new maps.	123
6.11	Comparisons between RODE and RODE with shared role policies and role selector. The separation of the role selector and role policies is important to make the average of action representations well represent the roles.	124
6.12	RODE with different k values.	126
6.13	Comparisons between RODE using X-means clustering and RODE using k -means clustering, where $k = 3$ for maps with homogeneous enemies and $k = 5$ for maps with heterogeneous enemies. X-means does not require predefining the number of clusters.	127
7.1	Results on select SMAC maps, comparing IPPO to QMIX and IQL. Uncertainty regions depict $[0.25, 0.75]$ confidence intervals of the median test win rate.	142
7.2	Results on select SMAC maps comparing IPPO and MAPPO. . . .	144
7.3	Ablation study for IPPO with different combinations of policy and value clipping.	145
A.1	Learning curves of RIDE and FoMoRL on a variety of hard exploration problems in MiniGrid. Both methods receive partially observable RGB images as input to the policy and intrinsic reward module.	158
B.1	Additional Ablation results: Comparison between different action selection of UneVEn for $p \in \{0.004, 0.008\}$	164
B.2	Additional Zero-shot generalization results for $p \in \{0.2, 0.3, 0.5, 1.0\}$	164
B.3	Comparison between UneVEn, VDN and QMIX on SMAC maps with penalty $p = 0.5$	165
C.1	Comparisons between RODE and baselines on all super hard maps.	168
C.2	Comparisons between RODE and baselines on all hard maps. . . .	169
C.3	Comparisons between RODE and baselines on all easy maps. . . .	170
D.1	Relative overgeneralization, as experienced by select MARL algorithms on Stag Hunt with penalty -2 (see Böhmer et al. [2020]). Note that DCG is not fully decentralized during execution.	174
D.2	Results on select SMAC maps comparing IPPO and IPPO-C with QMIX, VDN and IQL.	175

List of Tables

3.1	Performance of RIDE and FoMoRL on a variety of hard exploration problems in MiniGrid. Both methods receive partially observable RGB images as input to the policy and intrinsic reward module. The values indicate the final mean episodic return averaged over multiple seeds after training on $[xM]$ frames. The bracketed values (xM) indicate the average number of training frames (in millions) required to achieve the convergence value. The (-) indicates no convergence in $[xM]$ frames.	39
3.2	Performance of FoMoRL on very-hard exploration tasks N10-S10 and N12-S10 in MiniGrid with partially observable RGB images as input to the policy and fully observable RGB images as input to the intrinsic reward module. The values indicate the final mean episodic return averaged over multiple seeds after training on $[xM]$ frames. The bracketed values (xM) indicate the average number of training frames (in millions) required to achieve the convergence value. . . .	41
3.3	Sensitivity of the intrinsic reward coefficient hyper parameter (α) on final performance of FoMoRL. The value 0.0 (-) indicates no convergence after training on fixed number of frames. A dash(-) implies the hyper-parameter was not tested.	41
5.1	Joint-Reward function of three agents surrounding a prey. The first table indicates joint-rewards when Agent 1 takes capture action and second table indicates joint-rewards when Agent 1 takes any other action. Notice that there are numerous joint actions leading to penalty p	88
7.1	Median win test rates on select SMAC maps at 10M steps, comparing IPPO to MAPPO results reported by Yu et al. [2022].	143

7.2	IPPO hyperparameters across different SMAC maps. Fixed hyperparameters: $\gamma = 0.99$, $\lambda = 0.95$, $e_{clip} = 0.2$, $grad\ norm = 0.5$ and $\epsilon = 0.2$. We do not employ any learning rate schedule, and always normalize the advantage function. We fix the number of actors to 8. <i>Net arch</i> indicates successive channels across layers for <i>conv1d</i> architecture, and number of activations for <i>mlp</i> . All convolutions have stride 3.	146
-----	--	-----

1

Introduction

Contents

1.1 Motivation	1
1.1.1 Challenges in RL	4
1.1.2 Challenges in Multi-Agent RL	5
1.2 Thesis Structure and Contributions	7

1.1 Motivation

Artificial Intelligence (AI) is revolutionizing the world in numerous ways, profoundly impacting our daily lives and how businesses operate. Self-driving cars are able to navigate roads, recognize traffic patterns, and make decisions in real-time, potentially reducing accidents due to human error and improving transportation efficiency [Peña, 2023]. A recent AI breakthrough in healthcare has provided a faster, more affordable, and safer method for detecting heart disease, which, once fully validated, could reduce the time patients spend in an MRI scanner by half, cutting scan costs by more than half while yielding clearer, more diagnostically valuable images that are easier to interpret [Zhang et al., 2021]. Moreover, AI applications in healthcare extend to robotic surgery [Nawrat, 2023], where they can assist surgeons with enhanced precision and control, and in drug discovery [Mock et al., 2023], speeding

up the process of finding new medications and therapies. In our day-to-day lives, AI used in intelligent assistants, navigation, and personalized recommendation systems enhances convenience and efficiency in everyday tasks. In discussing the ubiquity of computers in the 21st century, Mark Weiser, known as the father of ubiquitous computing, states, “The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it” [Weiser, 1991]. As AI systems continue to advance, there is potential for AI to become as widespread and integral in our daily lives as computers are today.

The success of AI can be attributed to several key factors. First, the significant reduction of computing costs with smaller, more efficient, and cost-effective microchips has allowed our smartphones in our pockets to run AI algorithms. Some phone devices are getting shipped with neural engines and GPUs, providing unprecedented computing power directly at our fingertips. In contrast, in the early 1950s, only well-known universities and major technology corporations had the resources to venture into these then-uncharted territories. The second factor is the ability to learn directly from data instead of relying on predefined heuristics. This shift marked a transition from classical AI to modern Machine Learning (ML) approaches. For example, classical AI for chess, like IBM’s Deep Blue [Hsu, 2002], used extensive heuristics and rules designed by experts to evaluate positions and make decisions. In contrast, modern chess engines like AlphaZero [Silver et al., 2017] learn to play chess by playing against themselves, analyzing vast amounts of games, and learning strategies without predefined heuristics.

The field of machine learning (ML) has been significantly influenced by advancements in computational power and cost reductions, leading to the development of increasingly large and complex models. Early ML models, such as the perceptron algorithm [Anlauf and Biehl, 1989], were limited to processing linearly separable data due to their simplicity and limited computational requirements. However, the landscape of ML has dramatically changed, as seen with the emergence of models like GPT-4 [OpenAI, 2023], with billions of parameters. This growth in model size

can be attributed not only to cheaper and more powerful computing resources but also to innovations in algorithm design and data processing capabilities. As a result, contemporary deep learning models are equipped to handle intricate, nonlinear data patterns. This transition from shallow to deep learning (DL) has allowed for a more nuanced understanding and processing of data, as seen in complex tasks such as image and speech recognition and natural language processing. Moreover, the availability of vast amounts of human data from a continuous stream of user-generated content on social media, including text, images, and video, has made it easier to train large models capable of learning and exploiting structure in the data. This, in turn, has enabled the models to generalize to unseen data.

The fields of natural language processing (NLP) and computer vision (CV) have achieved significant advancements by utilizing deep neural networks. The evolution of NLP in recent years has been marked by sequence-to-sequence models [Sutskever et al., 2014] pivotal for tasks like machine translation. The integration of memory into these models, such as Gated Recurrent Units (GRUs) [Chung et al., 2014] and Long Short-Term Memory (LSTM) [Memory, 2010], allowed for better context retention over longer text spans. Transformers [Vaswani et al., 2017] marked a significant advancement in NLP. They use attention mechanisms to concentrate on critical parts of the input. Unlike seq-to-seq models, which process data sequentially, transformers process all parts of the sequence in parallel. This parallelism, combined with the attention mechanism, enables transformers to capture long-range dependencies within a sequence effectively. This led to quicker-to-train models, which set higher benchmarks in NLP tasks. Similarly, computer vision has seen remarkable progress [Hassaballah and Hosny, 2019] by pushing the boundaries of object detection and localization, depth estimation, image classification, segmentation, and generation.

Despite big advancements in natural language processing and computer vision, we're still far from having AI that makes decisions like humans and quickly adapts to the real world. Human decisions are often a chain of choices, one leading to the next.

But supervised learning in AI usually looks at decisions one by one, assuming i.i.d data, not as a chain. That is why we need the framework of Reinforcement Learning (RL). RL allows an agent to learn from trial and error as the agent can interact with the environment in its current state and receive feedback for its actions in the form of rewards and the new resulting state. In order to solve the task, value-based RL methods estimate the value of different actions or states, guiding the agent to choose actions that seem most beneficial based on these estimates. In contrast, policy-based RL directly learns the policy that maps states to actions. Deep RL integrates deep learning with RL, harnessing the power of deep neural networks to process complex, high-dimensional data. We next focus on identifying some of the key challenges that impede the learning of RL agents within their specific environments.

1.1.1 Challenges in RL

A long-standing problem in RL is that of exploration. Exploration plays a pivotal role in RL, as an agent must explore its environment to achieve optimal performance. Without exploring, an agent misses out on crucial rewards and might fail to learn the best behavior. The challenge of exploration intensifies as RL is applied to larger and more complex environments. Simple exploration methods that work in smaller settings do not suffice in these expanded scenarios. Therefore, developing sophisticated and targeted exploration strategies is important, especially in Deep RL, where exploration is often the key hurdle in tackling complex problems. One key challenge arises from sparse reward environments where the agent does not receive any feedback from the environment until the task is completed, making exploration more challenging. Imagine an agent in a maze that is learning to reach a *fixed* goal in an environment but does not receive any feedback from the environment until it either reaches the desired goal or runs out of time. The agent has to pick up keys and objects, open doors with corresponding keys, and reach the desired goal. In this case, the agent might have to considerably explore the environment and interact with different objects before finding the goal and receiving a reward.

This process can be time-intensive and demand a lot of computational resources. This inefficiency becomes more evident in intricate or vast environments, where randomly discovering a reward is less probable. Now, additionally, imagine that the environment is also procedurally generated, i.e., the goal, the size, and the components of the maze change every time the agent respawns. In this case, the agent is unlikely to visit a state more than once, and therefore, the learning method needs to generalize to unseen scenarios. In Chapter 3 of this thesis, we focus on such settings and propose a method that can drive exploration towards semantically meaningful states and outperform state-of-the-art methods in challenging sparse-reward procedurally generated environments.

1.1.2 Challenges in Multi-Agent RL

Multi-Agent Reinforcement Learning (MARL) extends to the usual RL setting, where we consider multiple agents interacting in the same environment. These agents must learn complex decision-making in cooperative, competitive, or mixed settings. In Part II of this thesis, we focus on cooperative settings where the agents must learn how to behave sensibly and cooperate with the other agents to solve a shared task. MARL holds considerable promise to help address a variety of complex cooperative decision-making problems arising in the real world, such as autonomous driving vehicles [Cao et al., 2012, Shalev-Shwartz et al., 2016], automated factories [Dotoli et al., 2017], sensor networks [Zhang and Lesser, 2011], coordination of robot swarms [Hüttenrauch et al., 2017], crew less aerial vehicles [Pham et al., 2018].

As we discussed some challenges faced in single-agent RL, MARL suffers from *additional* unique challenges that can impede agents' learning. Learning control policies for MARL is challenging as agents must search the joint action space, which grows exponentially with the number of agents. This makes the task of exploration harder as the transition dynamics of the environment now depend on the actions of all the agents. Moreover, agents have to act solely based on their local observations in a decentralized manner during execution time due to partial

observability and communication constraints with no access to information about other agents. For example, a self-driving car should be able to make safe and sensible decisions based on only what it can see without relying on information from external sources. Applying independent forms of learning where agents ignore other agents in the environment might lead to learning instability arising from a non-stationary environment. On the other hand, learning a centralized controller with joint action of all agents becomes intractable with an exponentially increasing number of joint actions. It does not allow for decentralized execution of agents without expensive communication.

Part II(A) of the thesis focuses on improving value-based MARL methods in cooperative settings with varying complexities. Chapter 5 focuses on multi-agent domains requiring agents to coordinate simultaneously, i.e., multiple agents are required to take a specific action at the same time step simultaneously to avoid penalties, making exploration even harder. We propose universal value exploration, which can solve tasks requiring significant coordination among agents where other state-of-the-art MARL methods fail. Our method provides zero-shot generalization to harder tasks. Chapter 6 focuses on improving multi-agent exploration by proposing a scalable learning method that decomposes the joint action space into restricted role-based action spaces by clustering actions according to their effects on the environment and other agents. This decomposition helps in reducing the joint action search space, simplifying exploration.

In Part II(B) of the thesis, we focus exclusively on the policy-based methods for a well-known cooperative MARL domain named StarCraft Multi-Agent Challenge (SMAC) [Samvelyan et al., 2019]. SMAC is based on the popular real-time strategy game StarCraft II, which focuses on micromanagement challenges where each unit controlled by an independent agent must learn rich cooperative behaviors under partial observability. Chapter 7 investigates independent policy-based methods in cooperative MARL and showcases that such methods can perform better than

state-of-the-art joint learning approaches on the SMAC benchmark despite the purported limitations of independent learning.

1.2 Thesis Structure and Contributions

This section provides a detailed overview of the thesis structure and the specific contributions. In instances where I am listed as a first author, I assume the role of lead author. In this capacity, the contributions include conceptualizing and brainstorming ideas with co-authors, implementing and running experiments, steering the overall research direction of the project, conducting ablation studies, and taking the lead on the writing and presentation aspects of the paper, including its appendices. I want to extend my heartfelt gratitude to my co-authors and colleagues, whose invaluable insights and collaboration have been instrumental in completing this research.

We begin with introducing the background on Reinforcement Learning and Multi-Agent Reinforcement Learning in Chapter 2 and setting up standard notations, which will be helpful throughout the thesis. Additional notations and background specific to each chapter will be introduced within the chapter itself. The remainder of this thesis is divided into two parts. Part I of this thesis focuses on semantic exploration in single-agent sparse reward procedurally generated environments. Chapter 3 proposes a foundation model based intrinsic motivation scheme for RL, FoMoRL, which helps guide exploration towards semantically meaningful states. This work is based on the following paper:

- Foundation Models for Semantic Novelty in Reinforcement Learning.
Tarun Gupta, Peter Karkus, Tong Che, Danfei Xu, Marco Pavone.
NeurIPS 2022. Workshop on Foundation Models and Decision Making.

Part II then focuses on cooperative multi-agent RL. Chapter 4 presents a (non-exhaustive) literature review of cooperative multi-agent RL with a discussion of various methods. Part II(A) of this thesis introduces novel value-based cooperative

MARL algorithms. Chapter 5 introduces Universal Value Exploration (UneVEN) that helps improve the joint exploration of all agents during training, especially for tasks that require significant coordination among agents at a particular time step. Empirical results on a set of exploration games, challenging cooperative predator-prey tasks requiring significant coordination among agents, and SMAC benchmarks show that UneVEN can solve tasks where other state-of-the-art MARL methods fail. This work is based on the following paper:

- UneVEN : Universal Value Exploration for Multi-Agent Reinforcement Learning.

Tarun Gupta, Anuj Mahajan, Bei Peng, Wendelin Böhmer, Shimon Whiteson.
 Proceedings of the 38th International Conference on Machine Learning, PMLR
 139:3930-3941, ICML 2021.

Chapter 6 then introduces role-based learning in cooperative MARL. We propose simplifying multi-agent exploration by decomposing joint action spaces into restricted role action spaces by clustering actions according to their effects on the environment and other agents. Our method outperforms the state-of-the-art MARL algorithms on 9 of the 14 scenarios that comprise the challenging SMAC benchmark and achieves rapid transfer to new environments with three times the number of agents. This work is based on the following paper:

- RODE: Learning Roles to Decompose Multi-Agent Tasks.

Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson,
 Chongjie Zhang.

Proceedings of the Ninth International Conference on Learning Representations, ICLR 2021.

- My contributions to this paper were to help in the ideation process, writing, and helping with implementing and testing MARL domains.

Part II(B) of this thesis focuses on policy-based methods for cooperative MARL. In Chapter 7, we study the empirical performance of Independent PPO (IPPO), a multi-agent variant of the popular PPO algorithm. Despite the purported limitations of independent learning for cooperative MARL tasks, we found that IPPO performs competitively on a range of SMAC benchmark tasks, outperforming state-of-the-art value-based methods on some maps. A number of ablation studies indicated that PPO’s policy clipping objective is crucial to performance, which helps mitigate some forms of environment non-stationarity, rendering fundamental theoretical limitations less critical to practical performance. This work is based on the following paper:

- Is Independent Learning All You Need in the StarCraft Multi-Agent Challenge?
Christian Schroeder de Witt*, Tarun Gupta*, Denys Makoviichuk, Viktor Makoviyshuk, Philip H.S. Torr, Mingfei Sun, Shimon Whiteson.

arXiv preprint arXiv:2011.09533.

* indicates equal contribution (joint first authors).

- I am a co-lead author on this paper. My contributions include implementing and running experiments, guiding the research direction of the project, performing ablation experiments, and co-writing the paper.

Lastly, we conclude this thesis with conclusions and broader impact. During my DPhil, I have also worked on the following papers during internships and otherwise, which have not been included in this thesis:

- Semi-On-Policy Training for Sample Efficient Multi-Agent Policy Gradients.
Bozhidar Vasilev, Tarun Gupta, Bei Peng, Shimon Whiteson.
AAMAS Adaptive and Learning Agents Workshop (ALA) at the 20th International Conference on Autonomous Agents and Multiagent Systems 2021.
- Generalization in Cooperative Multi-Agent Systems.
Anuj Mahajan, Mikayel Samvelyan, Tarun Gupta, Benjamin Ellis, Mingfei

Sun, Tim Rocktäschel, Shimon Whiteson.

arXiv preprint arXiv:2202.00104.

- Latent Hierarchical Imitation Learning for Stochastic Environments.
Maximilian Igl, Punit Shah, Paul Mousin, Sirish Srinivasan, Tarun Gupta, Brandyn White, Kyriacos Shiarlis, Shimon Whiteson.
Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS 2023), Detroit, USA (2023).
- WHAM! World and Human Action Modelling in a Modern Xbox Game.
MSR GAMR, Anssi Kanervisto*, David Bignell*, Gunshi Gupta*, Raluca Georgescu*, Sam Devlin*, Sergio Valcarcel Macua*, Shan Zheng Tan*, Tabish Rashid*, Tim Pearce*, Tarun Gupta*, Yuhao Cao*, Alison Shaw, Gavin Costello, Udit Arora, Katja Hofmann.
Under Review. *Authors listed alphabetically by first name.

2

Background

Contents

2.1	Reinforcement Learning	11
2.1.1	Q-Learning	14
2.1.2	Policy Gradient	15
2.2	Cooperative Multi-Agent RL	16

The following chapter introduces the reader to common background topics useful for the rest of this thesis. We introduce basic concepts in Reinforcement Learning (RL) and cooperative Multi-Agent Reinforcement Learning (MARL).

2.1 Reinforcement Learning

Markov Decision Processes (MDPs) provide a mathematical framework for modeling sequential decision-making problems in RL. In an MDP, an agent interacts with an environment in discrete time steps. At each time step, the agent observes the current state of the environment and interacts with it by choosing an action from a set of possible actions. The environment is assumed to have the Markov property, i.e., the future state depends only on the current state and the action taken, not on the sequence of events that preceded it. The action taken then leads to a new

state, and the agent receives a reward. The goal of RL is to learn a policy that maximizes the cumulative reward that the agent receives over time.

Formally, a fully observable single-agent sequential decision-making problem is typically formulated using MDPs consisting of a tuple $\langle \mathcal{S}, \mathcal{U}, \mathcal{T}, \mathcal{R}, \gamma \rangle$. $s \in \mathcal{S}$ describes the true state of the environment. At each time step, the agent takes action $u \in \mathcal{U}$ using policy $\pi(u|s)$, receives reward feedback $r \in \mathcal{R}$, and transitions to state $s' \in \mathcal{S}$ with transition probability $\mathcal{T}(s'|s, u)$. When acting in the environment, this process generates a trajectory $\tau : S_0, U_0, R_1, S_1, U_1, \dots$ (see the graphical model in Figure 2.1). In this work, we focus on *episodic* environments with discrete action spaces where the trajectory terminates when the agent enters a terminal state, and the environment is typically reset. However, for the purposes of upcoming definitions, we assume that the terminal state is *absorbing*, i.e., transitions to itself with a reward of zero. We also limit the scope to a model-free environment. The agent does not have access to both the transition probabilities ($\mathcal{T}$) and the reward function ($\mathcal{R}$) directly. The policy π has an action-value function:

$$Q^\pi(s, u) = \mathbb{E}_{\tau \sim \pi} [G_t | s_t = s, u_t = u],$$

where $G_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$ is the discounted return from time-step t , and $\gamma \in [0, 1]$ is a discount factor. An optimal policy π^* for the MDP maximizes the Q -value function, $Q^{\pi^*}(s, u) = \max_{\pi} Q^\pi(s, u)$. The optimal Q -value function can be written recursively using the Bellman equation:

$$Q^{\pi^*}(s, u) = \mathbb{E}_{\tau \sim \pi^*} [R_{t+1} + \gamma \max_{u'} Q^{\pi^*}(S_{t+1}, u') | S_t = s, U_t = u].$$

Partial Observability

In contrast to the full observability discussed above, an agent in the real world rarely sees the global Markov state of the system and instead receives local observations from the environment. A partially observable Markov decision process (POMDP) is an extension of MDP where the agent no longer sees the current global state s and instead receives an observation $o \in \Omega$ at each time step based on a probability

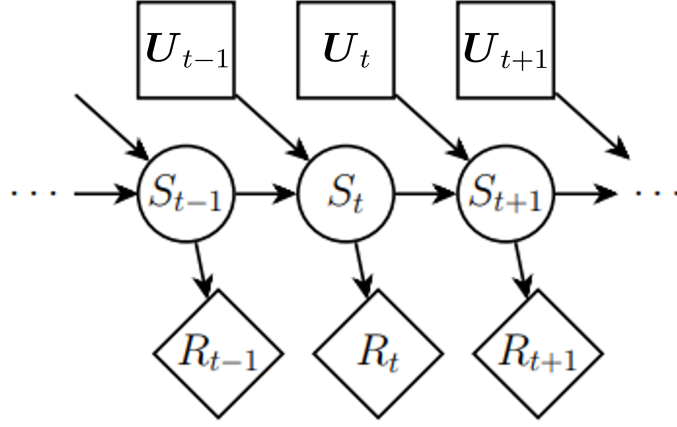


Figure 2.1: Graphical model representation of MDP.

distribution $O : \mathcal{S} \times \Omega \rightarrow \mathbb{R}$, where Ω is a finite set of possible observations. Such an agent typically learns a policy $\pi(u_t|\tau_t)$, where τ_t is the entire action-observation history: $\tau_t \in \mathcal{T}_t \equiv (\Omega \times \mathcal{U})^t \times \Omega$. It is also possible to define a *reactive* POMDP policy [Amato, 2024] $\pi(u_t|o_t)$ depending only on a single observation $o_t \in \Omega$ rather than the whole observation history.

Off-Policy vs On-Policy RL

Off-policy RL algorithms can learn the value of the optimal policy independently of the agent’s current policy. This implies that off-policy algorithms can use data collected from any source, including previous versions of the policy or even other agents. A famous example of an off-policy algorithm is Q-learning [Watkins and Dayan, 1992]. On the contrary, on-policy algorithms learn the value of the policy that the agent is actively using to make decisions. The learning process in on-policy algorithms is directly tied to the actions taken by the current policy, making them inherently dependent on the experiences generated by these actions, as seen in algorithms like SARSA (State-Action-Reward-State-Action) [Rummery and Niranjan, 1994, Sutton and Barto, 2018]. This distinction leads to different

capabilities in terms of data use and efficiency; off-policy algorithms are more flexible and can efficiently use a broader range of experiences, improving sample efficiency.

2.1.1 Q-Learning

Q-Learning [Watkins and Dayan, 1992] is a popular off-policy algorithm which learns optimal Q^{π^*} in a model-free manner based on the Bellman optimality equation above. The update rule for Q-Learning is as follows:

$$Q(S_t, U_t) \leftarrow Q(S_t, U_t) + \alpha \left[R_{t+1} + \gamma \max_{u' \in U} Q(S_{t+1}, u') - Q(S_t, U_t) \right].$$

The Q-learning method above stores the Q values for each state s and each action u in a tabular way. However, computing such a Q value function for problems with high-dimensional state spaces is infeasible. We can, therefore, learn parameterized Q value functions using universal function approximators.

Deep Q-Learning

Deep Q-learning is a state-of-the-art approach using a Deep Q-Network (DQN) [Mnih et al., 2015], parameterized by θ , to represent the action-value function $Q(s_t, u_t; \theta)$. The network parameters θ are learned end-to-end by gradient descent on the mean squared TD-error [Sutton and Barto, 2018]:

$$\mathcal{L}(\theta) = \mathbb{E} \left[\frac{1}{N} \sum_{t=0}^{N-1} \left(r_t + \gamma \max_{u'} Q(s_{t+1}, u'; \theta^-) - Q(s_t, u_t; \theta) \right)^2 \right], \quad (2.1)$$

where the expectation is approximated with mini-batches of episodes $\{s_t, u_t, r_t\}_{t=0}^N$ of varying length N , which are drawn *iid* from an experience replay buffer holding all recently observed episodes [Lin, 1992]. To stabilize optimization, the target values are computed with a regularly updated older copy of the network parameters θ^- [Mnih et al., 2015]. Double Q-learning [van Hasselt et al., 2016] further stabilizes DQN by choosing the action $u' = \arg \max_u Q(s_{t+1}, u; \theta)$ in the above loss based on the current parameters θ . In partially observable settings, Hausknecht and Stone [2015] propose Deep Recurrent Q-Networks (DRQN), which summarizes

the individual action-observation histories of agents by recurrent neural networks. For training on longer time scales, gated architectures such as long-short term memory (LSTM) [Hochreiter and Schmidhuber, 1997] or gated recurrent units (GRU) [Chung et al., 2014] are usually employed.

2.1.2 Policy Gradient

Instead of learning a value function, we can also learn a parameterized policy using gradient descent. A simplest policy gradient method is REINFORCE [Sutton et al., 1999a], which computes the gradient of policy π parameterized with θ as:

$$\begin{aligned}\nabla\eta(\theta) &= \mathbb{E}_{\pi} [G_t \nabla_{\theta} \log \pi(U_t|S_t; \theta)] \\ \theta &\leftarrow \theta + \alpha \nabla\eta(\theta),\end{aligned}$$

where α is the learning rate. As a stochastic gradient method, REINFORCE has good theoretical convergence properties, however it may suffer from large variance in gradient updates which scales unfavorably with the time horizon, since the effect of an action is confounded with the effects of past and future actions. REINFORCE with baseline tries to tackle the issue of variance in gradient updates by subtracting an action-independent baseline [Weaver and Tao, 2013] giving the update:

$$\nabla\eta(\theta) = \mathbb{E}_{\pi} [(G_t - b(S_t)) \nabla_{\theta} \log \pi(U_t|S_t; \theta)],$$

where $V^{\pi}(s)$ is a natural candidate for the baseline.

Actor-Critic

Actor critic methods [Sutton et al., 1999a, Konda and Tsitsiklis, 2000] introduce bias and make use of bootstrapping to estimate the value function in order to avoid high variance Monte Carlo updates from an unbiased estimate of V^{π} . The policy π is referred to as an actor, which is trained by an estimate of G_t computed by the critic. We typically use the advantage function:

$$A(S_t, U_t) = Q^{\pi}(S_t, U_t) - b(S_t)$$

as the critic estimation function obtaining an estimator with lower variance at the cost of introducing bias where the baseline $b(S_t)$ is chosen to be V^π . Other methods like Generalized Advantage Estimation [GAE, Schulman et al., 2015] can also be used. Asynchronous Advantage Actor-Critic method [A3C, Mnih et al., 2016] runs several instances of an advantage-based actor-critic model and updates the weights of a master node after asynchronously accumulating the gradients. This line of work culminated in the IMPALA architecture [Espeholt et al., 2018], which allows the distributed training of actor-critic agents in significantly less time than the original DQN.

Exploration

The most common strategy for exploration is to utilize ϵ -greedy exploration or Boltzmann exploration. ϵ -greedy simply selects the greedy action i.e. $\arg \max_{u \in \mathcal{U}} Q(s, u; \theta)$ with probability $1 - \epsilon$, and a random action with probability ϵ . Boltzmann computes the probabilities of selecting each action by the softmax over the Q-value estimates. For an explicit policy-based agent, we can introduce an entropy regularisation term into the objective, which encourages the policy to be closer to uniform action selection [Mnih et al., 2016].

2.2 Cooperative Multi-Agent RL

A fully cooperative multi-agent task can be formalized as a decentralized partially observable Markov decision process [Dec-POMDP, Oliehoek et al., 2016] consisting of a tuple $G = \langle \mathcal{S}, \mathcal{U}, \mathcal{P}, \mathcal{R}, \Omega, O, n, \gamma \rangle$. $s \in \mathcal{S}$ describes the true state of the environment. At each time step, each agent $a \in \mathcal{A} \equiv \{1, \dots, n\}$ chooses an action $u^a \in \mathcal{U}$, forming a joint action $\mathbf{u} \in \mathcal{U} \equiv \mathcal{U}^n$. This causes a transition in the environment according to the state transition kernel $\mathcal{P}(s'|s, \mathbf{u}) : \mathcal{S} \times \mathcal{U} \times \mathcal{S} \rightarrow [0, 1]$. All agents are collaborative and share, therefore, the same reward function $\mathcal{R}(s, \mathbf{u}) : \mathcal{S} \times \mathcal{U} \rightarrow \mathbb{R}$. $\gamma \in [0, 1)$ is a discount factor.

Due to *partial observability*, each agent a cannot observe the true state s , but receives an observation $o^a \in \Omega$ drawn from observation kernel $o^a \sim O(s, a)$. At time t , each agent a has access to its action-observation history $\tau_t^a \in \mathcal{T}_t \equiv (\Omega \times \mathcal{U})^t \times \Omega$, on which it conditions a stochastic policy $\pi^a(u_t^a | \tau_t^a)$. $\boldsymbol{\tau}_t \in \mathcal{T}_t^n$ denotes the histories of all agents. The joint stochastic policy of all agents $\pi(\mathbf{u}_t | \boldsymbol{\tau}_t) \equiv \prod_{a=1}^n \pi^a(u_t^a | \tau_t^a)$ induces a joint action-value function:

$$Q^\pi(s_t, \mathbf{u}_t) = \mathbb{E}_{s_{t+1:\infty}, \mathbf{u}_{t+1:\infty}} [G_t | s_t, \mathbf{u}_t],$$

where $G_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$ is the *discounted return*.

CTDE

An important paradigm of MARL is Centralized Training and Decentralized Execution [CTDE, Kraemer and Banerjee, 2016], which allows access to all action-observation histories $\boldsymbol{\tau}_t$ and global state s_t during training, but each agent's decentralized policy π^a can only condition on its own action-observation history τ_t^a . This approach can exploit information that is not available during execution and also freely shares parameters and gradients, which improves the sample efficiency [see, e.g., Foerster et al., 2018, Rashid et al., 2020b, Böhmer et al., 2020].

Part I : Semantic Exploration in RL

Abstract

Effectively exploring the environment is a crucial challenge in Reinforcement Learning (RL), especially when feedback provided by the environment is very sparse. For example, defining a dense reward signal can be quite complex in practical scenarios like autonomous vehicles. Similarly, crafting a precise reward structure that guides an agent in making intricate surgical maneuvers without human intervention is equally challenging in robotic surgery. As such, it is often only feasible to get feedback in the form of a binary success or failure at the conclusion of a task. Due to the lack of any intermediate reward from the environment, it becomes pretty challenging for the agent to explore their environment. One popular way is to incentivize the agent to explore novel states by providing additional intermediate rewards to the agent. This is the concept of intrinsic motivation. An important question is, then, how should we define the novelty of states. Additionally, procedurally generated environments present a significant learning challenge as an agent is unlikely to visit a state more than once. Therefore, the learning method needs to generalize to unseen scenarios.

In this thesis, we investigate whether semantically meaningful state abstractions informed by natural language can be beneficial for speeding up exploration. We propose a novel intrinsic reward based on a foundation model, such as contrastive language image pretraining (CLIP), which can encode a wealth of domain-independent semantic visual-language knowledge about the world. Specifically, our intrinsic reward is defined based on pre-trained CLIP embeddings without any fine-tuning or learning on the target RL task. We demonstrate that CLIP-based intrinsic rewards can drive exploration toward semantically meaningful states and outperform state-of-the-art methods in a challenging, sparse-reward, procedurally generated environment.

3

Language Guided Exploration

Contents

3.1	Introduction	22
3.2	Background	25
3.2.1	Exploration in Deep RL	25
3.2.2	Foundation Models in RL	30
3.2.3	IMPALA	31
3.2.4	CLIP	32
3.2.5	Problem Formulation	33
3.3	FoMoRL	33
3.3.1	Semantic Intrinsic Motivation	33
3.3.2	Training Setup	35
3.4	Experiments	36
3.4.1	MiniGrid Domain	36
3.4.2	Baseline	37
3.4.3	Results	38
3.4.4	CLIP Analysis on MiniGrid	41
3.5	Further Discussions and Limitations	44
3.5.1	Promoting Novelty in Pixel Space vs CLIP Representations	44
3.5.2	Discussion on the choice of foundation model	45
3.5.3	Limitations	46
3.6	Conclusion and Future Work	47

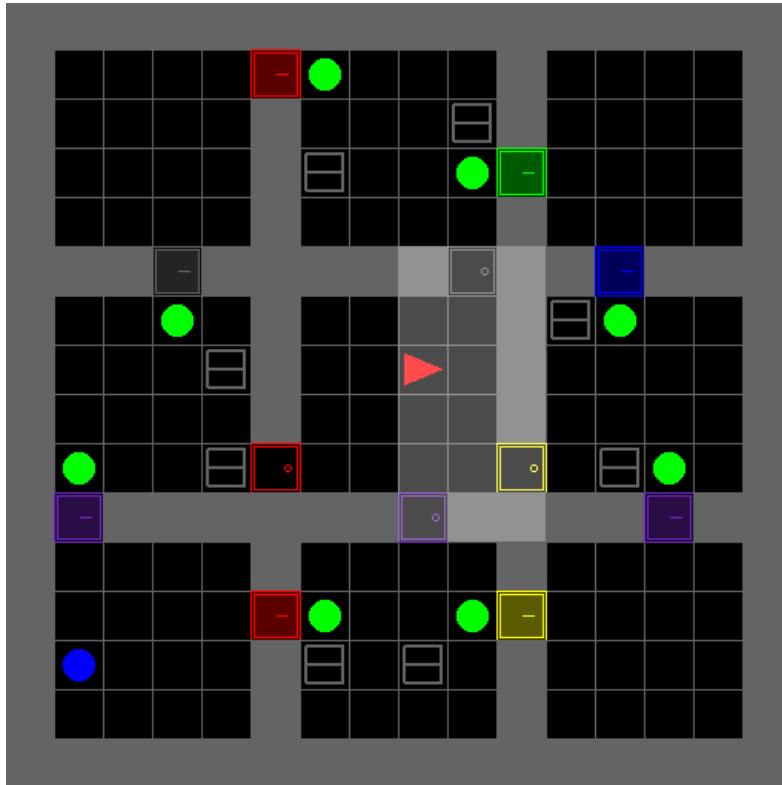


Figure 3.1: Example of a fully observable view of an agent (marked with red triangle) in an obstructed maze from MiniGrid Environment [Chevalier-Boisvert et al., 2018] learning to navigate to the goal location (marked with the blue ball). The shaded part shows the partially observable image that the agent can see. Note that the agent cannot see past locked doors.

3.1 Introduction

Exploration is a key challenge in reinforcement learning (RL), especially when extrinsic feedback is sparsely provided by the environment. Consider an agent (marked with a red triangle) in Figure 3.1 learning to navigate to a *goal* location (i.e., the blue ball) in an obstructed maze environment consisting of different types of objects such as locked doors, keys hidden in boxes and doors obstructed by balls. The agent only receives success or failure feedback from the environment at the end of a fixed time limit. The agent needs to learn to successfully navigate to the desired goal. Due to the lack of any intermediate reward from the environment, it becomes quite challenging for the agent to explore their environment and reach the goal within a fixed time limit. Intrinsic motivation provides an effective way

for agents to explore the environment by rewarding the visitation of *novel* states [Oudeyer and Kaplan, 2009, Aubret et al., 2019]. An important question is then the definition of state novelty. Quoting from Tam et al. [2022], “acquiring state representations that support effective exploration is a chicken-and-egg problem — knowing whether two states should be considered similar requires the type of understanding that an agent can only acquire after effectively exploring its environment.”. Additionally, when the environment is partially observable, the agent cannot observe the full state and needs to reach the goal solely based on *partially observable* pixel images that its camera can see. In such cases, we may reward novel observations or novel histories of observations.

In this chapter, we investigate whether semantically meaningful state abstractions informed by natural language can be beneficial for speeding up exploration. Language abstractions can provide an intuitive and compact representation of the environment’s state. This is especially useful in complex environments where the state space is large or difficult to navigate. The RL agent can more efficiently process and understand its surroundings by abstracting details into linguistic concepts. Additionally, language as a tool for abstraction allows knowledge to be more easily transferred across different tasks or domains. For example, an RL agent trained in one environment can leverage language-based descriptions to apply learned strategies to new but related environments. This makes the learning process more flexible and generalizable. In our obstructed maze scenario, the novelty of visiting states where the agent finds keys to doors and opens doors should be higher than moving to different cells within the same room, as it facilitates faster exploration and might lead the agent to reach the goal location within the time limit. A language-guided state representation can concisely capture such semantics and can, therefore, enable more efficient language-guided exploration.

Several existing methods compute state novelty scores based on some distance function in a learned state representation, which is typically learned simultaneously with the policy [Bellemare et al., 2016, Pathak et al., 2017]. A famous recent

work, RIDE [Raileanu and Rocktäschel, 2020], introduces impact-driven intrinsic reward based on differences in state representations of consecutive states. The state representation is learned from learning a transition function via the forward and inverse dynamics model. However, learning a transition function can take a long time and could be very difficult to learn for partially-observable high-dimensional RGB images. Another work on random network distillation (RND) [Burda et al., 2018] uses a fixed randomly initialized neural network to represent the states and reward the exploration based on the error of a neural network predicting the fixed features. This chapter proposes a *foundation model* based intrinsic motivation scheme for RL, FoMoRL. Instead of computing novelty scores based on domain-specific learned state representations, FoMoRL uses the pre-trained representation of a foundation model, i.e., a generic model trained on extensive domain-independent visual-language data, such as contrastive language image pretraining (CLIP) [Radford et al., 2021]. In this manner, FoMoRL may incorporate semantic knowledge about the state space without requiring any domain-specific pretraining and thus help guide exploration towards semantically meaningful states such as picking up keys, opening the doors, and interacting with other useful objects relevant to the task.

The idea of FoMoRL is general to the choice of RL algorithm, motivation scheme, and the choice of foundation model. We apply FoMoRL on top of RIDE [Raileanu and Rocktäschel, 2020] and the IMPALA distributed off-policy RL algorithm [Espeholt et al., 2018] and use CLIP as our foundation model [Radford et al., 2021]. Formally, FoMoRL can be applied to any visual RL tasks, with or without language conditioning; however, FoMoRL is expected to be effective in domains where human-interpretable semantic knowledge is helpful to succeed. FoMoRL would be particularly useful in scenarios where the environment is complex, large, or procedurally generated, and where intrinsic motivation is necessary to drive effective exploration due to sparse extrinsic rewards. In such settings, human-interpretable semantic knowledge might be crucial for navigating and understanding the environment. Examples include tasks requiring high-level abstractions and

semantic understanding, such as robotic navigation in dynamic environments, autonomous vehicles, and intricate problem-solving tasks where context and object interactions matter. However, FoMoRL might not be as effective in environments where state changes are rapid and subtle, requiring highly specific and detailed knowledge that falls outside the scope of CLIP’s generalized training data. Please refer to Section 3.5 for a broader discussion on limitations of our approach. We evaluate FoMoRL on the challenging sparse-reward procedurally generated MiniGrid domain [Chevalier-Boisvert et al., 2018]. Procedurally generated environments present a significant learning challenge as an agent is unlikely to visit a state more than once. Therefore, the learning method needs to generalize to unseen scenarios.

Somewhat surprisingly, our results suggest that CLIP representations can be very effective even in grid-world environments, despite the fact that CLIP was trained on general human language-vision data that contains little (if any) grid-world-like data points. Our analysis shows that CLIP representations encode fine-grained discrimination between different objects, colors, counts, and other visual attributes, which makes them practical for novelty-based exploration even in grid worlds. FoMoRL significantly outperforms the state-of-the-art RIDE [Raileanu and Rocktäschel, 2020] method in 6 out of 9 challenging MiniGrid exploration tasks and performs similarly in the rest of the tasks. FoMoRL learns to solve the tasks 20% faster and achieves over 86% higher average return per episode across all tasks.

3.2 Background

We now provide a brief survey of exploration and foundation models in Deep RL. For a more detailed reference, please refer to Amin et al. [2021] and Zhou et al. [2023].

3.2.1 Exploration in Deep RL

There are many techniques for efficient exploration in model-free single-agent RL, including but not limited to intrinsic novelty reward [Bellemare et al., 2016, Tang et al., 2017], predictability [Pathak et al., 2017, Oudeyer et al., 2007, Stadie et al.,

2015], pure curiosity [Burda et al., 2019], Bayesian posteriors [Osband et al., 2016b, Gal et al., 2017, Fortunato et al., 2018, O’Donoghue et al., 2018], information gain [Houthoofd et al., 2016] or empowerment [Klyubin et al., 2005, Choshen et al., 2018]. In this thesis, we focus on intrinsic motivation, which augments the environment reward with an additional reward to encourage exploration.

Count Based Intrinsic Exploration

Count-based intrinsic exploration methods often focus on the discovery of *novel* states by keeping track of how often a state is visited through *state visitation counts* and give a reward based on this. Bellemare et al. [2016] propose *pseudo-counts* to make use of count-based methods for intrinsic motivation in large state spaces. They propose a method for obtaining a pseudo-count with the use of a density model over the state space. Following up, Van den Oord et al. [2016] propose PixelCNN as a more expressive density model estimating the joint probability distribution of pixels in an image. PixelCNN is autoregressive, i.e., each pixel is predicted based on the pixels that have already been generated.

Tang et al. [2017] introduced an alternative method for approximating counts across a vast state space, employing locally sensitive hashing alongside an approximate counting mechanism. Fu et al. [2017] capitalize on the discriminative power of neural networks, developing models that map out densities across the state space. Given a dataset of n points, they propose to train an *exemplar* model consisting of a set of n classifiers. Each classifier is trained to discriminate a single exemplar point from the rest of the points in the dataset. The key idea driving this methodology is that a state that stands out easily from the rest indicates novelty, in contrast to a state that appears more commonly.

Prediction Based Intrinsic Exploration

Prediction-based intrinsic exploration methods aim to reward an agent for improving its knowledge about the environment by modeling the environment dynamics through

a prediction model. Stadie et al. [2015] aims to learn the environment’s transition function in a compressed feature space created via auto-encoder training, using surprise as the basis for intrinsic motivation. Houthoofd et al. [2016] propose intrinsic bonuses with Bayesian Surprise, using a Bayesian Neural Network to model environmental transitions. The intrinsic reward is derived from KL divergence between the transition model before and after observing an experience. Their approach notably enhances performance in low-dimensional continuous control tasks, particularly in environments with sparse rewards.

Pathak et al. [2017] propose to calculate surprise in a feature space tailored to the agent, learning an inverse dynamics model to predict actions based on the current and subsequent state. This approach focuses the agent on state space features critical for action prediction, ignoring irrelevant elements. Haber et al. [2018] also develop an inverse dynamics model (termed the world model) alongside a self-model, which predicts the world model’s errors. The self-model chooses actions to maximize the world model’s error, thereby steering the agent towards unexpected situations.

Sukhbaatar et al. [2017] introduce an intrinsic motivation approach using self-play, dividing the agent into two parts: Alice and Bob. Alice creates a task by performing it, and Bob aims to complete it quickly, rewarded with the negative of the time taken to complete the task (up to $t_{\max}$). Alice’s reward is $\max(0, t_{\text{Bob}} - t_{\text{Alice}})$, motivating her to set challenging yet efficient tasks. This dynamic leads to a self-generated curriculum by Alice, progressively urging Bob to explore further. Their findings demonstrate that this unsupervised method enhances exploration and accelerates Bob’s learning in new tasks within the same environment.

Raileanu and Rocktäschel [2020] proposed RIDE, which computes novelty scores based on differences between state representations of consecutive states. The state representations are learned from learning a transition function via the forward and inverse dynamics model. Another work on random network distillation (RND) [Burda et al., 2018] uses a fixed randomly initialized neural network to represent the states and reward the exploration based on the error of a neural network

predicting the fixed features. Never Give Up (NGU; [Badia et al., 2020]) combines an episodic novelty module based on inverse dynamics features with RND as a lifelong novelty module. In contrast to these works, we propose using pre-trained foundation models to define intrinsic rewards.

Concurrent with our efforts, Tam et al. [2022] proposed the idea of using pre-trained representations from foundation models for exploration in RL. The high-level idea in the concurrent work is the *same* as ours; however, there are important differences in its application. The concurrent work builds on top of RND [Burda et al., 2018] and NGU [Badia et al., 2020] whereas we use RIDE [Raileanu and Rocktäschel, 2020]; and the concurrent work mostly focuses on Unity-based 3D environments with language oracles, whereas we experiment with procedurally-generated long-horizon grid-world tasks that have no language oracle, and whose visual appearance differs significantly from the visual data used to pretrain the CLIP model. This further affirms that the idea of FoMoRL is general to the choice of RL algorithm, motivation scheme, and the choice of foundation model.

Bayesian Approaches

Bayesian approaches in RL involves learning distributions over key quantities. For example, Strens [2000] establishes the framework for Posterior Sampling based approaches, which maintain distributions over Markov Decision Processes (MDPs), particularly the state-transition and reward functions, independently for each state-action pair. These algorithms sample from these distributions, compute an optimal policy, and act greedily based on it. Abbasi-Yadkori and Szepesvári [2015] extended Strens [2000] posterior sampling approach to MDPs with parameterized transition functions. They introduced lazy posterior sampling, updating the posterior over parameters at fixed intervals. Lipton et al. [2018] applied Bayes-by-Backprop to maintain a distribution over Q-values and used Thompson Sampling to sample from this distribution at each timestep, acting greedily based on the Q-value estimates. They demonstrated significant advantages over ϵ -greedy, Boltzmann exploration,

and Bootstrapped-DQN in a dialogue learning task. Tang and Kucukelbir [2017] also used variational inference for a posterior over Q-values, improving exploration in small environments with Thompson Sampling. Gal and Ghahramani [2016] proposed using Dropout to maintain an approximate distribution over Q-values, demonstrating its effectiveness with Thompson Sampling in a simple 2D domain. Henderson et al. [2017] found that applying Dropout to value functions leads to more stable training in an actor-critic framework.

Osband et al. [2019] argue that maintaining a distribution over value functions is impractical. Instead, they approximate this by injecting Gaussian noise into the value function parameters during training with Least Squares Value Iteration, showing this is similar to posterior sampling under certain assumptions. Bootstrapped DQN [Osband et al., 2016a] extends randomized value functions to neural networks by maintaining multiple Q-value heads that produce estimates from a shared representation. Using non-parametric bootstrap and varied initialization, they approximate a distribution over Q-values. Osband et al. [2018] enhanced Bootstrapped DQN by incorporating a prior to improve uncertainty estimates. They used the parameters of a randomly initialized network as the prior, encoding no prior task knowledge.

Other Approaches for Sample Efficient Exploration

AlphaGo Zero [Silver et al., 2016] uses Monte Carlo Tree Search to enhance policy by exploring diverse actions beyond its greedy policy, avoiding local optima. Kulkarni et al. [2016a] apply hierarchical reinforcement learning to provide sub-goals, enabling the agent to learn a Q-function incorporating state, action, and sub-goals. The Q-functions over states, actions and goals are trained purely on intrinsic rewards for reaching the sub-goals, whereas the Q-function over sub-goals is trained purely on extrinsic rewards, providing expert domain knowledge to an agent. Andrychowicz et al. [2017] introduce Hindsight Experience Replay, utilizing Universal Value Functions to learn Q-values for multiple goals. They reuse experiences by changing

the goal when sampling from the replay buffer, a method foundational for many hierarchical approaches, as environmental transitions are goal-independent.

Gregor et al. [2016] propose learning a set of Options by maximizing mutual information between Options and their termination states, ensuring distinct termination states. This method allows pre-training an agent without rewards, leading to faster exploration when a goal is introduced. Eysenbach et al. [2018] argue for training maximum entropy policies to learn more varied skills, as these policies encourage greater diversity in exploring the state space by ensuring different parts of the state space are reached. Gupta et al. [2018a] explore Meta-Reinforcement Learning for improved exploration in a multi-task setting. They train a single policy for all tasks, using injected Gaussian noise to encourage exploration. The goal is to develop a policy that can first explore and then adapt to new tasks using knowledge from previous tasks.

3.2.2 Foundation Models in RL

Mu et al. [2022] uses language to improve exploration via intrinsic rewards instead of using raw states. However, their method requires an oracle language annotator, which is not easily available for many RL environments. Khandelwal et al. [2022] investigate the effectiveness of CLIP visual representations directly for control on Embodied AI tasks [Batra et al., 2020] by bypassing the learning of policy representations with CLIP embeddings. Their results demonstrated the effectiveness of CLIP representations for control on navigation-heavy Embodied AI tasks. Shridhar et al. [2022] proposed CLIPort, which uses CLIP representations with imitation learning for robotic manipulation tasks, adding a spatial understanding of the scenes as well. SayCan [Ahn et al., 2022] uses a Large Language Model (LLM) to supply high-level instructions to solve the tasks and pair it with an affordance-based value function to see if the robot has the required skills to perform those instructions. Similarly, Patel and Pavlick [2021] shows that LLMs such as GPT-2 [Radford et al., 2019] or GPT-3 [Brown et al., 2020] can learn to ground the concepts such as direction or

color that it is explicitly taught and also generalize to several instances of unseen concepts as well. In contrast to the above works, we explore using intrinsic rewards based on foundational models for efficient semantic exploration in visual RL tasks.

3.2.3 IMPALA

Importance Weighted Actor-Learner Architectures [IMPALA, Espeholt et al., 2018] is a distributed reinforcement learning (RL) agent. It is designed to handle large-scale, multi-task learning using a single set of parameters. In contrast to the A3C-based [Mnih et al., 2016] agents where workers share gradients related to policy parameters with a central server, IMPALA actors transmit full experiences, including state, action, and reward sequences, to a centralized learner, which then parallelize time-independent operations to achieve higher throughput. Since the policy used to generate experiences differs from learner policy while computing gradients, the authors propose a *V-trace* off-policy actor-critic algorithm. Given a behaviour policy μ used by the actor to generate trajectories, current learner policy π_θ parameterized by θ , and value function V_ϕ parameterized by ϕ . The value function parameters for state at time t , i.e. $V(s_t; \phi)$ are updated to minimize the L_2 loss to the target v_t defined as:

$$v_t = V(s_t; \phi) + \sum_{t'=t}^{t+n-1} \gamma^{t'-t} \left(\prod_{i=t}^{t'-1} c_i \right) \delta_{t'} V,$$

where $\delta_{t'} V = \rho_{t'} (r_{t'} + \gamma V(s_{t'+1}; \phi) - V(s_{t'}; \phi))$ is TD error at time step t' . $\rho_{t'}$ and c_i are truncated importance sampling weights. $\rho_{t'}$ is defined as $\min \left(\tilde{\rho}, \frac{\pi(u_{t'}|s_{t'})}{\mu(u_{t'}|s_{t'})} \right)$ and it affects which policy's value function we converge to. c_i is defined as $\min \left(\tilde{c}, \frac{\pi(u_i|s_i)}{\mu(u_i|s_i)} \right)$ and it impacts the speed of convergence. $\tilde{\rho}$ and $\tilde{c}$ are corresponding truncation levels and $\tilde{\rho} \geq \tilde{c}$. The policy parameters θ are updated in the direction of policy gradient given by:

$$\rho_t \nabla_\theta \log \pi_\theta(u_t|s_t) (r_t + \gamma v_{t+1} - V(s_t; \phi)).$$

For a more detailed introduction, please refer to Espeholt et al. [2018].

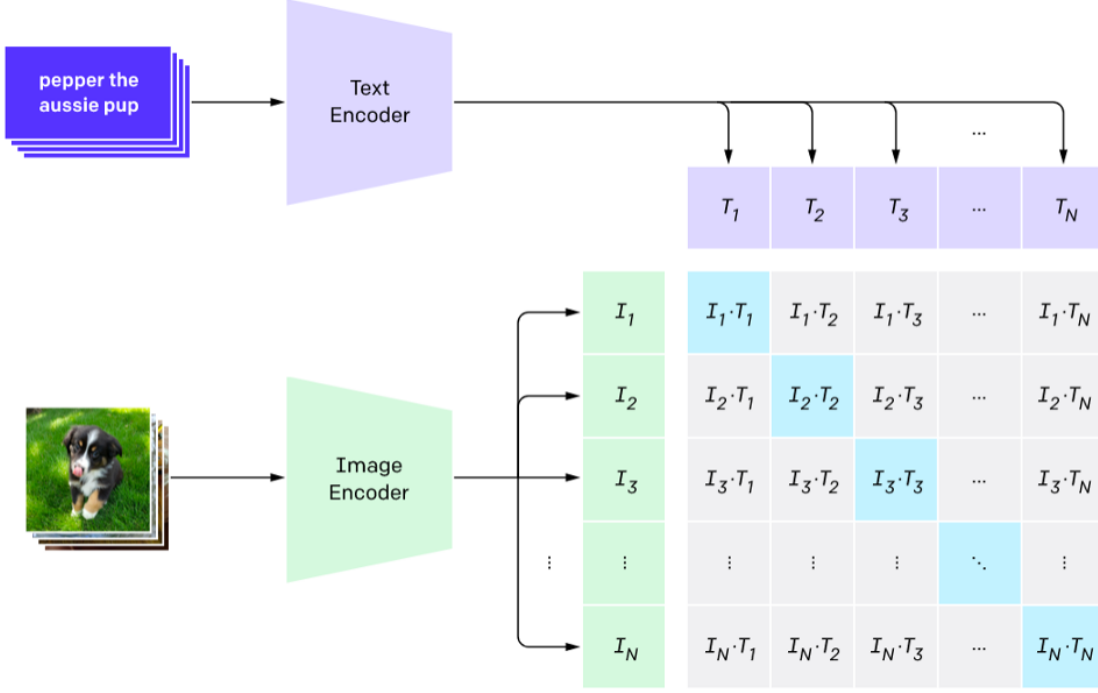


Figure 3.2: Training Architecture for CLIP’s visual and language encoder with a contrastive loss directly taken from Radford et al. [2021]. The visual representations of Image I_1 are trained to be *close* to its corresponding caption representations T_1 but *far* from other caption representations $T_2, \dots, T_N$ and vice-versa.

3.2.4 CLIP

Contrastive Language–Image Pretraining (CLIP) [Radford et al., 2021] is a large foundation model trained on a big human dataset containing images paired with descriptive texts. It uses a contrastive loss to learn representations such that images match their corresponding text descriptions and vice-versa, effectively understanding content across visual and textual domains. Central to CLIP is acquiring perceptual understanding through guidance embedded in natural language. This results in visual representations grounded in language abstractions, enabling fine-grained zero-shot classification. CLIP has shown remarkable performance in various benchmarks; for example, a zero-shot CLIP classifier *outperforms* a fully supervised linear classifier fitted on ResNet-50 features on 16 datasets on the ImageNet dataset. Note that the supervised baseline was trained on multiple datasets *including* ImageNet. Figure

3.2 shows an illustration of training architecture for CLIP’s visual and language encoder with a contrastive loss.

3.2.5 Problem Formulation

A partially observable Markov decision process (POMDP) is defined as a tuple $G = \langle \mathcal{S}, \mathcal{U}, \mathcal{P}, \mathcal{R}, \Omega, O, \gamma \rangle$. The true state of the environment is denoted by $s \in \mathcal{S}$. At each time step, an agent chooses an action $u \in \mathcal{U}$, which causes a transition in the environment according to the state transition kernel $\mathcal{P}(s'|s, u) : \mathcal{S} \times \mathcal{U} \times \mathcal{S} \rightarrow [0, 1]$, and the agent receives an extrinsic reward $\mathcal{R}_{\text{ext}} : \mathcal{S} \times \mathcal{U} \rightarrow \mathbb{R}$. $\gamma \in [0, 1)$ is a discount factor. Due to partial observability, the agent cannot observe the true state s but receives a partially observable visual image $o \in \Omega$ drawn from observation kernel $o \sim O(s, u)$. At time t , the agent has access to its action-observation history $\tau_t \in \mathcal{T}_t \equiv (\Omega \times \mathcal{U})^t \times \Omega$, on which it conditions a stochastic policy $\pi(u_t|\tau_t)$. The stochastic policy induces an action-value function : $Q^\pi(s_t, u_t) = \mathbb{E}_{s_{t+1:\infty}, u_{t+1:\infty}} [G_t|s_t, u_t]$, where $G_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$ is the *discounted return*. In this chapter, we consider a POMDP setting, where the learning method is allowed access to the state information, only during training.

3.3 FoMoRL

FoMoRL assigns an intrinsic reward based on CLIP’s [Radford et al., 2021] visual encoder embeddings. CLIP embeddings enable semantic understanding of the images as they have been contrastively trained with image-captioned large-scale human data, thereby incorporating language abstractions. These language abstractions can summarise the images with important fine-grained details and yet be very concise, much like our natural languages [Borghi and Binkofski, 2014].

3.3.1 Semantic Intrinsic Motivation

We now describe our intrinsic reward module which computes the intrinsic reward to incentivize the agent to explore semantically meaningful observations. We define

the representation of the current observation $\phi(o_t)$ as the pre-trained representation from the CLIP encoder, i.e., $\phi(o_t) = \text{clip}(o_t)$. The intrinsic reward $\mathcal{R}_{\text{int}}$ at time step t is computed as the L_2 -norm $\|\phi(o_{t+1}) - \phi(o_t)\|_2$ of the difference in the clip representation between consecutive time steps discounted by episodic state visitation counts similar to Raileanu and Rocktäschel [2020]:

$$r_{\text{int}}(s_t, u_t, o_t) = \frac{\|\text{clip}(o_{t+1}) - \text{clip}(o_t)\|_2}{\sqrt{N_{ep}(s_{t+1})}},$$

where $N_{ep}(s_{t+1})$ is the visitation count of state s_{t+1} during the current episode, which is initialized to 1 at the beginning of the episode. In our setup, we assume that the intrinsic reward module has access to the global state of the environment to compute the denominator in the above equation. The learnt policy is still conditional *only* on the action-observation history of the agent. As a result, please note that the state information is only required during training for the intrinsic reward module, and will not be required during execution. We present a way in the next section on how to get past this limitation if the state information is not available during training. The above intrinsic reward encourages the visitation of observations that have significantly different CLIP representations from the current observation. CLIP representations can perform fine-grained classification of abstract concepts in the visual observations and can, therefore, guide exploration towards semantically meaningful observations, which are useful for the downstream task. The division by state counts ensures that the agent does not exploit the intrinsic reward function by swinging between two observations with large differences in CLIP embeddings. For this chapter, we use the officially released ResNet50 $\times$ 16 model for the CLIP encoder, which outputs observation representations of dimension 1024.

Discussion on State Counts

The state visitation count can be directly computed for MDPs with a small number of discrete states but can also be approximated for MDPs with large discrete or continuous state spaces [Martin et al., 2017, Machado et al., 2020]. For example, one

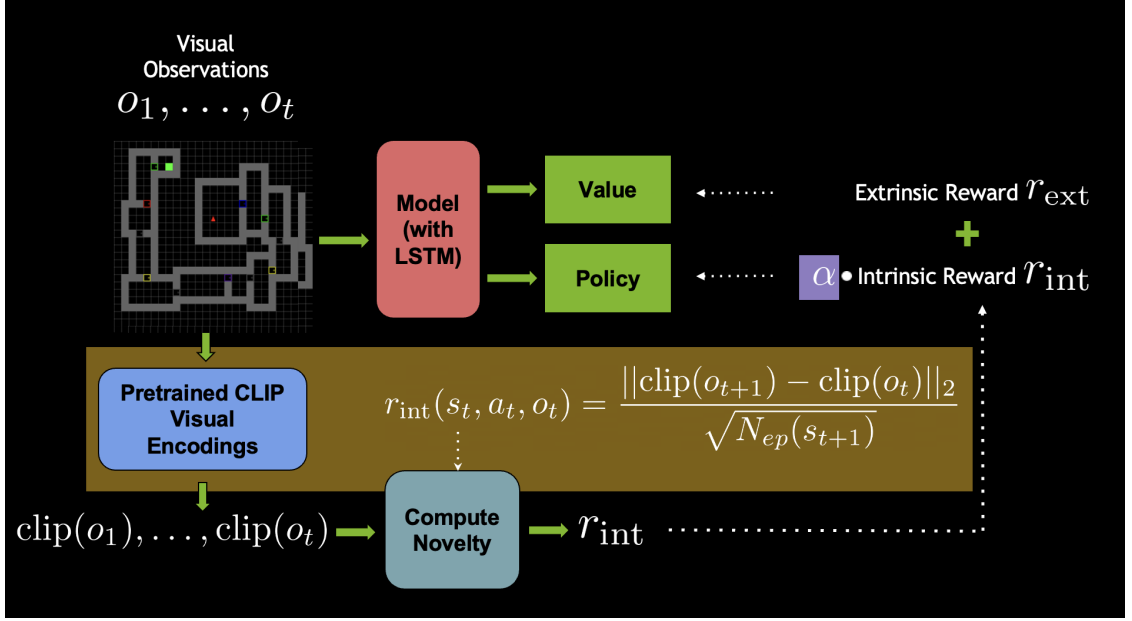


Figure 3.3: An illustration of different components of FoMoRL.

can use Random Network Distillation (RND) [Burda et al., 2018] to approximate state counts. The RND exploration bonus is defined as the error of a neural network $\hat{f}(s_t)$ predicting features of the states given by a fixed randomly initialized neural network $f(s_t)$ i.e. $|\hat{f}(s_t; \theta) - f(s_t)|_2^2$. The more an agent explores s_t , the less the prediction error will be, and therefore, we can use the inverse of RND loss as an approximate state visitation count, i.e.

$$N_{\text{ep}}(s_t) = \frac{1}{|\hat{f}(s_t; \theta) - f(s_t)|_2^2}.$$

In the context of POMDPs, where state information is unavailable during training, we can approximate state as action-observation history (τ_t) of the agent with an LSTM, and then use the RND state count approximations shown above by predicting fixed features of the history: $f(\tau_t)$.

3.3.2 Training Setup

We train our policy using IMPALA [Espeholt et al., 2018] in a distributed off-policy setup with multiple actors. The policy is trained to maximize the combined

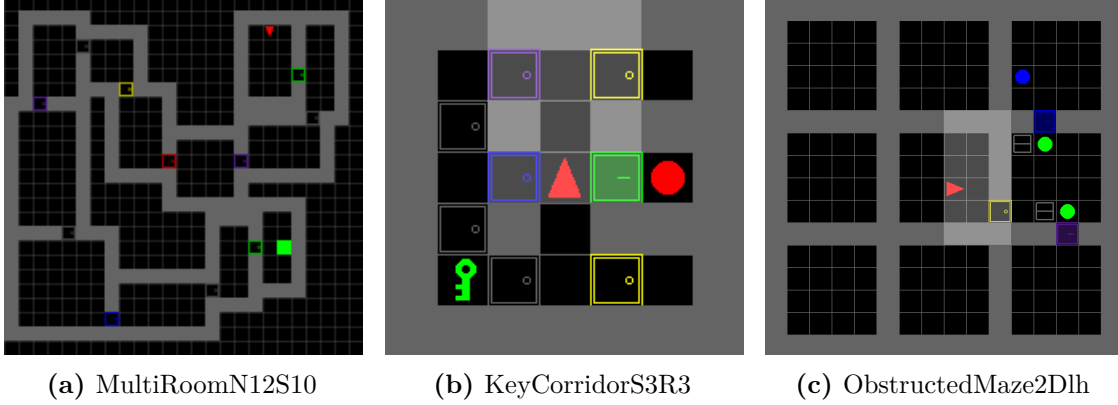


Figure 3.4: Representative examples of different MiniGrid environments. The highlighted area showcases the partial observations the agent receives.

reward (r_{comb}):

$$r_{\text{comb}}(s_t, u_t) = \alpha r_{\text{int}}(s_t, u_t) + r_{\text{ext}}(s_t, u_t),$$

where α is a hyperparameter. The policy learns its own state representations from scratch, while the pre-trained representations are just used for guiding exploration through intrinsic rewards. The total loss L_{tot} function is composed of four components: L_{pg} : policy gradient loss, L_{base} : value function loss, and L_{ent} : entropy loss, with respective hyper parameter coefficients i.e.

$$L_{\text{tot}} = L_{\text{pg}} + c_{\text{base}}L_{\text{base}} + c_{\text{ent}}L_{\text{ent}}$$

Since our policy is conditioned on partial observations, we use an LSTM [Hochreiter and Schmidhuber, 1997] to summarize the action-observation history τ_t . The overall architecture is shown in Figure 3.3.

3.4 Experiments

3.4.1 MiniGrid Domain

We evaluate our proposed method on three types of hard exploration tasks from the MiniGrid domain [Chevalier-Boisvert et al., 2018]: MultiRoomNXSY, KeyCorridorS3R3, and ObstructedMaze2Dlh. The environment consists of an $N \times M$ grid world with each cell either empty or consisting of one of the objects such as

wall, floor, lava, door, key, ball, box, and goal. The agent can pick up and carry exactly one object, and each object has a discrete color. In order to open up a locked door, the agent has to pick up and carry the key matching the color of the door. The action space is discrete and allows the agent to move forward, turn left/right, pick up an object, drop the object being carried, toggle (open doors or interact with objects), and a done action to mark the task complete. At each time step, the agent receives a partial and egocentric observation, and the agent cannot see through walls or closed doors.

The MultiRoomNXSY task (Figure 3.4a) consists of X rooms, with size at most Y , connected in random orientations in every episode. This task has a series of connected rooms with doors that must be opened in order to get to the next room, with the green goal square as the final destination the agent must get to. In the KeyCorridorS3R3 task (Figure 3.4b), the agent has to explore the environment to find a key hidden in another room and then use the key to open the corresponding door and pick up the ball. In the ObstructedMaze2Dlh task (Figure 3.4c), the agent has to pick up a box that is placed in the corner of a maze. The doors are locked, the keys are hidden in boxes, and the doors are obstructed by balls. The NoisyTV [Burda et al., 2018] version of MultiRoomNXSY adds stochasticity to the environment by adding a random ball that changes color every time the agent takes a particular action. The agents need to learn to avoid this distraction as it is irrelevant to the task.

The environment provides different input representations of the state: (1) compact grid encoding using three integer values describing the type and color of the object in the cell and a flag indicating whether doors are open or closed; (2) raw RGB visual observations from the procedurally generated environment.

3.4.2 Baseline

For all our experiments, we use IMPALA [Espeholt et al., 2018] as the base RL algorithm and compare it against the popular RIDE method by Raileanu and

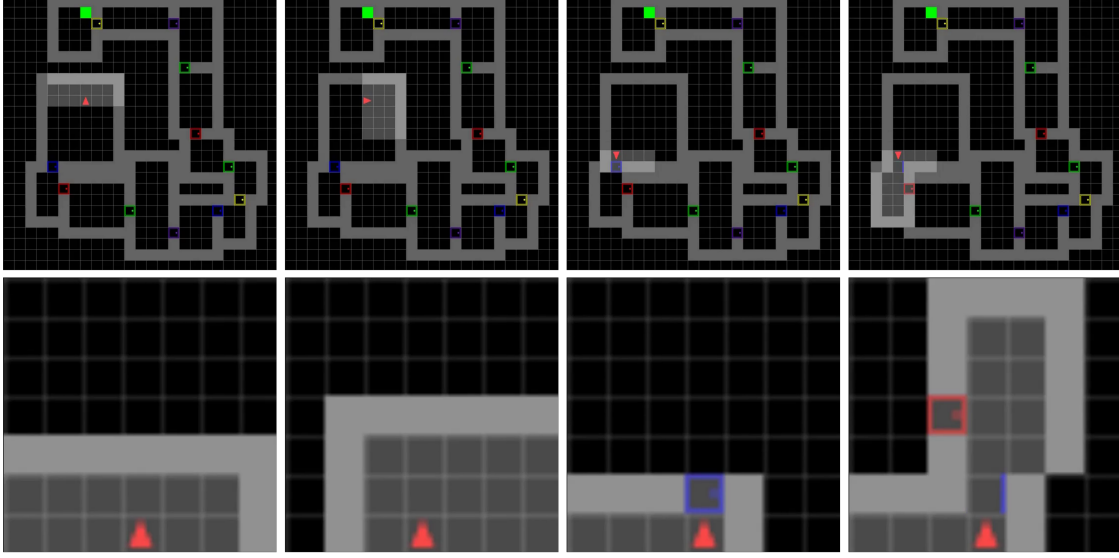


Figure 3.5: A few examples of the fully observable view (top) vs corresponding partially observable view (bottom) available to the agent.

Rocktäschel [2020]. The original RIDE method also uses IMPALA to train the policy with the intrinsic reward based on learning state representations, which minimizes the forward and inverse dynamics loss. In the original paper, the authors use the compact grid encoded representation (see Section 3.4.1 for definition), which already contains all key required information about the visual observation, thereby making it simpler to learn a transition function and the state representations. In this chapter, we instead use the raw partially-observable visual RGB observations, which makes the task significantly harder as agents need to extract the features from raw images. Both the policy and intrinsic reward modules for the RIDE and FoMoRL methods receive a partial egocentric observation as input. Figure 3.5 shows the difference in a fully observable vs partially observable view of the environment for a few frames.

3.4.3 Results

Table 3.1 summarizes the results for various MiniGrid tasks. The values indicate the final mean episodic return averaged over multiple seeds after training on $[xM]$ frames. The bracketed values (xM) indicate the average number of training

Table 3.1: Performance of RIDE and FoMoRL on a variety of hard exploration problems in MiniGrid. Both methods receive partially observable RGB images as input to the policy and intrinsic reward module. The values indicate the final mean episodic return averaged over multiple seeds after training on $[xM]$ frames. The bracketed values (xM) indicate the average number of training frames (in millions) required to achieve the convergence value. The (-) indicates no convergence in $[xM]$ frames.

	RIDE	FoMoRL
DoorKey-5x5 [15M]	0.96 (5M)	0.96 (2.5M)
MultiRoom-N7-S4 [30M]	0.71 (15M)	0.76 (7M)
MultiRoomNoisyTV-N7-S4 [30M]	0.7 (20M)	0.74 (19M)
MultiRoom-N7-S8 [30M]	0.0 (-)	0.59 (29M)
MultiRoom-N10-S4 [30M]	0.66 (10M)	0.75 (6M)
MultiRoom-N10-S10 [80M]	0.0 (-)	0.0 (-)
MultiRoom-N12-S10 [80M]	0.0 (-)	0.0 (-)
ObstructedMaze-2Dlh [80M]	0.0 (-)	0.96 (45M)
KeyCorridorS3R3 [30M]	0.0 (-)	0.9 (18M)

frames (in millions) required to achieve the convergence value. The (-) indicates no convergence in $[xM]$ frames. As shown in the table, FoMoRL consistently outperforms the original baseline method RIDE with respect to both better final average return and time to reach convergence. RIDE is unable to learn in a number of scenarios, as it is difficult to learn efficient state representations in the partially observable RGB space. On the other hand, FoMoRL directly uses CLIP representations as state embeddings for intrinsic motivation and thereby completely bypasses the representation learning for novel state based intrinsic rewards altogether. The learning curves for both methods are present in Appendix A.1.

Discussion on Very Hard Exploration Tasks

As shown in the table, both RIDE and FoMoRL failed to learn anything on larger versions of the MultiRoom environment, i.e., MultiRoom-N10-S10 and MultiRoom-N12-S10, with 10 and 12 rooms, respectively, having a maximum room size of 10. We analyzed the training logs of these failed tasks and found that the failure is attributed to the partial observability of the RGB input image. Since the intrinsic

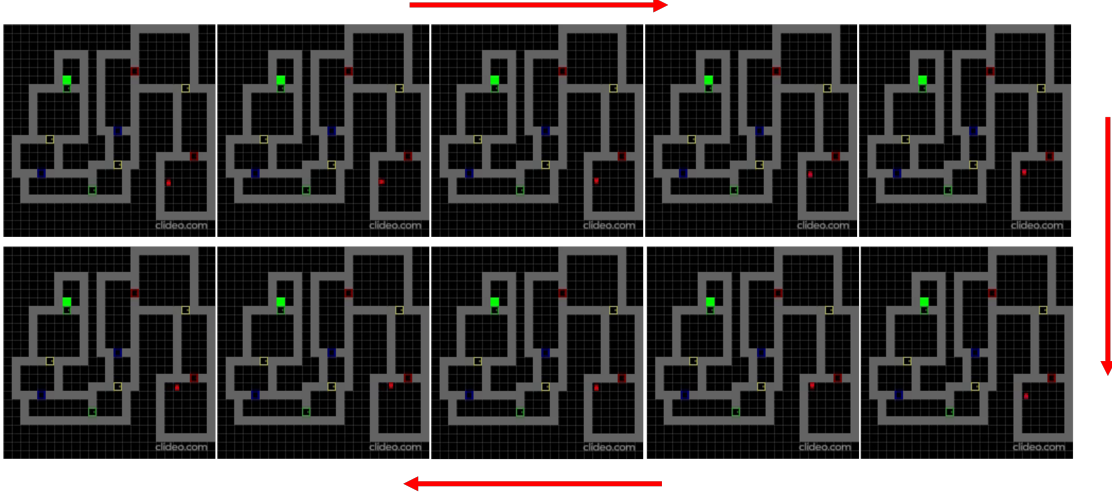


Figure 3.6: An example rollout of an agent in very hard exploration task MultiRoom-N10-S10 where agent has learnt to generate diversity within the partially observable restricted view by changing directions in a cell and then moving to the next cell. In the full rollout, the agent is able to move to the next room but soon runs out of time to solve the task.

reward module only sees a partial observable egocentric observation, it tries to generate diversity within that restricted view and gather intrinsic reward. Therefore, the agent learns to move around from one cell to another and open doors only when it is within the observable area. The moving around from cell to cell ends up taking most of the horizon time before the agent can reach the final destination. Figure 3.6 shows an example rollout (10 frames) of such an agent. We, therefore, tried to resolve this by allowing only the intrinsic module to access the full RGB image of the environment and keeping the policy input partially observable. Table 3.2 clearly shows that providing the intrinsic reward module access to fully observable RGB images resolves the issue for FoMoRL.

Discussion on Coefficient α

For some tasks, we noticed that a higher value of intrinsic reward coefficient α led the agent to completely get distracted from solving the original task but rather focus on generating more and more diversity, thereby making both RIDE and FoMoRL fail to learn. However, most environments were solvable by $\alpha=\{0.1, 0.5\}$ for RIDE and $\alpha=\{0.02, 0.005\}$ for FoMoRL. Table 3.3 provides a sensitivity analysis

Table 3.2: Performance of FoMoRL on very-hard exploration tasks N10-S10 and N12-S10 in MiniGrid with partially observable RGB images as input to the policy and fully observable RGB images as input to the intrinsic reward module. The values indicate the final mean episodic return averaged over multiple seeds after training on $[xM]$ frames. The bracketed values (xM) indicate the average number of training frames (in millions) required to achieve the convergence value.

FoMoRL	
MultiRoom-N10-S10 [80M]	0.63 (22M)
MultiRoom-N12-S10 [80M]	0.64 (24M)

Table 3.3: Sensitivity of the intrinsic reward coefficient hyper parameter (α) on final performance of FoMoRL. The value 0.0 (-) indicates no convergence after training on fixed number of frames. A dash(-) implies the hyper-parameter was not tested.

	FoMoRL (IR-0.005)	RIDE (IR-0.02)	RIDE (IR-0.1)
DoorKey-5x5	-	0.96 (5M)	0.0 (-)
MultiRoom-N7-S8	-	0.59 (29M)	0.0 (-)
ObstructedMaze-2Dlh	0.96 (45M)	0.0 (-)	0.0 (-)
KeyCorridorS3R3	0.9 (18M)	0.0 (-)	-

for the intrinsic reward coefficient (α) as intrinsic motivation-based methods tend to be very sensitive to this hyperparameter.

3.4.4 CLIP Analysis on MiniGrid

In the previous section, we showed that CLIP representations are remarkably effective across multiple MiniGrid exploration tasks. We perform a small experiment to check what exactly CLIP understands from these RGB grid images. We input the RGB images to the CLIP’s visual encoder and then input proposed language descriptions through the CLIP’s language encoder and compute a similarity score. The higher score of similarity implies that CLIP representations prefer that description over others. Figure 3.7 shows the test image, which is inputted into the CLIP’s visual encoder. The proposed language descriptions and the similarity scores are shown in Figure 3.8. Figure 3.8 shows the results of testing the CLIP

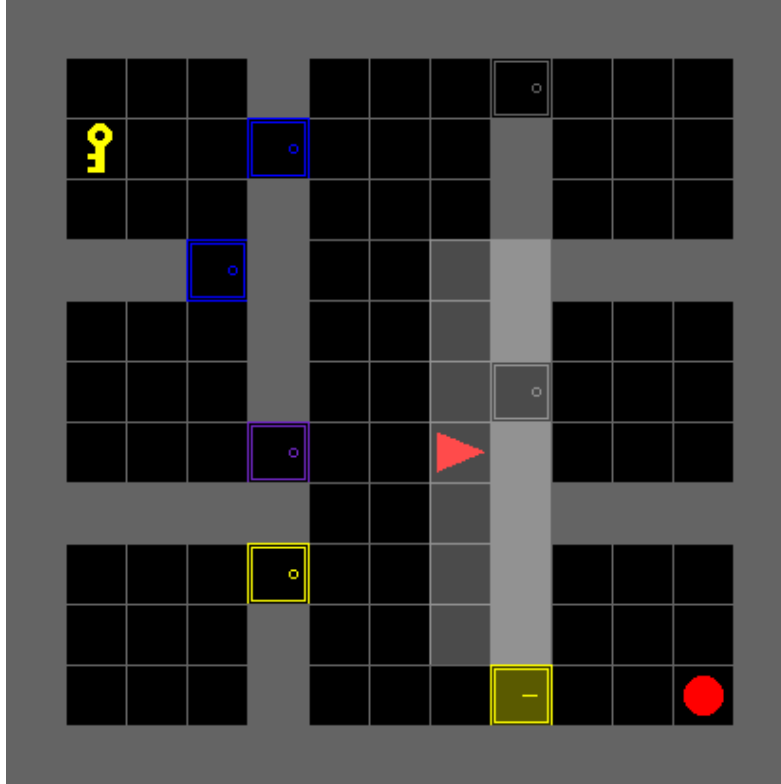


Figure 3.7: RGB Input Image for CLIP Embedding Analysis.

representations to encode the color and count of objects, their spatial information, and more fine-grained information. We test the color understanding of the CLIP representations by comparing a correct description of the color of squares with an incorrect one, as there are no green squares present in the test image (Figure 3.7). We then inspect if CLIP representations prefer a description with spatial information of the key-shaped object or not. To check the understanding of CLIP representations to represent the count of objects, we compare the number of blue squares with an incorrect description of three blue squares. Lastly, we test for more fine-grained information within the yellow squares. One of the yellow squares has a circle within it, and the other one has a dash within (semantically meaning that the door is unlocked/locked).

Figure 3.8 indicates that CLIP representations can clearly discriminate between different colors and counts of objects, their spatial understanding, and even more

Top Predictions for testing colour:

Red triangle, yellow key, grey squares, **blue** squares, yellow squares, red circle, purple square: **79.25%**

Red triangle, yellow key, grey squares, **green** squares, yellow squares, red circle, purple square: 19.73%

Top Predictions for spatial understanding of key shaped object:

Red triangle, **top left yellow key**, grey squares, blue squares, yellow squares, red circle, purple square: **56.74%**

Red triangle, **yellow key**, grey squares, blue squares, yellow squares, red circle, purple square: 42.82%

Top Predictions for testing count of blue squares:

Red triangle, top left yellow key, grey squares, **two** blue squares, yellow squares, red circle, purple square: **62.11%**

Red triangle, top left yellow key, grey squares, **three** blue squares, yellow squares, red circle, purple square: 37.67%

Top Predictions for testing fine-grained information within yellow squares:

Red triangle, top left yellow key, grey squares, two blue squares, **two yellow squares with a circle and a dash**, red circle, purple square: **87.21%**

Red triangle, top left yellow key, grey squares, two blue squares, **two yellow squares both with a circle**, red circle, purple square: 12.76%

Figure 3.8: Similarity scores for proposed language descriptions of the input test image. (a) Test to check whether CLIP representations prefer a description with the presence of a green square (incorrect) or a grey square (correct) in the test image. (b) Test to check the count of blue squares in the test image. (c) Test to check if CLIP representations prefer the description with spatial information of the key in the the test image or not. (d) Test to check more fine-grained understanding of the test image by comparing against the shapes present within the yellow square in the test image.

fine-grained information like whether a square has a mini circle or dash within. CLIP representations do not directly understand the significance of the objects within the RGB image; for example, it does not directly understand that a square indicates a door or a square with a mini circle indicates an unlocked door. However, the representations can understand and distinguish between the geometrical shapes very well, which is enough for diversity-based semantic exploration. On the contrary, CLIP representations have already been shown to be very effective for large range of tasks from text-based video retrieval [Fang et al., 2021, Luo et al., 2021], text-driven image manipulation [Patashnik et al., 2021] to embodied AI tasks [Khandelwal et al., 2022] based on realistic visual observations, for example, a kitchen scene containing a microwave.

3.5 Further Discussions and Limitations

3.5.1 Promoting Novelty in Pixel Space vs CLIP Representations

Working on novelty in CLIP representations offers several advantages over using more-direct image representations. CLIP combines visual and linguistic information, allowing agents to recognize and explore states that are semantically meaningful rather than just visually distinct. This semantic grounding enables more efficient exploration by focusing on novel and relevant states, reducing redundant state visits. Additionally, CLIP’s robust generalization capabilities, derived from its training on a vast and diverse dataset, help agents adapt more effectively to unseen scenarios. The domain-independent knowledge encoded in CLIP representations facilitates seamless transfer learning across different environments and tasks. Moreover, the compact and abstract nature of CLIP embeddings simplifies novelty detection and reduces the computational burden compared to high-dimensional image data.

3.5.2 Discussion on the choice of foundation model

In our work, we leverage the CLIP foundation model, which has been contrastively trained with vision-language data, grounding visual encoder embeddings in natural language abstractions. This enables our intrinsic reward module to promote diversity within the compact language space. The sensitivity of our method to the choice of the foundation model lies in the quality and breadth of the semantic knowledge encoded in the visual representations. For example, object detection, semantic segmentation and depth estimation are examples of a few desirable properties for such representations to be useful for intrinsic motivation.

Khandelwal et al. [2022] evaluate how well CLIP’s visual representations capture semantic information from input observations — essential elements for navigation-heavy embodied tasks such as object presence, localization, free space and reachability of objects. They find that CLIP’s representations encode these elements more effectively than ImageNet-pretrained backbones. Depth estimation is another critical aspect for learning agents, allowing them to better understand their surroundings, navigate more effectively, and make informed decisions about interactions with objects and obstacles. Anasosalu Vasu et al. [2024] recently showcased improving caption quality in image-text datasets significantly enhances CLIP’s visual representations, surpassing recent pretraining methods and achieving substantial improvements in semantic segmentation and depth estimation tasks, with notable gains in efficiency and performance.

Recent advancements in large language models (LLMs) and foundation models can further enhance this approach. Models with more extensive training on diverse datasets and improved understanding of context could provide even richer and more nuanced semantic representations. This progress would potentially improve the precision and relevance of intrinsic rewards, enabling more effective exploration in complex environments. For example, instead of relying on representations of a single foundation model, we can leverage multiple foundation models that excel in

specific tasks, such as DinoV2 [Oquab et al., 2023] and Segment Anything [Kirillov et al., 2023] models, for outstanding depth estimation and zero-shot segmentation.

3.5.3 Limitations

Biases and Hallucinations

It is important to acknowledge that relying on third-party foundation models poses significant risks in real-world deployments. One major concern is the presence of inscrutable biases inherent in these models. Since foundation models like CLIP are trained on vast and diverse datasets, they can inadvertently learn and propagate societal biases present in the data. This can result in biased decision-making processes that may unfairly disadvantage certain groups or lead to unethical outcomes. Additionally, these models are prone to generating hallucinations, where the model produces outputs that are not grounded in the input data. In critical applications such as autonomous vehicles or robotic surgery, such hallucinations can have severe consequences, leading to incorrect actions or interpretations that compromise safety and effectiveness. It is also important to note that foundation models often produce stochastic and unreliable outputs, leading to potential over-estimation of novelty due to meaningless fluctuations over time. This variability can cause the agent to misinterpret these fluctuations as novel states, potentially hindering effective exploration and learning. Therefore, while leveraging advanced foundation models can significantly enhance exploration strategies in RL, it is crucial to conduct thorough evaluations and implement mitigation strategies to address these risks.

Episodic vs Lifelong Novelty

Our approach promotes agents to enhance diversity within a single episode but does not monitor novelty across episodes. This limitation can result in agents potentially exploring similar states in each episode, thereby reducing the overall sample efficiency of our method. In contrast, methods like RND [Burda et al., 2018]

accumulate prediction knowledge over multiple episodes. To mitigate this, future work can investigate the use of foundation models with methods such as Never Give Up (NGU) [Badia et al., 2020], which combines an episodic novelty module that can rapidly adapt within one episode with RND as a lifelong novelty module.

3.6 Conclusion and Future Work

We showcase that language abstractions incorporated within foundation models such as CLIP can be very useful for semantic exploration in RL, even in grid-based environments. Our proposed method, FoMoRL, can be applied independently of the choice of RL algorithm, intrinsic motivation scheme, or foundation model type. FoMoRL achieves state-of-the-art performance with a significantly higher average return and solves the task much faster than the baseline RIDE method. This work shows a proof of concept for leveraging foundation models for semantic novelty in RL.

Future Work

Future work may investigate realistic 3D environments like Habitat ObjectNav Challenge [Batra et al., 2020] and language conditioned tasks such as CALVIN [Mees et al., 2022]. It would also be interesting to leverage the language encoder from CLIP alongside the visual encoder to enable semantic exploration in language-conditioned tasks.

Part II : Cooperative Multi-Agent RL

Abstract

We now switch our focus to cooperative multi-agent settings where a team of agents must learn to coordinate and solve a shared task. Often, we want the agents to act solely based on their local observations in a decentralized manner during execution time due to partial observability and communication constraints. So far, we have discussed exploration in single-agent settings. Compared to single-agent settings, MARL suffers from *additional* unique challenges that can impede agents' learning. Learning control policies for such settings is challenging as agents must search the joint action space, which grows exponentially with the number of agents. This makes the task of exploration harder as the transition dynamics of the environment now depend on the actions of all the agents. Applying independent forms of learning where agents ignore other agents in the environment might lead to learning instability arising from a non-stationary environment. On the other hand, learning a centralized controller with joint action of all agents becomes intractable with an exponentially increasing number of joint actions. It does not allow for decentralized execution of agents without expensive communication. We present a brief survey of cooperative MARL methods in Chapter 4 to familiarize the reader with various methods.

4

Brief Survey of Cooperative Multi-Agent RL

Contents

4.1	Independent Learners	53
4.2	Centralized Execution	55
4.3	Communication	56
4.4	Centralized Training with Decentralized Execution (CTDE)	57
4.4.1	Value Function Factorization	57
4.4.2	Actor-Critic Methods	62
4.5	Diversity in MARL	64

One of the simplest approaches to MARL is Independent Q -Learning [IQL, Tan, 1993], in which each agent independently learns its own policy while treating other agents as part of the environment. Each agent learns a policy that solely conditions on its own local action-observation history (τ) and maximizes the team reward. Since IQL directly learns decentralized policies, it avoids the scalability problems of centralized learning. On the other hand, it creates a separate issue. Since a learning IQL agent treats other agents who are themselves learning as part of the environment, the environment becomes non-stationary from the point of view of each agent. It implies that an agent can undertake the same action in the

same state at two distinct points of training and observe different signals due to simultaneous exploration and learning by other agents. Moreover, IQL, even under tabular settings, does not have any convergence guarantees. Despite the learning instability, IQL has shown to work well in practice [Matignon et al., 2012].

Dealing with Non-Stationarity: There are several works that try to alleviate the stability issues associated with IQL. Nair et al. [2003] proposes to alleviate the problem of non-stationarity by allowing only a single agent to train and fix the policies of other agents. However, learning rich cooperative behaviors (e.g., coordinating at a single time step) requires agents to update their policies simultaneously. Their approach assumes agents do not know each other’s observations and actions at run-time, but they know each other’s policies. To resolve the non-stationarity issue, Hyper Q -learning [Tesauro, 2003] reduces each agent learning problem to a stationary, much larger environment by augmenting the state space of agents with an estimate of other agents’ policies, computed via Bayesian inference. This helps with the non-stationarity associated with IQL, as agents can now make inferences over possible actions of other agents based on estimated strategies. However, this increases the size of state space, leading to poor scalability. Conitzer and Sandholm [2007] propose AWESOME (Adapt When Everybody is Stationary, Otherwise Move to Equilibrium), in which agents change their policies only when other agents appear to be not changing theirs. This approach heavily relies on the detection of whether other agents’ policies are changing or not and is therefore restricted to tabular settings. When other agents do not appear stationary, AWESOME retreats to a precomputed equilibrium strategy. Claus and Boutilier [1998] investigate the difference between independent learners and joint action learners (JALs), which learn value functions conditioned on joint actions. They compute Q -values for each agent conditioned on the policies of other agents as:

$$Q^a(u^a) = \sum_{\mathbf{u}^{-a}} Q(\mathbf{u}^{-a}, u^a) \boldsymbol{\pi}(\mathbf{u}^{-a}),$$

thereby avoiding the non-stationarity problem. However, this approach is very expensive as each agent needs access to the joint Q function and other agent policies, and joint action space grows exponentially with the number of agents.

Independent execution in MARL requires agents to act solely based on their own local observations under partial observability without communication. However, training these agents can be done in either a centralized manner with access to other agents' observations and global state, known as CTDE [Kraemer and Banerjee, 2016] or training can be done in a completely independent manner, treating other agents as part of the environment, known as independent learners.

4.1 Independent Learners

Lauer and Riedmiller [2000] proposes a MARL approach (under full observability with access to a full state of the system) called distributed Q -learning, which assumes that the low returns are a result of bad exploration by the team member and updates the Q -values only when there is a guaranteed improvement. They update Q -value estimates of an agent using:

$$Q_t^a(s, u^a) = \max_{(\mathbf{u}^{-a}, u^a)} Q_t(s, \mathbf{u}).$$

To tackle the issue of multiple joint actions having similar values, they learn an additional policy that only changes its actions if an improvement in the Q -values occurs. They show their approach converges to optimal policy in deterministic tabular settings, given that there is a unique optimal joint action. However, this approach requires tabular settings, fully observable deterministic settings, and a positive reward function, which severely restricts the applicability and scalability of their approach. Hysteretic Q -learning proposed by Matignon et al. [2007] assumes that the low returns are a result of environment uncertainty and updates Q -value by learning rate α when the TD-error is positive and with learning rate β otherwise, where $\beta < \alpha$. Therefore, unlike distributed Q -learning, they allow for the gradual degradation of overestimated Q -values.

Recently, a lot of advancements such as the use of replay memory, target networks [Mnih et al., 2015], and other extensions of DQN such as double Q -learning [van Hasselt et al., 2016], prioritized experience replay [Schaul et al., 2015b] and dueling DQN [Wang et al., 2016] has dramatically improved the stability and scalability of single-agent RL with function approximators [Hessel et al., 2018]. Tampuu et al. [2017] extend this approach to deep networks using DQN [Mnih et al., 2015]. One of the key advancements that helped achieve this was the use of replay memory, which not only stabilizes the training of a deep neural network but also improves sample efficiency by repeatedly using experience tuples for training. However, the combination of IQL with the replay buffer becomes problematic as the dynamics and policies that generate the data in the replay buffer do not reflect the current dynamics and policies, confusing the agents with obsolete experiences. To avoid this problem, Leibo et al. [2017] limits the use of experience replay to short, recent buffers, and Foerster et al. [2016] disables the replay memory altogether. However, these workarounds limit the sample efficiency and scalability of deep MARL. To resolve these issues, Foerster et al. [2017] propose two approaches. The first approach augments the experience tuples in the replay memory with a probability of joint actions according to the policies used at that time and then computes an off-policy style correction when the experience is sampled for training. They derive the importance of the weighted loss function for fully and partially observable settings. However, importance sampling often yields importance ratios with large and unbounded variance, and clipping the ratios introduces bias. The second approach augments the observation function of agents with iteration number and the current rate of exploration (ϵ), allowing each agent’s observation to disambiguate when, in training, an experience was sampled from.

Omidshafiei et al. [2017] consider multi-task multi-agent cooperative games with IQL agents learning in a partially observable setting. Firstly, they propose Concurrent Experience Replay Trajectories (CERTs) to store the experiences of agents concurrently and counter the non-stationary nature of local experiences when

sampling is done independently for each agent. Secondly, they propose HDRQNs (Hysteretic Dynamic Recurrent Q Networks) based on DRQN [Hausknecht and Stone, 2015] and Hysteretic Q -learning [Matignon et al., 2007] to learn IQL policies for each task for each agent. Finally, they distill (loss function motivated from Rusu et al. [2015]) the learned policies for various tasks for an agent into a single unified policy such that task identity is not required during execution. Panait et al. [2006b] propose *Leniency*, which ignores updates that lead to a reduction in the Q values (due to low returns arising from simultaneous exploration by teammates) with a probability that decays with more training. Palmer et al. [2017] extends Leniency to deep RL settings and Wei and Luke [2016] show that Leniency performs better than hysteretic Q learning in cooperative games. Unlike hysteretic Q learning and Leniency, Palmer et al. [2018] distinguishes between miscoordinated actions amongst agents and environment uncertainty by computing intervals for each action $u \in \mathcal{U}$ with a lower bound corresponding to the minimum reward for coordinated behavior involving u . This enables them to detect miscoordination when a reward lesser than the lower bound is received. Lyu and Amato [2018] utilize distributional RL [Bellemare et al., 2017] to distinguish non-stationary samples based on the likelihood of returns.

4.2 Centralized Execution

Centralized learning deals with the problem of non-stationarity naturally via learning a joint action-value function. In some settings, completely decentralized execution is not required, and therefore, we can use centralized value functions directly during execution. However, these are not practical to scale owing to the exponentially increasing space of joint actions with agents. Coordination Graphs [Guestin et al., 2002] uses prior domain knowledge to reduce the computational burden by decomposing joint reward into agent local terms (depending only on a subset of the action-observation space) such that conditional independence between agents can be exploited. Depending on the topology of the predefined graph, maximum joint

action can be computed based on message-passing algorithms [Kschischang et al., 2001]. Böhmer et al. [2020] extend coordination graphs to deep RL settings where they assume pairwise interactions between agents and improve sample efficiency with parameter sharing and low-rank approximations. However, specifying a coordination graph requires significant domain knowledge, which is usually unavailable for complex tasks. Zhou et al. [2019] similarly decompose a joint action-value function into pairwise interactions with coordinate ascent. Kok and Vlassis [2006] use the framework of coordination graphs to encode dependencies between agents and learn a joint action-value function in predefined states where coordination is required. For other states, they utilize independent learning to reduce computational burden.

4.3 Communication

There has been much work in the area of multi-agent communication, where the agents are able to communicate during execution. Additionally, the agents must also deal with partial observability where they are unable to view the full state, or the observations of the other agents, and instead must condition their actions on all their observation history. DIAL [Foerster et al., 2016] uses parameter sharing as well as gradient flow between agents to promote communication via backpropagation and allow limited communication during execution with unrestricted communication during learning. They relax the channel to be continuous for end-to-end training. Mordatch and Abbeel [2017] instead use gumble-softmax trick [Jang et al., 2016] to back propagate gradients through a discrete channel. Clustered Deep Q-Network (CDQN) [Pageaud et al., 2019] use a hierarchy in which low-level agents are clustered in groups and make decisions by consensus voting, and high-level agents manage low-level agents in a centralized setting. CommNet [Sukhbaatar et al., 2016] exchanges information via a centralized architecture in which hidden states of all agents are shared amongst all agents as a means of information sharing. BiCNet [Peng et al., 2017] uses bi-directional RNNs [Schuster and Paliwal, 1997] in an actor-critic framework for inter-agent communication. MAGNet [Malysheva et al., 2018] encodes

a relevance graph of the environment to each agent and allows agents to learn a central graph structure. Omidshafiei et al. [2019] performs peer-to-peer teaching with agents penalized to communicate their action recommendations to other agents.

4.4 Centralized Training with Decentralized Execution (CTDE)

Learning a centralized action-value function resolves the problem of nonstationarity associated with independent learning by conditioning the value function on the full state (or concatenated observations) and joint actions. However, it creates a separate issue of how to extract the correct decentralized policies from it such that agents can solely act based on their local observations. One way is to learn centralized critics and independent policies in an actor-critic framework, the majority of which are on-policy methods leading to poor sample efficiency. Another option is to learn a centralized but factored value function that admits efficient decentralization.

4.4.1 Value Function Factorization

Lin et al. [2019] propose to distill a learned centralized policy into a decentralized policy by mimicking the actions using supervised learning. They learn a centralized value function with access to observations of all agents. In order to avoid the explosion of observation space (for input) and action space (for output) with an increasing number of agents, they propose to use DDPG [Lillicrap et al., 2016] with Gumbel-Softmax trick [Jang et al., 2016] for action selection in discrete environments. Sunehag et al. [2018] proposes value decomposition networks (VDN) based on the literature of coordination graphs [Guestrin et al., 2002], which decomposes the joint action-value function (referred to as Q_{jt}) as a sum of independent utilities of agents:

$$Q_{jt}(\boldsymbol{\tau}, \mathbf{u}) = \sum_a Q^a(\tau^a, u^a).$$

Decentralization is trivially achieved as each Q^a still conditions only on individual action-observation histories and can be represented by an agent network that shares

parameters across all agents. The joint action value function Q_{jt} can be trained using Deep Q -Networks (DQN) [Mnih et al., 2015]. Importantly, each agent taking a greedy action with respect to its own Q -values is the same as selecting the greedy joint action. This allows tractable maximization of Q_{jt} in the Q learning update without exhaustively evaluating exponentially many joint actions and ensures consistency between the decentralized and centralized policies called Individual Global-Max principle [IGM, Son et al., 2019]:

$$\arg \max_{\mathbf{u}} Q_{jt}(\boldsymbol{\tau}, \mathbf{u}) = \left\{ \arg \max_{u^1} Q(\tau^1, u^1), \dots, \arg \max_{u^n} Q(\tau^n, u^n) \right\}. \quad (4.1)$$

The drawback of VDN is that it ignores any extra state information available during centralized training and severely limits the representational capacity of Q_{jt} by constraining it to be a simple sum of the agent’s utilities. The extension VDN-S [Rashid et al., 2020b] makes additional use of the available joint state s_t during training by extending the action-value function with a state-dependent bias term v , parameterized by ϕ , that is ignored during execution:

$$Q_{jt}(\boldsymbol{\tau}, \mathbf{u}) = \sum_a Q^a(\tau^a, u^a) + V(s_t; \phi).$$

To overcome the representational capacity of VDN, QMIX [Rashid et al., 2018, 2020b] proposes that in order to extract decentralized policies, we only need to ensure the IGM principle, which can be achieved by ensuring that Q_{jt} is monotonic with respect to each agent’s utilities:

$$\frac{\partial Q_{jt}}{\partial Q^a} \geq 0. \quad (4.2)$$

QMIX builds upon the architecture of VDN, but instead of factoring Q_{jt} as a simple sum of the agent’s utilities, they allow for a much richer mixing. Each agent uses a DRQN [Hausknecht and Stone, 2015] network to produce individual utilities, which are fed into a mixing network that combines/mixes them to produce the joint action Q_{jt} values. In order to ensure the monotonicity constraint from Equation 4.2, the weights of the mixing network are constrained to be non-negative. The weights of

the mixing network are produced by hypernetworks [Ha et al., 2016], which take in the state as input, thereby leveraging extra state information during training. Therefore, QMIX can represent any value function that factors into nonlinear *monotonic* combination of individual agent utilities such that:

$$Q^a(s_t, u^a) > Q^a(s_t, u'^a) \Leftrightarrow Q_{jt}(s_t, (\mathbf{u}^{-a}, u^a)) > Q_{jt}(s_t, (\mathbf{u}^{-a}, u'^a)).$$

However, this monotonic restriction prevents QMIX from correctly representing non-monotonic value functions [Mahajan et al., 2019], where an agent’s best action depends on what actions the other agents choose, i.e., in environments in which agents must coordinate at the same time step. For example, consider a predator-prey task where *at least three* agents need to coordinate to capture prey, and any capture attempts by fewer agents are penalized with a penalty of magnitude p . As a result of naive ϵ -greedy exploration and monotonicity restriction, both VDN and QMIX tend to get stuck in a sub-optimal equilibrium [also called the relative overgeneralization (RO) pathology, Panait et al., 2006a, Wei et al., 2018] in which agents simply avoid the prey [Mahajan et al., 2019, Böhmer et al., 2020]. The monotonically factorized joint action-value function lacks the representational capacity to distinguish the values of coordinated and uncoordinated joint actions during exploration in this non-monotonic task and may, therefore, fail to learn the optimal policy.

MAVEN [Mahajan et al., 2019] theoretically show that the monotonic joint action-value function of QMIX and VDN suffers from sub-optimal approximations on non-monotonic tasks under both uniform and ϵ -greedy exploration. It tries to address this problem through committed exploration during training by learning a diverse ensemble of monotonic joint action-value functions conditioned on a latent variable and optimizing the mutual information between the joint trajectory and the latent variable. This resulted in slightly improved performance on some SMAC maps over QMIX. Son et al. [2019] propose QTRAN, which addresses the monotonic restriction of QMIX by learning a (decentralizable) VDN-factored joint action-value function along with an unrestricted centralized critic. The corresponding utility functions

are distilled from the critic by solving a linear optimization problem involving all joint actions, but its exact implementation is computationally intractable, and the corresponding approximate versions using L_2 penalties via stochastic gradient descent have unstable performance. QTRAN++ [Son et al., 2020] is an extension of QTRAN, which tries to bridge the theoretical and empirical performance gap. Rashid et al. [2020a] learns a QMIX-factored joint action-value function along with an unrestricted centralized critic and proposes explicit weighting mechanisms to bias the monotonic approximation of the optimal joint action-value function towards important joint actions. In tabular settings, idealized centralized weighting proposes to weight sub-optimal actions lower than optimal ones, and optimistic weighting assigns a higher weighting to those joint actions that are underestimated relative to unrestricted Q , similar to Hysteretic Q -learning. However, this unrestricted value is only a critic of the factored model, which itself is prone to RO and often fails in practice due to insufficient ϵ -greedy exploration.

QPLEX [Wang et al., 2020a] uses a duplex dueling [Wang et al., 2016] network architecture to factorize the joint action-value function with linear decomposition structure and transforms the IGM consistency into constraints on the value range of the advantage function. Yang et al. [2020b] uses a multi-head attention mechanism [Vaswani et al., 2017] to replace the QMIX mixing network so as to model each individual agent’s impact at a per-agent level. SMIX proposed by Wen et al. [2020] replaces 1-step QMIX targets with SARSA(λ) targets. de Witt et al. [2020] extends QMIX to continuous action settings and propose COVDN and COMIX, which uses cross entropy method [De Boer et al., 2005] to perform action selection for each agent and make learning tractable in continuous settings.

Q -value path decomposition (QPD) proposed by Yang et al. [2020a] decomposes global joint Q -values into individual agent utilities using an integrated gradient attribution technique for better credit assignment. QR-MIX [Hu et al., 2020] combines distributional RL [Bellemare et al., 2017] with mixing networks and decomposes joint-action return *distribution* into individual agent utilities using

quantile regression along with a flexible loss function to approximate the monotonicity. Their distributional approach and flexible loss function make them robust against both the learning uncertainty of returns and the monotonicity constraint. TESSERACT [Mahajan et al., 2021] proposes a tensor decomposition-based method for learning arbitrary action-value functions; they also provide sample complexity analysis for their method.

Böhmer et al. [2019] propose independent centrally-assisted Q -learning (ICQL), which improves IQL by introducing an intrinsically motivated centralized agent sharing control and replay buffer with the IQL agent, hence stabilizing the exploratory impact of agents. The independent agents are trained solely on extrinsic rewards while still making use of the exploration incentives. While intrinsic motivation methods such as curiosity drive effective exploration [Burda et al., 2019] for individual agents, it does not take into account the transition-dependent nature of MARL and other agent interactions. To resolve this, Wang et al. [2020c] presents two intrinsic reward-based exploration methods: information-theoretic influence (EITI) and decision-theoretic influence (EDTI). EITI captures transition interaction between agents through information gain between one agents state transition given other agents state and action. To more explicitly quantify the value of these interactions, they propose EDTI, which computes the difference between agents value function and a counterfactual value function which does not consider the state and action of another agent. Zheng and Yue [2018] propose to coordinate exploration between agents by shared latent variables, whereas Jaques et al. [2018] propose intrinsic rewards for agents based on their casual influence over other agents by simulating alternate actions for an agent at a time step and their corresponding effect on other agents.

Mean field multi-agent RL [Yang et al., 2018] considers the average effect of all other agents by a mean action and proposes mean-field Q -learning and mean-field actor-critic by approximating agent-level joint action-value critic $Q^a(s, \mathbf{u})$ using Taylor’s theorem as $Q^a(s, u^a, \mathbf{u}^{-a})$, where $\mathbf{u}^{-a}$ denotes a mean action from

neighboring agents. However, the mean action cannot discriminate between different agents and is thus unsuitable for learning cooperation. Graph convolutional RL [Jiang et al., 2018] models agents a node in a graph with local observations serving as features of the node. They use graph convolutions with multi-head attention mechanism [Vaswani et al., 2017] to extract relations amongst agents and convolve the learned features to learn cooperative policies.

4.4.2 Actor-Critic Methods

MADDPG [Lowe et al., 2017] uses deterministic policy gradients (DDPG) [Silver et al., 2014, Lillicrap et al., 2016] to learn independent policies for each agent along with a per-agent centralized critic which incorporates information from other agents, and therefore addresses the non-stationarity problem. They test their method on multi-agent scenarios with continuous action spaces and show that MADDPG outperforms other decentralized learning methods on a variety of cooperative and competitive tasks. However, learning a centralized critic suffers from poor scalability with more agents due to increased input size. Instead of learning a centralized critic for each agent, COMA [Foerster et al., 2018] learns a single centralized critic conditioned on the state and joint actions in fully cooperative settings. To perform effective multi-agent credit assignment, they use a counterfactual baseline inspired by difference rewards [Wolpert and Tumer, 2002] to marginalize out an agent’s actions while keeping other agents’ actions fixed. This enables an agent to directly evaluate the contribution of its actions to the team reward and assign credit appropriately. COMA computes the advantage function and the policy gradient as follows:

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \mathbb{E}_{\pi} \left[\sum_a \nabla_{\theta} \log \pi(u^a | \tau^a) A^a(s, \mathbf{u}) \right] \\ A^a(s, \mathbf{u}) &= Q(s, \mathbf{u}) - \underbrace{\sum_{u'^a} \pi(u'^a | \tau^a) Q(s, (\mathbf{u}^{-a}, u'^a))}_{\text{counterfactual baseline}}.\end{aligned}$$

To compute the marginalization over agent actions efficiently, they propose an efficient critic architecture. However, as pointed out by Vasilev et al. [2021],

COMA suffer from poorly designed neural network architectures which can result in inconsistent estimates of joint action-value function and thereby inhibit good policy gradient updates. To alleviate the curse of dimensionality associated with learning centralized critics with an increasing number of agents, Iqbal and Sha [2019a] propose per-agent critics with an attention-based mechanism that selects relevant information for each agent at every time step. Their approach has added benefits of allowing different action spaces for each agent and allowing both shared and individual rewards as well.

Du et al. [2019] proposes to learn individual intrinsic rewards for each agent by defining a proxy reward function as a sum of external joint reward and agent-specific intrinsic reward [Zheng et al., 2018]. They propose a bi-level optimization such that the intrinsic reward parameters are learned to maximize the joint team reward, and the policy of each agent maximizes the proxy reward function in an actor-critic framework. However, their approach results in poor computational efficiency owing to bi-level optimization. Off-Policy Multi-Agent Decomposed Policy Gradients (DOP) proposed by Wang et al. [2020e] introduces value function decomposition for multi-agent actor-critic algorithms by training a centralized critic as a weighted linear combination of individual agent critics. They propose to use off-policy data by adapting the tree backup technique [Precup, 2000] to multi-agent settings and leverage the factored critic representation for tractable off-policy evaluations of stochastic policies, resulting in better sample efficiency. In addition, the factored critic also addresses the centralized-decentralized mismatch (CDM) issue, which negatively affects the learning of agents' policies during training arising from joint critic getting affected by suboptimality of other agents' policies.

Zhou et al. [2020] formulated centralized critic as a hypernetwork [Ha et al., 2016], which takes as input the global state and maps it to a set of weights. These set of weights, in turn, mixes individual agent policies by mapping them to joint Q estimate. This enables agents to implicitly perform credit assignment by allowing a representation of the state to integrate into action-value gradients

through multiplicative association. For the sustained level of exploration, they propose adaptive entropy regularization, which dynamically adapts the magnitude of entropy parameters during training. de Witt et al. [2018] propose a multi-agent actor-critic algorithm that learns a hierarchical policy in cooperative settings by exploiting common knowledge between groups of agents. de Witt et al. [2020] extends MADDPG by factoring the centralized critic and shows that it leads to performance improvements over MADDPG owing to better credit assignment in the centralized-decentralized mismatch (CDM) issue. All algorithms, except MADDPG and DOP above, are on-policy methods that have poor sample efficiency compared to value-based MARL methods. Despite the progress in multi-agent actor-critic methods, they still significantly underperform value-based methods on popular benchmarks such as Starcraft Multi-Agent Challenge (SMAC) [Samvelyan et al., 2019].

4.5 Diversity in MARL

Unsupervised skill discovery [Gregor et al., 2016, Eysenbach et al., 2018] allows an agent to learn diverse skills (policies conditioned on latent variables) in an environment without access to any reward signal. This enables agents to pick up skills without conforming to any task, which can be later used in a variety of tasks. The key to learning diverse skills is to incentivize the diversification of trajectories and maximize the mutual information between the trajectories or observed states and the latent variable. Learning skills in MARL without a reward function could be an efficient way to improve the sample efficiency as agents can learn to cooperate in a diverse variety of ways, which can be used for a wide variety of coordination tasks. However, in MARL, the presence of other agents requiring coordination requires the need to learn skills in a centralized fashion, which is not scalable.

Yang et al. [2019] introduce a bi-level hierarchical MARL algorithm to learn diverse skills for each agent via independent Q -learning and high-level centralized policy picks latent skill variables. However, they do not consider diversity in high-level policy, which limits the ability of the algorithm to learn to cooperate in

complex scenarios. Yang et al. [2020c] remove the restrictive assumptions of joint value function decomposition using a determinantal point process to acquire diverse behaviors without explicit structural constraints imposed on value functions. Iqbal and Sha [2019b] formulate a hierarchical approach in which a low-level controller trains policies on intrinsic rewards, and a high-level controller selects policies learned on diverse intrinsic rewards. They define intrinsic rewards as varying combinations of individual agents' novelty functions, which take into account the novelty of observations encountered. The policies for each reward are trained using the soft-actor-critic method [Haarnoja et al., 2018]. Wang et al. [2020b] propose ROMA, which learns role-conditioned policies for agents without any domain knowledge or prior structure on roles, such that agents with similar roles exhibit similar behaviors, hence sharing experiences and improving sample efficiency. They propose to learn roles that maximize the mutual information between the role and trajectory of an agent, given the current observation. To ensure the specialization of roles, they propose role regularization methods.

Part II(A) : Cooperative Value-Based Multi-Agent RL

Abstract

We now switch our focus to *value-based* cooperative multi-agent settings under the centralized training and decentralized execution (CTDE) framework. As discussed in the last chapter, CTDE allows agents to leverage the global state and observations of other agents during training. This allows agents to overcome the instability arising from a non-stationary environment by learning a centralized value function monotonically factored into independent agent utilities, allowing for completely decentralized policies. However, the value factorization creates a novel challenge for multi-agent tasks requiring agents to coordinate simultaneously (i.e., take a specific action at the same time step simultaneously). We focus on such tasks in Chapter 5. We show that most current state-of-the-art (SOTA) deep MARL methods take a long time to learn even simple grid world tasks. The value factorization can contribute to learning pathologies like Relative Overgeneralization (RO), where a sub-optimal Nash equilibrium in the joint action space is preferred over an optimal Nash equilibrium. Furthermore, the policies learned by current SOTA methods may fail to adapt to simple changes in the environment (such as change in reward function) and may require learning even slightly changed tasks from scratch, restricting the real-world usage of MARL. Therefore, reusing knowledge that can come from previous experience or other agents is indispensable to scaling up MARL algorithms. We propose universal value exploration, which can solve tasks requiring significant coordination among agents where other state-of-the-art MARL methods fail. Our method provides zero-shot generalization to harder tasks.

Chapter 6 focuses on improving exploration in multi-agent settings. During training, CTDE algorithms mostly search in the joint action-observation space, which grows exponentially with the number of agents. This makes learning difficult when the number of agents is large. Role-based learning holds promise for achieving scalable multi-agent learning by decomposing complex tasks using roles. However, how to efficiently discover such a set of roles is largely unclear. We propose a method to decompose joint action spaces into restricted role action spaces by clustering actions according to their effects on the environment and other agents. Learning a

role selector based on action effects makes role discovery much easier because it forms a bi-level learning hierarchy: the role selector searches in a smaller role space and at a lower temporal resolution, while role policies learn in significantly reduced primitive action-observation spaces. We further integrate information about action effects into the role policies to boost learning efficiency and policy generalization. With these advances, our method outperforms the current state-of-the-art MARL algorithms on 9 of the 14 scenarios that comprise the challenging SMAC benchmark and achieves rapid transfer to new environments with three times the number of agents.

5

Universal Value Exploration

Contents

5.1	Introduction	70
5.2	Relative Overgeneralization	71
5.3	Background on Successor Features	72
5.3.1	Task based Universal Value Functions	72
5.3.2	Successor Features	73
5.3.3	Universal Successor Features and Generalized Policy Improvement	73
5.4	Illustrative Example	75
5.5	Multi-Agent Universal Successor Features	76
5.5.1	Decentralized Execution	78
5.5.2	Decentralized Local GPI	78
5.5.3	Training	79
5.6	UneVEn	80
5.6.1	Action-Selection Schemes	81
5.7	Related Work on Multi-Agent RL	83
5.7.1	Exploration	83
5.8	Related Work on Successor Features	84
5.9	Experiments	85
5.9.1	Domain 1 : m-Step Matrix Game	85
5.9.2	Domain 2 : Cooperative Predator-Prey	86
5.9.3	Domain 3 : Starcraft Multi-Agent Challenge (SMAC)	93
5.10	Conclusion, Limitations and Future Work	99

5.1 Introduction

Learning control policies for cooperative MARL remains challenging as agents must search the joint action space, which grows exponentially with the number of agents. Current state-of-the-art value-based methods such as VDN [Sunehag et al., 2018] and QMIX [Rashid et al., 2020b] learn a *centralized* joint action value function as a *monotonic* factorization of *decentralized* agent utility functions. Due to this monotonic factorization, the joint action value function can be decentrally maximized as each agent can simply select the action that maximizes its corresponding utility function, known as the Individual Global Maximum principle [IGM, Son et al., 2019].

This monotonic restriction cannot represent the value of all joint actions, and an agent’s utility depends on the policies of the other agents [nonmonotonicity, Mahajan et al., 2019]. Even in collaborative tasks, this can exhibit *relative overgeneralization* [RO, Panait et al., 2006a] when the optimal action’s utility falls below that of a suboptimal action [Wei et al., 2018, 2019]. While this pathology depends in practice on the agents’ random experiences, we show in Section 5.4 that in expectation, RO prevents VDN from learning a large set of predator-prey games during the critical phase of uniform exploration.

QTRAN [Son et al., 2019] and WQMIX [Rashid et al., 2020a] show that this problem can be avoided by weighting the *joint actions of the optimal policy* higher. They propose to deduce this weighting from an unrestricted joint value function that is learned simultaneously. However, this unrestricted value is only a critic of the factored model, which itself is prone to RO, and often fails in practice due to insufficient ϵ -greedy exploration. MAVEN [Mahajan et al., 2019] improves exploration by learning an ensemble of monotonic joint action value functions through committed exploration and maximizing diversity in the joint team behavior. However, it does not specifically target optimal actions and may not work in tasks with strong RO.

The core idea of this chapter is that even when a *target task* exhibits RO under value factorization, there may be *similar tasks* that do not. If their optimal actions overlap in some states with the target task, executing these simpler tasks can implicitly weight exploration to overcome RO. We call this novel paradigm Universal Value Exploration (UneVEn). To learn different MARL tasks simultaneously, we extend Universal Successor Features [USFs, Borsa et al., 2018] to Multi-Agent USFs (MAUSFs), using a VDN decomposition. During execution, UneVEn samples task descriptions from a Gaussian distribution centered around the target and executes the task with the highest value. This biases exploration towards the optimal actions of tasks that are similar but have already been solved. Sampling these reduces RO for every task that shares the same optimal action and thus increases the number of solvable tasks, eventually overcoming RO for the target task, too. This is different from exploration methods like MAVEN, which keep the task the same but reweigh explorative behaviors using the task returns. We show in Section 5.4 on a classic RO example that UneVEn can gradually increase the set of *solvable tasks* until it includes the target task.

We evaluate our novel approach against several ablations in predator-prey tasks that require significant coordination among agents and highlight the RO pathology, as well as other commonly used benchmarks like StarCraft II [Samvelyan et al., 2019]. We also show empirically that UneVEn significantly outperforms current state-of-the-art value-based methods on target tasks that exhibit strong RO and in zero-shot generalization [Borsa et al., 2018] to other reward functions.

5.2 Relative Overgeneralization

Relative Overgeneralization (RO) [Wei et al., 2018] is a learning pathology that occurs when a sub-optimal joint action is preferred over an optimal one because each agent’s individual action in the sub-optimal joint action is favored when matched

to arbitrary actions from the collaborating agents i.e.

$$Q^a(s, u^a) > Q^a(s, u'^a) \quad \text{when} \quad Q_{jt}(s, (\mathbf{u}^{-a}, u^a)) < Q_{jt}(s, (\mathbf{u}^{-a}, u'^a)).$$

Definition 1. [Taken from Yu et al., 2019] “A Q -function is relative overgeneralization (RO) free if for any agent a and any of its two actions u^a and u'^a , we always have:

$$\mathbb{E}_{\pi^{-a}} [Q_{jt}(s, u^a, \mathbf{u}^{-a})] > \mathbb{E}_{\pi^{-a}} [Q_{jt}(s, u'^a, \mathbf{u}^{-a})],$$

given that $Q_{jt}(s, u^a, \mathbf{u}^{\star^{-a}}) > Q_{jt}(s, u'^a, \mathbf{u}^{\star^{-a}})$, where π^{-a} is some arbitrary policy of all the other agents, and $\mathbf{u}^{\star^{-a}}$ is the best response action of all the other agents.”

In summary, Definition 1 implies that a Q function does not suffer from RO when it’s ordering over an agent’s action values stays consistent with the arbitrary or optimal joint policy of other agents. Due to the monotonicity restriction of QMIX [Rashid et al., 2018], VDN [Sunehag et al., 2018], and other state-of-the-art value-based MARL methods, they may fail to learn on tasks exhibiting RO. In CTDE, RO is, in expectation, a property of the task, the value factorization, and the other agents’ policies (i.e., exploration), but the sampled trajectories from the replay buffer encountered in practice decide whether or not RO occurs. So, for each task, exploration strategy, and factorization, there should be a fixed probability of encountering RO. However, this probability is different for different tasks (e.g., modulated by the punishment in predator-prey tasks discussed in Section 5.4).

5.3 Background on Successor Features

5.3.1 Task based Universal Value Functions

In this chapter, we consider tasks that differ only in their reward functions

$$R_w(s, \mathbf{u}) \equiv \mathbf{w}^\top \boldsymbol{\phi}(s, \mathbf{u}),$$

which are linear combinations of a set of basis functions $\boldsymbol{\phi} : \mathcal{S} \times \mathcal{U} \rightarrow \mathbb{R}^d$. Intuitively, the basis functions $\boldsymbol{\phi}$ encode potentially rewarded events, such as opening a door

or picking up an object. We use the weight vector $\mathbf{w}$ to denote the task with reward function $R_{\mathbf{w}}$. Universal Value Functions [UVFs, Schaul et al., 2015a] extend DQNs to learn a *generalizable* value function conditioned on tasks. UVFs are typically of the form $Q^\pi(s_t, \mathbf{u}_t, \mathbf{w})$ to indicate the action-value function of task $\mathbf{w}$ under policy π at time t as:

$$\begin{aligned} Q^\pi(s_t, \mathbf{u}_t, \mathbf{w}) &= \mathbb{E}^\pi \left[\sum_{i=0}^{\infty} \gamma^i R_{\mathbf{w}}(s_{t+i}, \mathbf{u}_{t+i}) \middle| s_t, \mathbf{u}_t \right] \\ &= \mathbb{E}^\pi \left[\sum_{i=0}^{\infty} \gamma^i \phi(s_{t+i}, \mathbf{u}_{t+i})^\top \mathbf{w} \middle| s_t, \mathbf{u}_t \right]. \end{aligned} \quad (5.1)$$

5.3.2 Successor Features

The Successor Representation [Dayan, 1993] has been widely used in single-agent settings [Barreto et al., 2017, 2018, Borsa et al., 2018] to generalize across tasks with given reward specifications. By simply rewriting the definition of the action value function $Q^\pi(s_t, \mathbf{u}_t, \mathbf{w})$ of task $\mathbf{w}$ from Equation 5.1 we have:

$$\begin{aligned} Q^\pi(s_t, \mathbf{u}_t, \mathbf{w}) &= \mathbb{E}^\pi \left[\sum_{i=0}^{\infty} \gamma^i \phi(s_{t+i}, \mathbf{u}_{t+i}) \middle| s_t, \mathbf{u}_t \right]^\top \mathbf{w} \\ &\equiv \boldsymbol{\psi}^\pi(s_t, \mathbf{u}_t)^\top \mathbf{w}, \end{aligned} \quad (5.2)$$

where $\boldsymbol{\psi}^\pi(s, \mathbf{u})$ are the Successor Features (SFs) under policy π . For the optimal policy π_z^* of task $\mathbf{z}$, the SFs $\boldsymbol{\psi}^{\pi_z^*}$ summarize the dynamics under this policy, which can then be weighted with any reward vector $\mathbf{w} \in \mathbb{R}^d$ to instantly evaluate policy π_z^* on it:

$$Q^{\pi_z^*}(s, \mathbf{u}, \mathbf{w}) = \boldsymbol{\psi}^{\pi_z^*}(s, \mathbf{u})^\top \mathbf{w}.$$

5.3.3 Universal Successor Features and Generalized Policy Improvement

Borsa et al. [2018] introduce universal successor features (USFs) that learn SFs conditioned on tasks using the *generalization* power of UVFs. Specifically, they define UVFs of the form $Q(s, \mathbf{u}, \mathbf{z}, \mathbf{w})$ which represents the value function of

policy π_z evaluated on task $\mathbf{w} \in \mathbb{R}^d$. These UVFs can be factored using the SFs property (Equation 5.2) as:

$$Q(s, \mathbf{u}, \mathbf{z}, \mathbf{w}) = \boldsymbol{\psi}(s, \mathbf{u}, \mathbf{z})^\top \mathbf{w},$$

where $\boldsymbol{\psi}(s, \mathbf{u}, \mathbf{z})$ are the USFs that generate the SFs induced by task-specific policy π_z . One major advantage of using SFs is the ability to *efficiently* do generalized policy improvement [GPI, Barreto et al., 2017], which allows a new policy to be computed for *any unseen* task based on instant policy evaluation of a *set* of policies on that unseen task with a simple dot-product. Formally, given a set $\mathcal{C} \subseteq \mathbb{R}^d$ of tasks and their corresponding SFs $\{\boldsymbol{\psi}(s, \mathbf{u}, \mathbf{z})\}_{\mathbf{z} \in \mathcal{C}}$ induced by corresponding policies $\{\pi_z\}_{\mathbf{z} \in \mathcal{C}}$, a new policy π'_w for any unseen task $\mathbf{w} \in \mathbb{R}^d$ can be derived using:

$$\begin{aligned} \pi'_w(s) &\in \arg \max_{\mathbf{u} \in \mathcal{U}} \max_{\mathbf{z} \in \mathcal{C}} Q(s, \mathbf{u}, \mathbf{z}, \mathbf{w}) \\ &\in \arg \max_{\mathbf{u} \in \mathcal{U}} \max_{\mathbf{z} \in \mathcal{C}} \boldsymbol{\psi}(s, \mathbf{u}, \mathbf{z})^\top \mathbf{w}. \end{aligned} \quad (5.3)$$

Setting $\mathcal{C} = \{\mathbf{w}\}$ allows us to revert back to UVFs, as we evaluate SFs induced by policy π_w on task $\mathbf{w}$ itself. However, we can use any set of tasks that are similar to $\mathbf{w}$ based on some similarity distribution $\mathcal{D}(\cdot|\mathbf{w})$. The computed policy π'_w is guaranteed to perform no worse on task $\mathbf{w}$ than *each* of the policies $\{\pi_z\}_{\mathbf{z} \in \mathcal{C}}$ [Barreto et al., 2017], but often performs much better. SFs thus enable efficient use of GPI, which allows *reuse* of learned knowledge for zero-shot generalization. Using Proposition 1 from Borsa et al. [2018]: Given a GPI policy computed for a task based on a set of related tasks, the maximum difference between optimal policy for the task and the GPI policy for any state-action pair is upper bounded by two major terms: (a) minimum of the norm of the distance between task and any task, (b) the approximation error of SFs of the task. The first term error can be taken care of by choosing related tasks to be closely related to the target task with respect to a similarity distribution $\mathcal{D}(\cdot|\mathbf{w}')$. The second error can be taken care of by training on related tasks simultaneously, thereby reducing the approximation error.

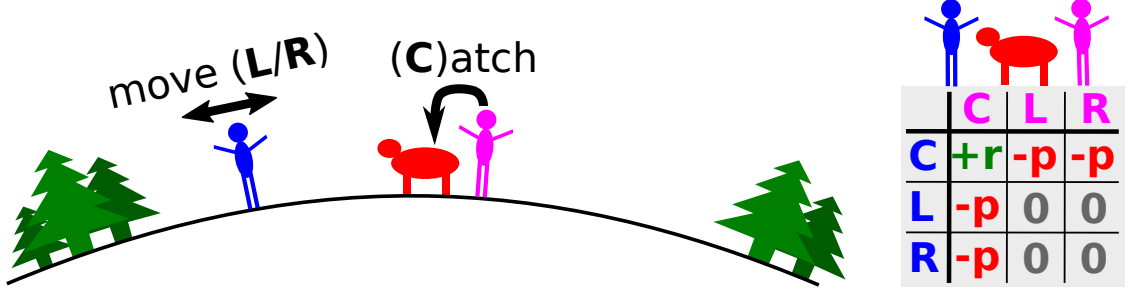


Figure 5.1: Simplified predator-prey environment (left), where two agents are only rewarded when they both stand next to prey (right).

5.4 Illustrative Example

Figure 5.1 sketches a simplified predator-prey task where two agents (blue and magenta) can both move left or right (L/R) and execute a special ‘catch’ (C) action when they both stand next to the stationary prey (red). The agents are collaboratively rewarded (shown on the right of Figure 5.1) $+r$ when they catch the prey together and punished $-p$ if they attempt it alone, both ending the episode. For large p , both VDN and QMIX can lead to relative overgeneralization [RO, Panait et al., 2006a] in the rewarded state s when agent 1’s utility $Q^1(s, u^1)$ of the catch action $u^1 = C$ drops below that of the movement actions $u^1 \in \{L, R\}$. At the beginning of training, when the value estimates are near zero and both agents explore random actions, we have:

$$Q^1(s, C) < Q^1(s, L/R) \Rightarrow r < p \left(\frac{1}{\pi^2(C|s)} - 2 \right) + c,$$

where c is a constant that depends mainly on future values. See Appendix B.1 for a formal derivation. The threshold at which p yields RO depends strongly on agent 2’s probability of choosing $u^2 = C$, i.e., $\pi^2(C|s)$. For uniform exploration, this criterion is fulfilled if $p > r$, but reinforcing other actions than $u^2 = C$ can lower this threshold significantly. However, if agent 2 chooses $u^2 = C$ for more than half of its actions, no amount of punishment can prevent learning of the correct greedy policy.

Figure 5.2 plots the entire task space w.r.t. p and r , marking the set of tasks solvable under uniform exploration (green area) and tasks exhibiting RO on average

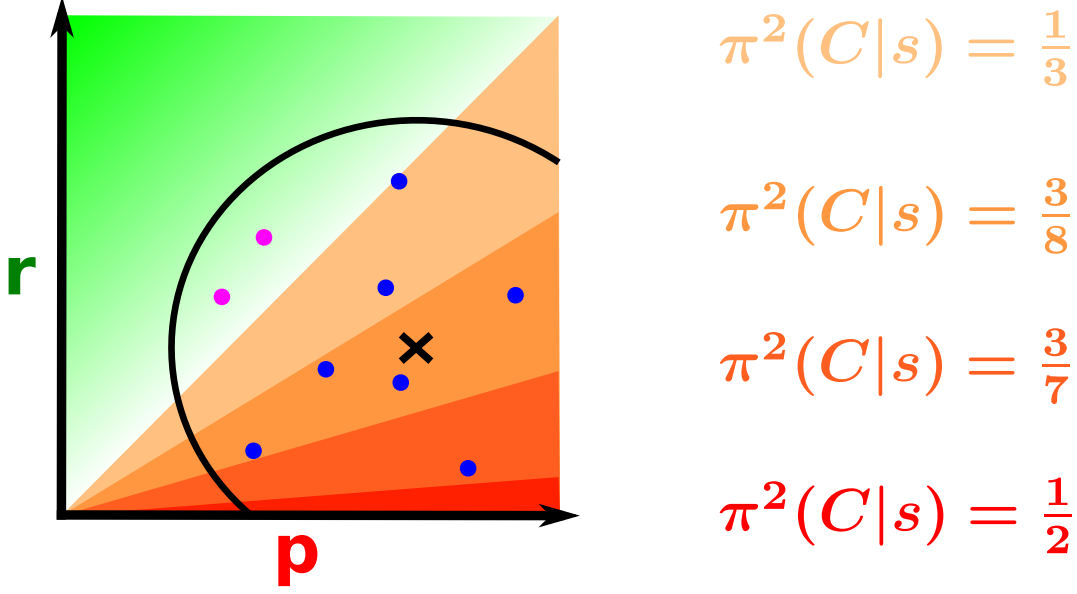


Figure 5.2: Task space of Figure 5.1. Tasks solvable under uniform exploration ($\pi^2(C|s) = \frac{1}{3}$) are green, shades of red represent tasks that on average exhibit RO for different $\pi^2(C|s)$.

for varying $\pi^2(C|s)$. MAUSFs uses a VDN decomposition of successor features to learn all tasks in the black circle simultaneously, which initially can only solve monotonic tasks in the green area. UneVEN changes this by evaluating random tasks (blue dots) and exploring those already solved (magenta dots). This increases the fraction of observed $u^2 = C$, and thus the set of solvable tasks, which eventually reaches the *target task* (cross). A visualization of this process is shown in Figure 5.3.

5.5 Multi-Agent Universal Successor Features

In this section, we introduce Multi-Agent Universal Successor Features (MAUSFs), extending single-agent USFs [Borsa et al., 2018] to multi-agent settings and show how we can learn generalized *decentralized* greedy policies for agents. The USFs based centralized joint action value function $Q_{tot}(\boldsymbol{\tau}, \mathbf{u}, \mathbf{z}, \mathbf{w})$ allows evaluation of joint policy $\boldsymbol{\pi}_{\mathbf{z}} = \langle \pi_{\mathbf{z}}^1, \dots, \pi_{\mathbf{z}}^n \rangle$ comprised of local agent policies $\pi_{\mathbf{z}}^a$ of the *same* task $\mathbf{z}$ on task $\mathbf{w}$. However, each agent a may execute a different policy $\pi_{\mathbf{z}^a}^a$ of different task $\mathbf{z}^a \in \mathcal{C}$, resulting in a combinatorial set of joint policies. Maximizing over all combinations $\bar{\mathbf{z}} \equiv \langle \mathbf{z}^1, \dots, \mathbf{z}^n \rangle \in \mathcal{C}^n$ should therefore enormously improve GPI.

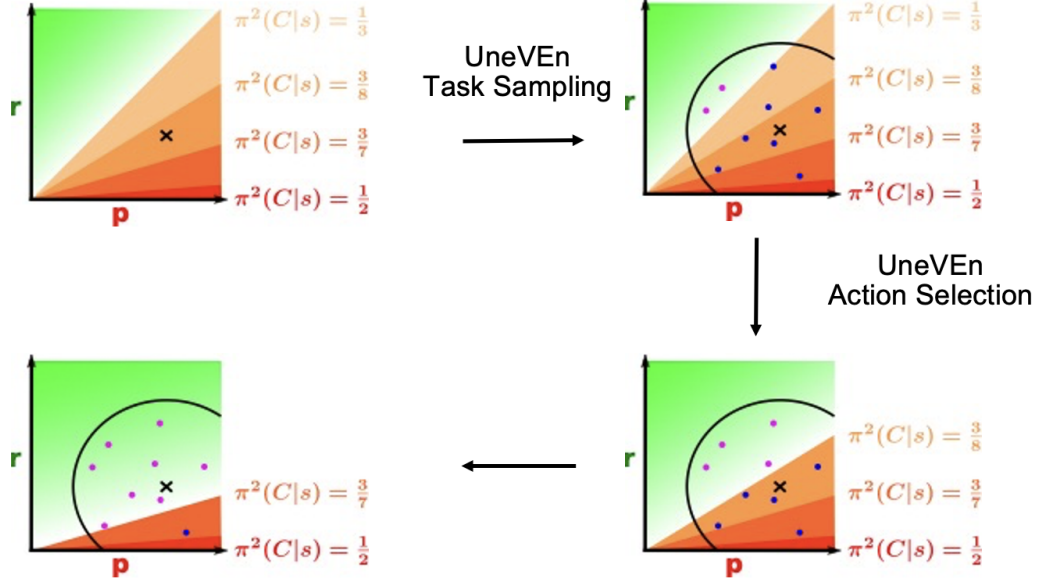


Figure 5.3: UneVEN works by evaluating random tasks (blue dots), and exploring already solved (magenta dots). This increases the fraction of observed $u^2 = C$, and thus the set of solvable tasks, which eventually reaches the *target task* (cross).

To enable this flexibility, we define the joint action-value function (Q_{tot}) of joint policy $\pi_{\bar{z}} = \{\pi_{z^a}^a\}_{z^a \in \mathcal{C}}$ evaluated on any task $\mathbf{w} \in \mathbb{R}^d$ as:

$$Q_{tot}(\boldsymbol{\tau}, \mathbf{u}, \bar{\mathbf{z}}, \mathbf{w}) = \boldsymbol{\psi}_{tot}(\boldsymbol{\tau}, \mathbf{u}, \bar{\mathbf{z}})^\top \mathbf{w},$$

where $\boldsymbol{\psi}_{tot}(\boldsymbol{\tau}, \mathbf{u}, \bar{\mathbf{z}})$ are the MAUSFs of $(\boldsymbol{\tau}, \mathbf{u})$ summarizing the joint dynamics of the environment under joint policy $\pi_{\bar{\mathbf{z}}}$. However, training centralized MAUSFs and using centralized GPI to maximize over a combinatorial space of $\bar{\mathbf{z}}$ becomes impractical when there are more than a handful of agents, since the joint action space ($\mathbf{u}$) and joint task space ($\mathcal{C}^n$) of the agents grows exponentially with the number of agents. To leverage CTDE and enable decentralized execution by agents, we therefore propose novel *agent-specific SFs* for each agent a following local policy $\pi_{z^a}^a$, which condition only on its own local action-observation history and task $\mathbf{z}^a$.

5.5.1 Decentralized Execution

We define local *utility* functions for each agent a as:

$$Q^a(\tau^a, u^a, \mathbf{z}^a, \mathbf{w}) = \boldsymbol{\psi}^a(\tau^a, u^a, \mathbf{z}^a; \theta)^\top \mathbf{w},$$

where $\boldsymbol{\psi}^a(\tau^a, u^a, \mathbf{z}^a; \theta)$ are the local agent-specific SFs induced by local policy $\pi_{\mathbf{z}^a}^a(u^a|\tau^a)$ of agent a sharing parameters θ . Intuitively, $Q^a(\tau^a, u^a, \mathbf{z}^a, \mathbf{w})$ is the utility function for agent a when local policy $\pi_{\mathbf{z}^a}^a(u^a|\tau^a)$ of task $\mathbf{z}^a$ is executed on task $\mathbf{w}$. We use VDN decomposition to represent MAUSFs $\boldsymbol{\psi}_{tot}$ as a sum of local agent-specific SFs for each agent a :

$$\begin{aligned} Q_{tot}(\boldsymbol{\tau}, \mathbf{u}, \bar{\mathbf{z}}, \mathbf{w}) &= \sum_{a=1}^n Q^a(\tau^a, u^a, \mathbf{z}^a, \mathbf{w}) \\ &= \underbrace{\sum_{a=1}^n \boldsymbol{\psi}^a(\tau^a, u^a, \mathbf{z}^a; \theta)^\top \mathbf{w}}_{\boldsymbol{\psi}_{tot}(\boldsymbol{\tau}, \mathbf{u}, \bar{\mathbf{z}}; \theta)} \end{aligned} \quad (5.4)$$

We can now learn local agent-specific SFs $\boldsymbol{\psi}^a$ for each agent a that can be instantly weighted with any task vector $\mathbf{w} \in \mathbb{R}^d$ to generate local utility functions Q^a , thereby allowing agents to use the GPI policy in a decentralized manner.

5.5.2 Decentralized Local GPI

Our novel agent-specific SFs allow each agent a to locally perform decentralized GPI by instant policy evaluation of a set $\mathcal{C}$ of local task policies $\{\pi_{\mathbf{z}^a}^a\}_{\mathbf{z}^a \in \mathcal{C}}$ on any unseen task $\mathbf{w}$ to compute a local GPI policy. Due to the linearity of the VDN decomposition, this is equivalent to maximization over all combinations of $\bar{\mathbf{z}} \equiv \langle \mathbf{z}^1, \dots, \mathbf{z}^n \rangle \in \mathcal{C} \times \dots \times \mathcal{C} \equiv \mathcal{C}^n$ as:

$$\begin{aligned} \pi'_w(\boldsymbol{\tau}) &\in \arg \max_{\mathbf{u} \in \mathcal{U}} \max_{\bar{\mathbf{z}} \in \mathcal{C}^n} Q_{tot}(\boldsymbol{\tau}, \mathbf{u}, \bar{\mathbf{z}}, \mathbf{w}) \\ &\in \left\{ \arg \max_{\mathbf{u}^a \in \mathcal{U}} \max_{\mathbf{z}^a \in \mathcal{C}} \boldsymbol{\psi}^a(\tau^a, u^a, \mathbf{z}^a; \theta)^\top \mathbf{w} \right\}_{a=1}^n. \end{aligned} \quad (5.5)$$

As all of the above relies on the linearity of the VDN decomposition, it cannot be directly applied to nonlinear mixing techniques like QMIX [Rashid et al., 2020b]. We empirically show that this limitation does not significantly affect performance for our method as UneVEN enables efficient monotonic approximations.

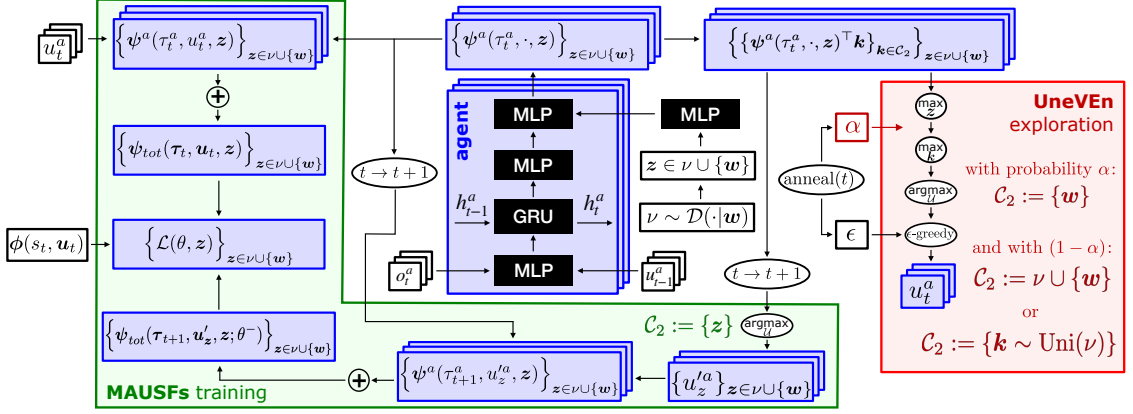


Figure 5.4: Schematic illustration of the MAUSFs training and UneVEn exploration with GPI policy.

5.5.3 Training

MAUSFs for task combination $\bar{z}$ are trained end-to-end by gradient descent on the loss:

$$\mathcal{L}(\theta, \bar{z}) = \mathbb{E}_{\sim \mathcal{B}} \left[\left\| \phi(s_t, \mathbf{u}_t) + \gamma \psi_{tot}(\tau_{t+1}, \mathbf{u}'_{\bar{z}}, \bar{z}; \theta^-) - \psi_{tot}(\tau_t, \mathbf{u}_t, \bar{z}; \theta) \right\|_2^2 \right], \quad (5.6)$$

where ϕ is the basis function which encodes potentially rewarded events (see Section 5.3.1) and the expectation is over a minibatch of samples $\{(s_t, \mathbf{u}_t, \tau_t)\}$ from the replay buffer $\mathcal{B}$ [Lin, 1992], θ^- denotes the parameters of a target network [Mnih et al., 2015] and joint actions $\mathbf{u}'_{\bar{z}} = \{u'_{z^a}\}_{a=1}^n$ are selected individually by each agent network using the current parameters θ [called Double Q -learning, van Hasselt et al., 2016]:

$$u'_{z^a} = \arg \max_{u \in \mathcal{U}} \psi^a(\tau_{t+1}^a, u, \mathbf{z}^a; \theta)^\top \mathbf{z}^a.$$

Each agent learns therefore local agent-specific SFs $\psi^a(\tau^a, u^a, \mathbf{z}; \theta)$ by gradient descent on $\mathcal{L}(\theta, \bar{z})$ for all $\mathbf{z} \in \mathcal{C} \equiv \nu \cup \{w\}$, where $\nu \sim \mathcal{D}(\cdot | w)$ is drawn from a distance measure around target task w . The green region of Figure 5.4 shows a CTDE based architecture to train MAUSFs for a given target task w . A detailed algorithm is presented in Appendix B.2.

5.6 UneVEn

In this section, we present UneVEn (red region of Figure 5.4), which leverages MAUSFs and decentralized GPI to help overcome relative overgeneralization on the target task $\mathbf{w}$. At the beginning of every exploration episode, we sample a set of related tasks

$$\nu = \{\mathbf{z} \sim \mathcal{D}(\cdot|\mathbf{w})\},$$

containing potentially simpler reward functions, from a distribution $\mathcal{D}$ around the target task. The basic idea is that *some* of these related tasks can be efficiently learned with a factored value function. These tasks are therefore solved early, and exploration concentrates on the state-action pairs that are useful to them. If other tasks close to those already solved have the same optimal actions, this implicit weighting allows us to solve them too [shown by Rashid et al., 2020a]. Furthermore, tasks closer to $\mathbf{w}$ are sampled more frequently, which biases the process to eventually overcome relative overgeneralization on the target task itself.

Our method assumes that the basis functions ϕ and the reward-weights $\mathbf{w}$ of the target task are known, but Barreto et al. [2020] show that both can be learned using multi-task regression. A discussion of potential limitations of such approximation is presented in Section 5.10 at the end of this chapter. Many choices for $\mathcal{D}$ are possible, but in the following we sample related tasks using a normal distribution centered around the target task $\mathbf{w} \in \mathbb{R}^d$ with a small variance σ as $\mathcal{D} = \mathcal{N}(\mathbf{w}, \sigma \mathbf{I}_d)$. This samples similar tasks closer to $\mathbf{w}$ more frequently. As long as σ is large enough to cover tasks that do not induce RO (see Figure 5.2), our method works well and, therefore, does not rely on any domain knowledge. The resulting task vectors weight the basis functions ϕ differently and represent different reward functions. For example, the predator-prey task discussed in the introduction encodes penalties associated with miscoordination in the task weight vector $\mathbf{w}$. Therefore, sampling related tasks using the normal distribution results in related

tasks with different weightings of the basis function compared to the target task. In particular, the varied reward functions can make these tasks much easier, but also harder, to solve with monotonic value functions. The consequences of sampling harder tasks on learning are discussed with the below action-selection schemes. Intuitively, UneVEn relies on the existence of tasks related to the target task such that the reward functions of the related tasks implies a simpler version or subset of the target task, and hence easier to solve. Solving such simpler tasks enables UneVEn to select actions which might also be optimal for the target task. However, UneVEn may fail if there are *no* sampled related tasks such that their optimal actions' are also optimal for the target task.

5.6.1 Action-Selection Schemes

UneVEn uses two novel schemes to enable action selection based on related tasks. To emphasize the importance of the target task, we define a probability α of selecting actions based on the target task. Therefore, with probability $1 - \alpha$, the action is selected based on the related task. Similar to other exploration schemes, α is annealed from 0.3 to 1.0 in our experiments over a fixed number of steps at the beginning of training. Once this exploration stage is finished (i.e., $\alpha = 1$), actions are always taken based on the target task's joint action-value function. Each action-selection scheme employs a local decentralized GPI policy, which maximizes over a set of policies π_z based on $\mathbf{z} \in \mathcal{C}_1$ (also referred to as the *evaluation* set) to estimate the Q -values of another set of tasks $\mathbf{k} \in \mathcal{C}_2$ (also referred to as the *target* set) using:

$$\mathbf{u}_t = \left\{ u_t^a = \arg \max_{u \in \mathcal{U}} \max_{\mathbf{k} \in \mathcal{C}_2} \max_{\mathbf{z} \in \mathcal{C}_1} \overbrace{\psi^a(\tau_t^a, u, \mathbf{z}; \theta)^\top \mathbf{k}}^{Q^a(\tau_t^a, u, \mathbf{z}, \mathbf{k})} \right\}_{a \in \mathcal{A}}. \quad (5.7)$$

Here $\mathcal{C}_1 = \nu \cup \{\mathbf{w}\}$ is the set of target and related tasks that induce the policies that are evaluated (dot-product) on the set of tasks $\mathcal{C}_2$, which varies with different action-selection schemes. The red box in Figure 5.4 illustrates UneVEn exploration. For example, Q -learning always picks actions based on the target task, i.e., the target set $\mathcal{C}_2 = \{\mathbf{w}\}$. However, this scheme does not favor important joint actions. We call

this default action-selection scheme **target GPI** and execute it with probability α . We now propose two novel action-selection schemes based on related tasks with probability $1 - \alpha$, thereby implicitly weighting joint actions during learning.

Uniform GPI

At the beginning of each episode, this action-selection scheme uniformly picks *one* related task, i.e., the target set $\mathcal{C}_2 = \{\mathbf{k} \sim \text{Uniform}(\nu)\}$, and selects actions based on that task using the GPI policy throughout the episode. This uniform task selection explores the learned policies of all related tasks in $\mathcal{D}$. This works well in practice as there are often enough simpler tasks to induce the required bias over important joint actions. However, if the sampled related task is harder than the target task, the action selection based on these harder tasks might hurt learning on the target task and lead to higher variance during training.

Greedy GPI

At every time-step t , this action-selection scheme picks the task $\mathbf{k} \in \nu \cup \{\mathbf{w}\}$ that gives the highest Q -value amongst the related and target tasks, i.e., the target set becomes $\mathcal{C}_2 = \nu \cup \{\mathbf{w}\}$. Due to the greedy nature of this action-selection scheme, exploration is biased towards solved tasks, as those have larger values. We are thus exploring the solutions of tasks that are both solvable and similar to the target task $\mathbf{w}$, which should, at least in some states, lead to the same *optimal joint actions* as $\mathbf{w}$.

NO-GPI

To demonstrate the influence of GPI on the above schemes, we also investigate ablations, where we define the evaluation set $\mathcal{C}_1 = \{\mathbf{k}\}$ to only contain the currently estimated task $\mathbf{k}$, i.e., using:

$$\mathbf{u}_t = \{u_t^a = \arg \max_{u \in \mathcal{U}} \max_{\mathbf{k} \in \mathcal{C}_2} \psi^a(\tau_t^a, u, \mathbf{k}; \theta)^\top \mathbf{k}\}_{a \in \mathcal{A}}$$

for action selection.

5.7 Related Work on Multi-Agent RL

In this chapter, we focus on cooperative MARL settings where it is often necessary to learn *fully* decentralized policies for agents due to partial observability and communication constraints.

5.7.1 Exploration

There are many techniques for exploration in model-free single-agent RL, based on intrinsic novelty reward [Bellemare et al., 2016, Tang et al., 2017], predictability [Pathak et al., 2017], pure curiosity [Burda et al., 2019] or Bayesian posteriors [Osband et al., 2016b, Gal et al., 2017, Fortunato et al., 2018, O’Donoghue et al., 2018]. In the context of multi-agent RL, Böhmer et al. [2019] discuss the influence of unreliable intrinsic reward and Wang et al. [2020c] quantify the influence that agents have on each other’s return. Zheng and Yue [2018] propose to coordinate exploration between agents by shared latent variables, whereas Jaques et al. [2018] investigate the social motivations of competitive agents. However, these techniques aim to visit as much of the state-action space as possible, which exacerbates the relative overgeneralization pathology. Approaches that use state-action space abstraction can speed up exploration; these include those which can automatically learn the abstractions [e.g., Mahajan and Tulabandhula, 2017a,b] and those which require prior knowledge [e.g., Roderick et al., 2018]. However they are difficult to scale for multi-agent scenarios. In contrast, UneVEN explores similar *tasks*. This guides exploration to states and actions that prove *useful*, which restricts the explored space and overcomes relative overgeneralization. To the best of our knowledge, the only other work that explores the task space is Leibo et al. [2019]: they use the evolution of competing agents as an auto-curriculum of harder and harder tasks. Collaborative agents cannot compete against each other, though, and their approach, therefore, is not suitable for learning cooperation.

5.8 Related Work on Successor Features

Most of the work on SFs has been focused on single-agent settings [Dayan, 1993, Kulkarni et al., 2016b, Lehnert et al., 2017, Barreto et al., 2017, 2018, Borsa et al., 2018, Lee et al., 2019a, Hansen et al., 2019] for transfer learning and zero-shot generalization across tasks with different reward functions. In order for an agent to perform well in tasks with different reward functions, we need a mechanism to successfully transfer knowledge between tasks. Successor Representation [Dayan, 1993] decomposes a value function into a representation of the policy termed as successor features and reward weight vectors, thereby decoupling the policy dynamics and the rewards. This enables SFs to perform Generalized Policy Improvement (GPI) [Barreto et al., 2017], which allows instant policy evaluation of a set of tasks (*evaluation set*) onto an unseen task. The computed GPI policy performs better than any of the individual policies in the evaluation set before any learning has taken place on the unseen task. Kulkarni et al. [2016b] proposes deep successor RL, which generalizes the Successor Representation (SR) within an end-to-end deep reinforcement learning framework. Borsa et al. [2018] and Ma et al. [2020] propose Universal Successor Features (USFs), which learn single-agent SFs for different tasks using UVFs and enable efficient use of GPI.

The above mentioned approaches assume that the basis functions are known, or learn them via multi-task regression [Barreto et al., 2017]. However, approximating basis functions requires defining a set of base tasks, which restricts the learned features to a subset of representable reward functions. Hansen et al. [2019] proposed Variational Intrinsic SFs which leverages the SFs framework to generalize beyond the finite set of behaviors being explicitly trained using a mutual information objective. They propose to learn basis functions such that they can represent a diverse set of policies (skills) in an unsupervised manner using skill discovery. Gupta et al. [2019] use single-agent SFs in a multi-agent setting to estimate the probability of events, but they only consider transition independent MARL [Becker et al., 2004,

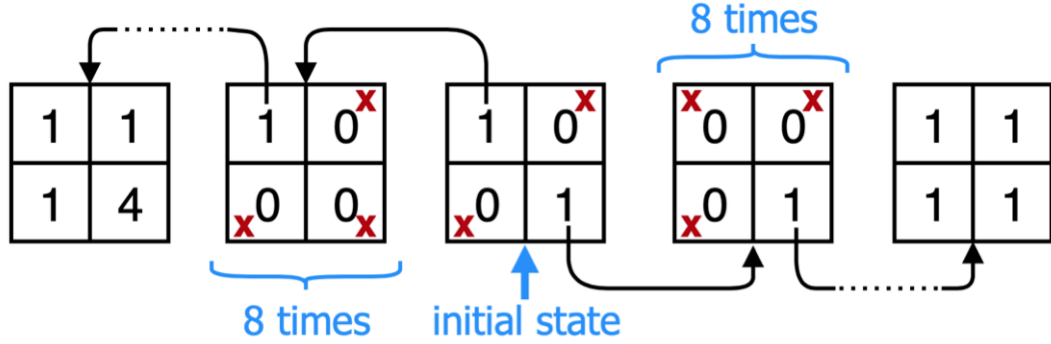


Figure 5.5: m -step matrix game from Mahajan et al. [2019] for $m = 10$. The red cross means that selecting that joint action will lead to termination of the episode.

Gupta et al., 2018b]. Our work is significantly different as we extend single-agent SFs to multi-agent settings (MAUSFs) and propose novel agent-specific SFs with learning based on a CTDE based training regime.

5.9 Experiments

In this section, we evaluate UneVEn on a variety of complex domains. For evaluation, all experiments are carried out with six random seeds and results are shown with $\pm$ standard error across seeds. We compare our method against a number of SOTA value-based MARL approaches: IQL [Tan, 1993], VDN [Sunehag et al., 2018], QMIX [Rashid et al., 2020b], MAVEN [Mahajan et al., 2019], WQMIX [Rashid et al., 2020a], QTRAN [Son et al., 2019], and QPLEX [Wang et al., 2020a].

5.9.1 Domain 1 : m -Step Matrix Game

We first evaluate UneVEn on the m -step matrix game proposed by Mahajan et al. [2019]. Figure 5.5 shows the m -step matrix game for $m = 10$, where there are $m - 2$ intermediate steps, and selecting a joint-action with zero reward leads to termination of the episode. This task is difficult to solve using simple ϵ -greedy exploration policies as committed exploration is required to achieve the optimal return. In this game, one initial joint action can reach a return of up to $m + 3$, whereas another only allows for m . This challenges monotonic value functions, as

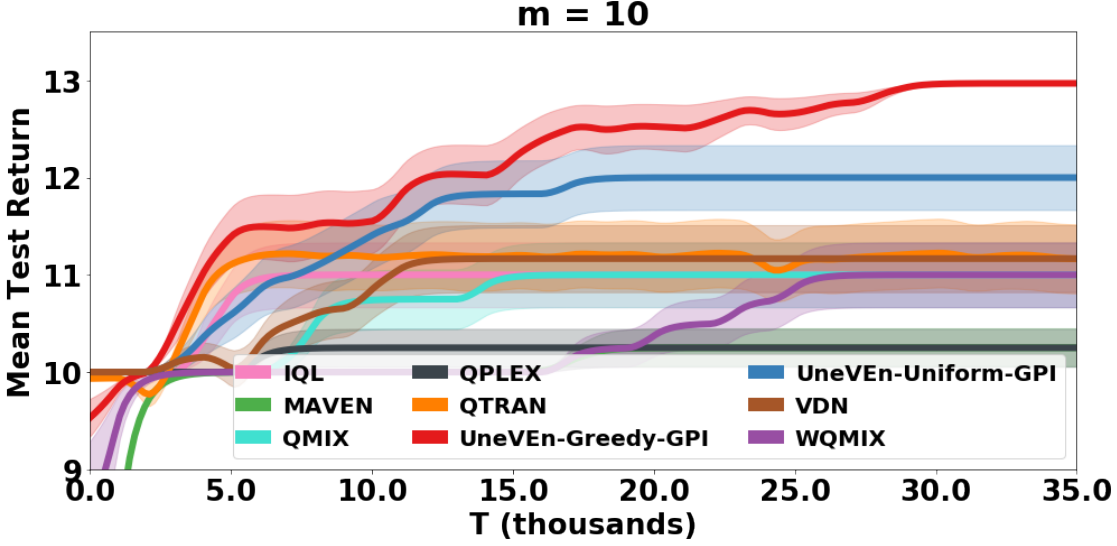


Figure 5.6: SOTA comparison on the m -step matrix game ($m = 10$).

the target task exhibits RO. The optimal policy of this task is to take the top-left joint action first, then stick to the same joint action for $m - 2$ steps and finally take bottom-right joint action leading to a return of $+13$.

Figure 5.6 shows the results of all methods on this task for $m = 10$ after training for $35k$ steps. UneVEn with greedy (UneVEn-Greedy-GPI) action selection scheme converges to an optimal return and both greedy and uniform (UneVEn-Uniform-GPI) schemes outperforms all other methods, which suffer from poor ϵ -greedy exploration and often learn to take the suboptimal action in the beginning. We sample related tasks based on normal distribution centered around the target task $\mathbf{w}$, where $\mathbf{w}$ captures the values from the matrices defined in Figure 5.5. This could result in sampling higher reward tasks where the agent might get higher returns, or does not get terminated for selecting the zero reward action in the target task, and hence able to simplify exploration. Due to RO, it becomes difficult to change the policy later, leading to suboptimal returns and only rarely converging to optimal solutions.

5.9.2 Domain 2 : Cooperative Predator-Prey

We next evaluate UneVEn on challenging cooperative predator-prey tasks similar to one proposed by Son et al. [2019], but significantly more complex in terms of

the coordination required amongst agents. We use a complex partially observable predator-prey (PP) task involving eight agents (predators) and three preys that is designed to test coordination between agents, as each prey needs to be captured by at least three surrounding agents with a simultaneous *capture* action. If only one or two surrounding agents attempt to capture the prey, a negative reward of magnitude p is given. Successful capture yields a positive reward of $+1$. Each agent can take 6 actions i.e. move in one of the 4 directions (Up, Left, Down, Right), remain still (no-op), or try to catch (capture) any adjacent prey. The prey moves around in the grid with a probability of 0.7 and remains still at its position with probability 0.3. Impossible actions for both agents and prey are marked unavailable, for eg. moving into an occupied cell or trying to take a capture action with no adjacent prey.

If either a single or a pair of agents take a capture action on an adjacent prey, a negative reward of magnitude p is given. If three or more agents take the capture action on an adjacent prey, it leads to a successful capture of that prey and yield a positive reward of $+1$. The maximum possible reward for capturing all prey is therefore $+3$. Each agent observes a 5×5 grid centered around its position which contains information showing other agents and prey relative to its position. An episode ends if all prey have been captured or after 800 time steps. This task is challenging for two reasons. First, depending on the magnitude of p , exploration is difficult as even if a single agent miscoordinates, the penalty is given, and therefore, any steps toward successful coordination are penalized. Second, the agents must be able to differentiate between the values of successful and unsuccessful collaborative actions, which monotonic value functions fail to do on tasks exhibiting RO.

Proposition. For, the predator-prey game defined above, the optimal joint action reward function for any group of $2 \leq k \leq n$ predator agents surrounding a prey is *non-monotonic* [as defined by Mahajan et al., 2019] iff $p > 0$.

Proof. Without loss of generality, we assume a single prey surrounded by three agents (A_1, A_2, A_3) in the environment. The joint reward function for this group

A1-Capture			A1-Other		
	A2-Capture	A2-Other		A2-Capture	A2-Other
A3-Capture	+1	$-p$	A3-Capture	$-p$	$-p$
A3-Other	$-p$	$-p$	A3-Other	$-p$	0

Table 5.1: Joint-Reward function of three agents surrounding a prey. The first table indicates joint-rewards when Agent 1 takes capture action and second table indicates joint-rewards when Agent 1 takes any other action. Notice that there are numerous joint actions leading to penalty p .

of three agents is defined in Table 5.1. For the case $p > 0$ the proposition can be easily verified using the definition of non-monotonicity [Mahajan et al., 2019]. For any $3 \leq k \leq n$ agents attempting to catch a prey in state s , we fix the actions of any $k - 3$ agents to be “other” indicating either of up, down, left, right, and noop actions and represent it with $\mathbf{u}^{k-3}$. Next we consider the rewards r for two cases:

- If we fix the action of any *two* of the remaining three agents as “other” represented as $\mathbf{u}^2$, the action of the remaining agent becomes:

$$u^1 = \arg \max_{u \in \mathcal{U}} r(s, \langle u, \mathbf{u}^2, \mathbf{u}^{k-3} \rangle) = \text{“other”}.$$

- If we fix the $\mathbf{u}^2$ to be “capture”, we have :

$$u^1 = \arg \max_{u \in \mathcal{U}} r(s, \langle u, \mathbf{u}^2, \mathbf{u}^{k-3} \rangle) = \text{“capture”}.$$

Thus the best action for agent A_1 in state s depends on the actions taken by the other agents and the rewards $R(s)$ are non-monotonic. Finally for the equivalence, we note that for the case $p = 0$, a default action of “capture” is always optimal for any group of k predators surrounding the prey. Thus the rewards are monotonic as the best action for any agent is independent of the rest. $\square$

Simpler PP Tasks

We first demonstrate that both VDN and QMIX with monotonic value functions can learn on target tasks with simpler reward functions. To generate a simpler task, we remove the penalty associated with miscoordination, i.e., $p = 0$, thereby

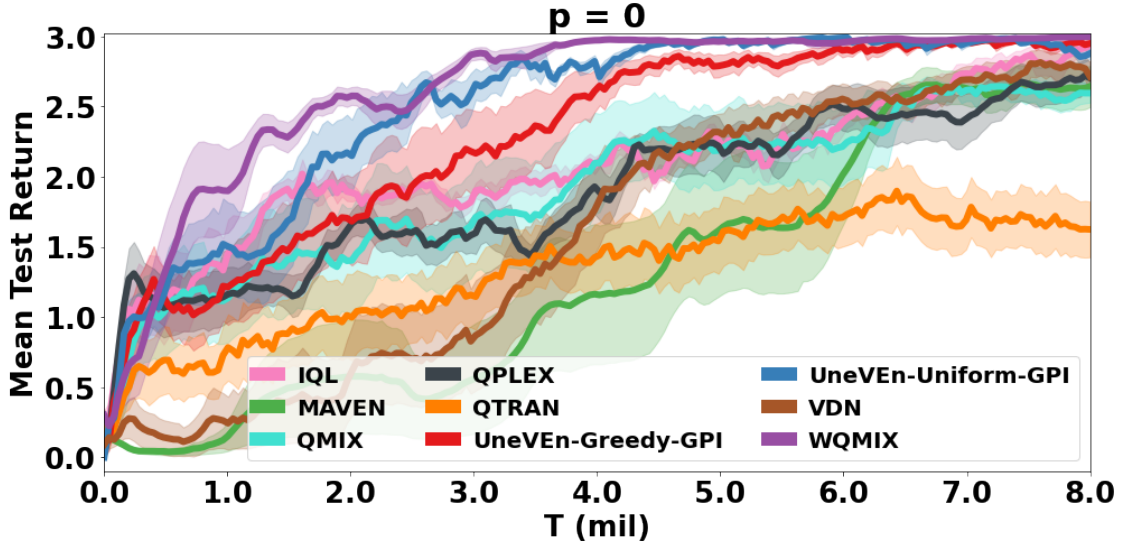


Figure 5.7: Baseline predator-prey results without RO ($p = 0$).

making the target task free from RO. Figure 5.7 shows that both QMIX and VDN can solve this task as there is no miscoordination penalty and the monotonic value function can learn to efficiently represent the optimal joint action values. Other SOTA value-based approaches (MAVEN, WQMIX, and QPLEX) and UneVEN with both uniform (UneVEN-Uniform-GPI) and greedy (UneVEN-Greedy-GPI) action-selection schemes can solve this target task as well.

Harder PP Tasks

We now raise the chance that RO occurs in the target task by increasing the magnitude of the penalty associated with each miscoordination, i.e.,

$$p \in \{0.004, 0.008, 0.012, 0.016\}.$$

For a smaller penalty of $p = 0.004$, Figure 5.8 (top left) shows that VDN can still solve the task, further suggesting that simpler reward related tasks (with lower penalties) can be solved with monotonic approximations. However, both QMIX and VDN fail to learn on three other higher penalty target tasks due to their monotonic constraints, which hinder the accurate learning of the joint action value functions. Intuitively, when uncoordinated joint actions are much more likely

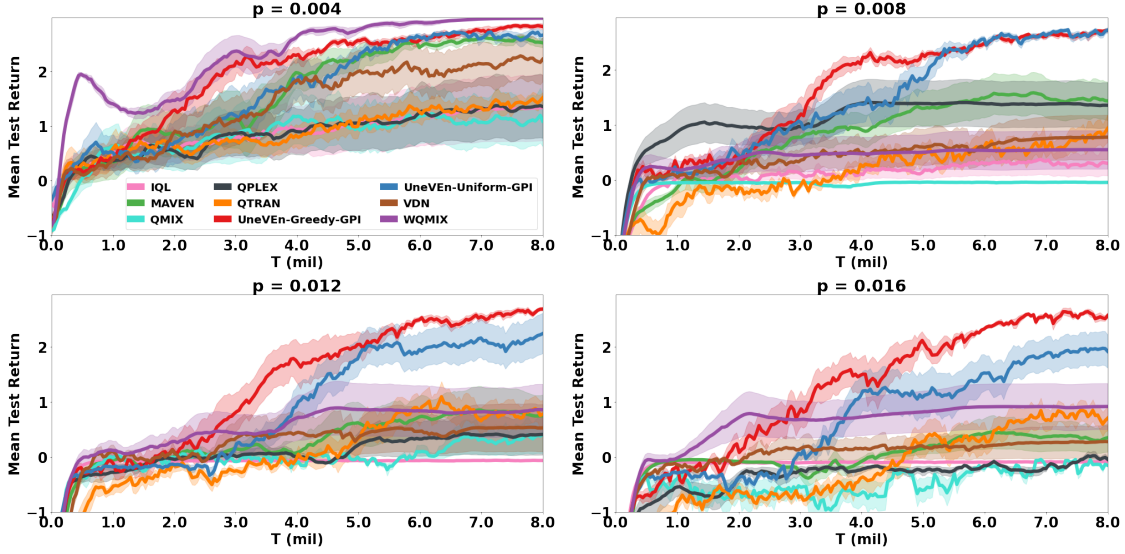


Figure 5.8: Comparison between UneVEN and SOTA MARL baselines with $p \in \{0.004, 0.008, 0.012, 0.016\}$.

than coordinated ones, the penalty term can dominate the average value estimated by each agent’s utility. This makes it difficult to learn an accurate monotonic approximation that selects the optimal joint actions.

Interestingly, other SOTA value-based approaches that aim to address the monotonicity restriction of QMIX and VDN such as MAVEN, QTRAN, WQMIX, and QPLEX also fail to learn on higher penalty tasks. WQMIX solves the task when $p = 0.004$, but fails on the other three higher penalty target tasks. Although WQMIX uses an explicit weighting mechanism to bias learning towards the target task’s optimal joint actions, it must identify these actions by learning an unrestricted value function first. An ϵ -greedy exploration based on the target task takes a long time to learn such a value function, which is visible in the large standard error for $p \in \{0.008, 0.012, 0.016\}$ in Figure 5.8. By contrast, both UneVEN-Uniform-GPI and UneVEN-Greedy-GPI can approximate value functions of target tasks exhibiting severe RO more accurately and solve the task for all values of p . As expected, the variance of UneVEN-Uniform-GPI is high on higher penalty target tasks (for e.g., $p = 0.016$) as exploration suffers from action selection based on harder related tasks. UneVEN-Greedy-GPI does not suffer from this problem. UneVEN might

struggle to approximate the value functions of target tasks when the penalty is high, if the variance of the sampled related tasks is too low, making even simpler sampled tasks still too difficult to solve. Therefore, the variance of the sampled related task may play a key role in performance. However, for all our experiments, variance $\sigma \in \{0.1, 0.2\}$ was enough to learn. Videos of learnt policies are available at <https://sites.google.com/view/uneven-marl/>.

We emphasize that while having an unrestricted joint value function (such as WQMIX, QPLEX and QTRAN) may allow to overcome RO, these algorithms are not guaranteed to do so in any reasonable amount of time. For example, Rashid et al. [2020a] note in Sec. 6.2.2 that training such a joint function can require much longer ϵ -greedy exploration schedules. The exact optimization of QTRAN is computationally intractable and a lot of recent work has shown the unstable performance of the corresponding approximate versions. Similarly QPLEX has shown to perform poorly on hard SMAC maps in our results and in Rashid et al. [2020a]. On the other hand, MAVEN learns a diverse ensemble of monotonic action-value functions, but does not concentrate on joint actions that would overcome RO if explored more.

Ablation on Action Selection Schemes

Figure 5.9 shows ablation results for higher penalty tasks, i.e., $p = \{0.012, 0.016\}$. To contrast the effect of UneVEN on exploration, we compare our two novel action-selection schemes to UneVEN-Target-GPI, which only selects the greedy actions of the target task. The results clearly show that UneVEN-Target-GPI fails to solve the higher penalty RO tasks as the employed monotonic joint value function of the target task fails to accurately represent the values of different joint actions. This demonstrates the critical role of UneVEN and its action-selection schemes.

Ablation on GPI

Next we evaluate the effect of GPI by comparing against UneVEN with MAUSFs without using the GPI policy, i.e., setting the evaluation set $\mathcal{C}_1 = \{\mathbf{k}\}$ in Equation 5.7.

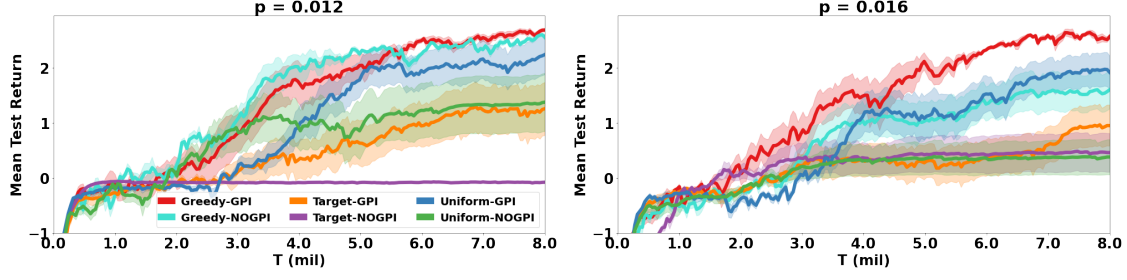


Figure 5.9: Ablation results. Comparison between different action selection schemes of UneVEn for $p \in \{0.012, 0.016\}$.

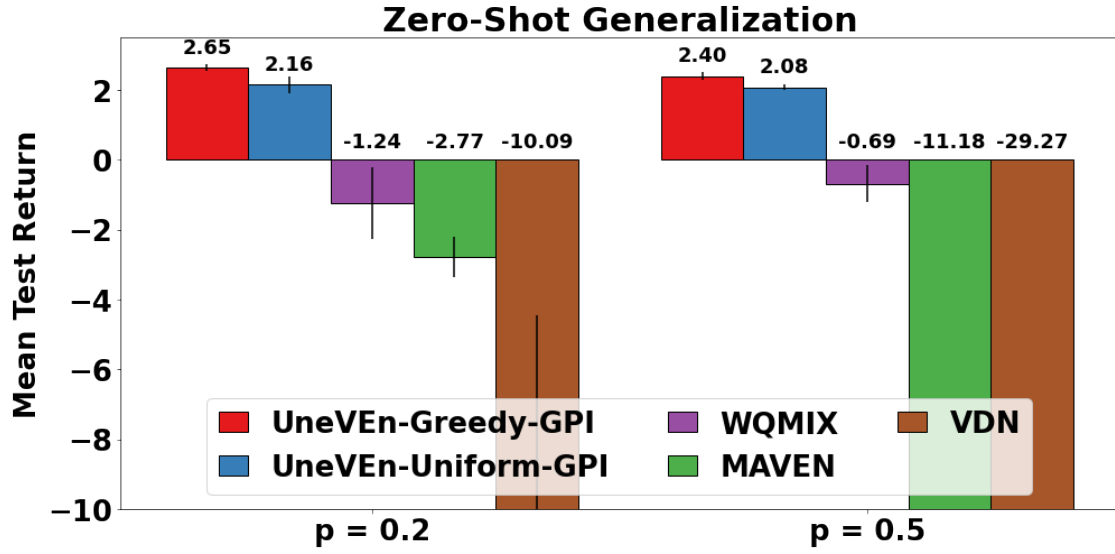


Figure 5.10: Zero-shot generalization comparison between UneVEn and SOTA MARL baselines; training on $p = 0.004$, testing on $p \in \{0.2, 0.5\}$.

First, UneVEn using a NOGPI policy with both uniform (Uniform-NOGPI) and greedy (Greedy-NOGPI) action selection outperforms Target-NOGPI, further strengthening the claim that UneVEn with its novel action-selection scheme enables efficient exploration and bias towards optimal joint actions. In addition, Figure 5.9 shows that for each corresponding action-selection scheme (uniform, greedy, and target), using a GPI policy (*-GPI) is consistently favourable as it performs either similarly to the NOGPI policy (*-NOGPI) or much better. GPI appears to improve zero-shot generalization of MAUSFs across tasks, which in turn enables good action selection for related tasks during UneVEn exploration.

Zero-Shot Generalization

Lastly, we evaluate this zero-shot generalization for all methods to check if the learnt policies are useful for unseen high penalty test tasks. We train all methods for 8 million environmental steps on a task with $p = 0.004$, and test 60 rollouts of the resulting policies of all methods that are able to solve the training task, i.e., UneVEn-Greedy-GPI, UneVEn-Uniform-GPI, VDN, MAVEN, and WQMIX, on tasks with $p \in \{0.2, 0.5\}$. For policies trained with UneVEn-Greedy-GPI and UneVEn-Uniform-GPI, we use the NOGPI policy for the zero-shot testing, i.e., $\mathcal{C}_1 = \mathcal{C}_2 = \{\mathbf{w}\}$. Figure 5.10 shows that UneVEn with both uniform and greedy schemes exhibits great zero-shot generalization and solves both test tasks even with very high penalties. As MAUSFs learn the reward’s basis functions, rather than the reward itself, zero-shot generalization to larger penalties follows naturally. Furthermore, using UneVEn exploration allows the agents to collect enough diverse behaviour to come up with a near optimal policy for the test tasks. On the other hand, the learnt policies for all other methods that solve the target task with $p = 0.004$ are ineffective in these higher penalty RO tasks, as they do not learn to avoid unsuccessful capture attempts. See Appendix B.3 for details about the implementations and Appendix B.4 for additional ablation experiments.

5.9.3 Domain 3 : Starcraft Multi-Agent Challenge (SMAC)

We now evaluate UneVEn on challenging cooperative StarCraft II (SC2) maps from the popular SMAC benchmark [Samvelyan et al., 2019]. Each unit is treated as an agent, which is able to move or to attack an enemy that is within its attack range. Each unit is treated as an agent, which is able to move or to attack an enemy that is within its attack range. Each agent’s observation are the positions of the allies and enemy units that are within its sight range but they cannot see the health of enemies. The state information, only available during centralized training, includes the positions and health of all allied and enemy units, as well as the last actions taken by the allied units. The environment is carefully designed requiring learning

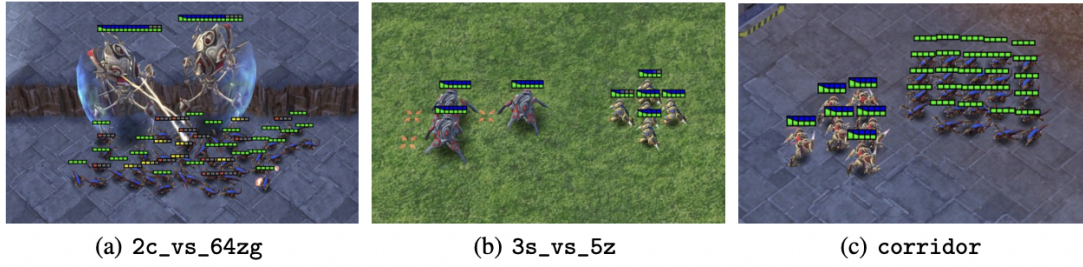


Figure 5.11: Representative images of a few maps from the SMAC environment taken from Samvelyan et al. [2019].

rich cooperative behaviours under partial observability, e.g., focus fire i.e. ordering units to jointly attack and kill enemy units; kiting i.e. making enemy units chase while maintaining enough distance so that little or no damage is incurred. Figure 5.11 showcases a few representative images from the SMAC environment.

Discussion on Reward Functions in SMAC

We first discuss different reward functions in the SMAC benchmark [Samvelyan et al., 2019]. The default SMAC reward function depends on three major components: (1) **delta-enemy**: accumulates difference in health and shield of all *enemy* units between last time step and current time step, (2) **delta-ally**: accumulates difference in health and shield of all *ally* units between last time step and current time step scaled by **reward-negative-scale**, (3) **delta-deaths**: defined below.

Original Reward Function (Positive Only Rewards)

The original reward function in the SMAC paper [Samvelyan et al., 2019] uses **reward-only-positive = True**, where **delta-deaths** is defined as positive reward of **reward-death-value** for every enemy unit killed. The final reward is:

$$\text{abs}(\text{delta-enemy} + \text{delta-deaths}).$$

Notice that some of the units have shield regeneration capabilities and therefore, **delta-enemy** might contain negative values as current time step health of enemy unit might be higher than last time step. Therefore, to enforce positive rewards, **abs** function is used.

Negative Reward Function

Contrary to the original reward function which only allows positive rewards, the environment also supports reward functions which allows negative rewards. For `reward-only-positive = False`, `delta-deaths` is defined as positive reward of `reward-death-value` for every enemy unit killed, and penalizes

$$\text{reward-negative-scale} * \text{reward-death-value}$$

for every ally unit killed. The final reward is simply: `delta-enemy + delta-deaths - delta-ally`. A few implications of this reward functions are as follows:

- The `reward-negative-scale` (interchangeably denoted by p) measures the relative importance of lives of ally units compared to enemy units.
- For `reward-negative-scale = 1.0`, both enemy and ally units lives are equally valued.
- For `reward-negative-scale = 0.5`, ally units are only valued half of enemy units, and for `reward-negative-scale = 0.0`, ally units lives are not valued at all.
- However, `reward-only-positive = False` with `reward-negative-scale = 0.0` is NOT the same as setting `reward-only-positive = True` as the latter uses an `abs` function.

To summarize, `reward-only-positive` decides whether there is an additional penalty for health reduction and death of ally units, and `reward-negative-scale` determines the relative importance of lives of ally units when `reward-only-positive = False`. Figure 5.12 and Figure 5.13 shows that for most of the maps, there is not a big performance difference for VDN and QMIX between different reward functions (original, $p = 0.0$ and $p = 0.5$). However, for some maps using `reward-only-positive = False` with either $p = 0.0$ and $p = 0.5$ improves

performance over the original reward function. We hypothesize that the use of `abs` in the original reward function can detriment the learning of agent as it might get positive absolute reward to increase the health of enemy units.

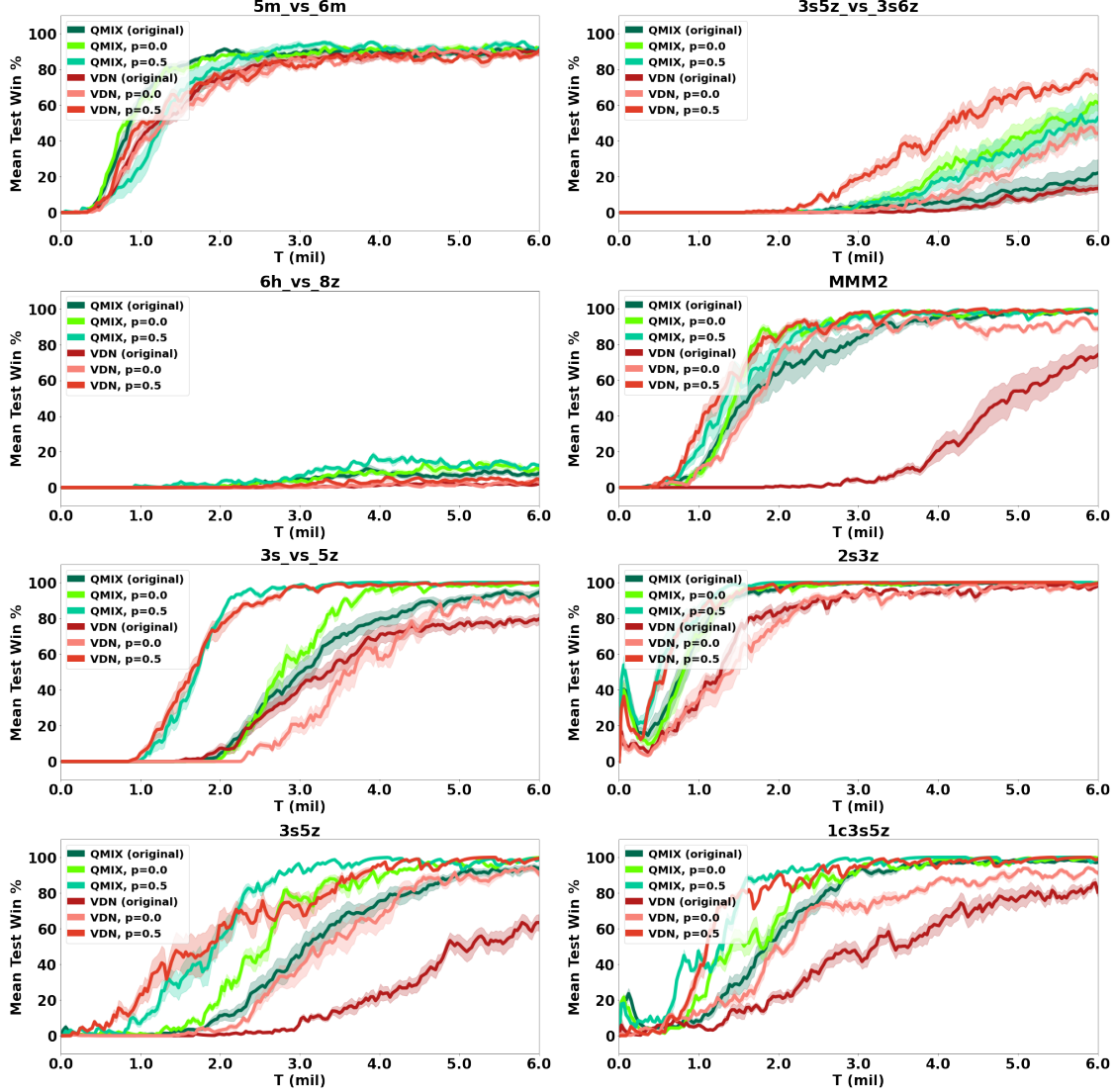


Figure 5.12: Comparison between VDN and QMIX baselines with original positive reward function and modified reward function with $p \in \{0.0, 0.5\}$ on SMAC maps.

SMAC with Negative Reward and $p = 1.0$

Our method aims to overcome relative overgeneralization (RO) which happens very often in practice with cooperative games [Wei and Luke, 2016]. However, the default reward function in SMAC does not suffer from RO as it has been

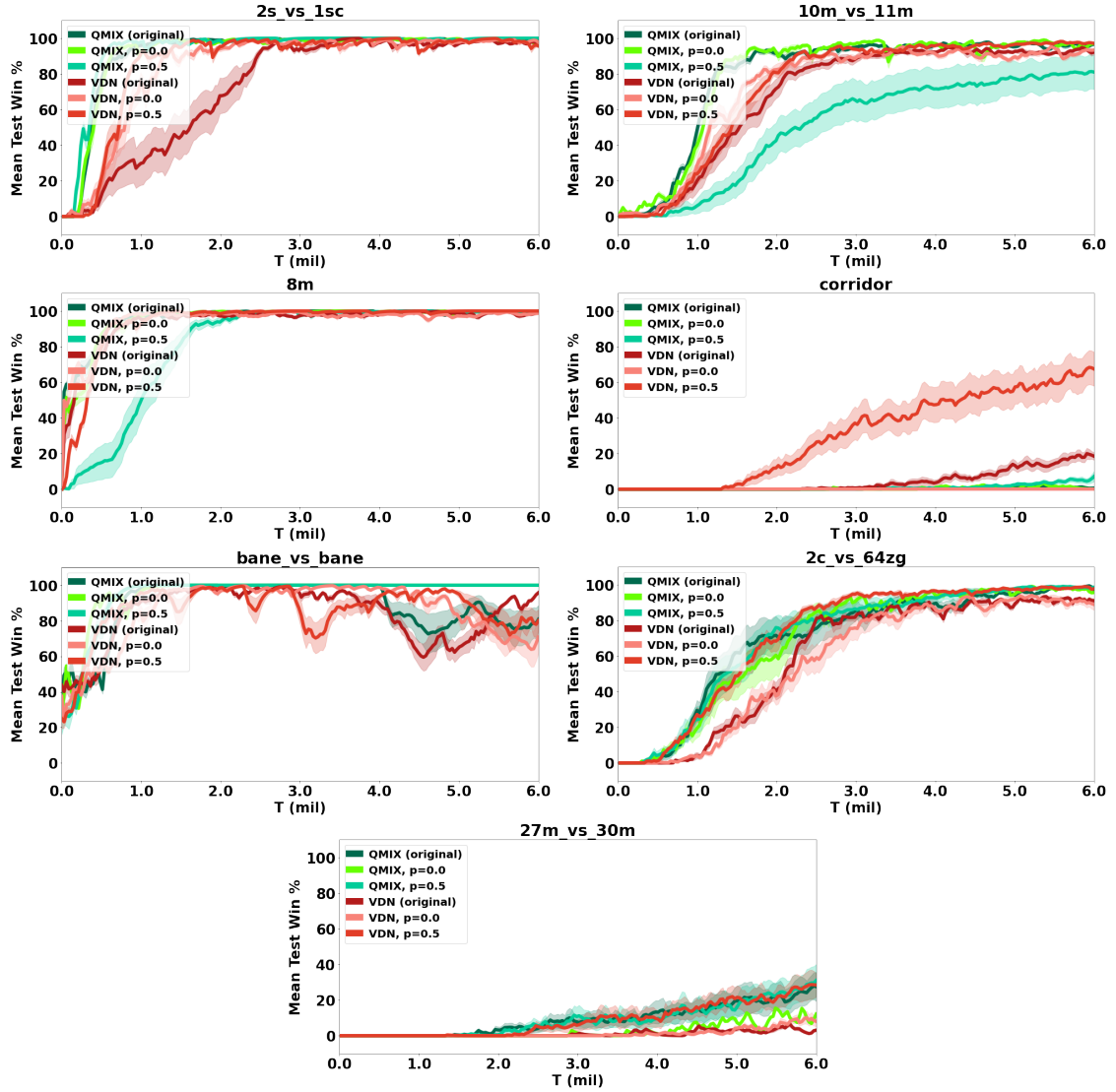


Figure 5.13: Comparison between VDN and QMIX baselines with original positive reward function and modified reward function with $p \in \{0.0, 0.5\}$ on the remaining SMAC maps.

designed with QMIX in mind, deliberately dropping any punishment for loosing ally units and thereby making it solvable for all considered baselines as shown in previous section. We therefore consider SMAC maps with negative reward functions (i.e. `reward-only-positive = False`) where each ally agent unit is additionally penalized using `reward-negative-scale` (denoted by p) for being killed or suffering damage from the enemy, in addition to receiving positive reward for killing/damage on enemy units. The additional penalty to the reward function (similar to QTRAN++, Son et al. [2020]) for losing our own ally units induces

RO in the same way as in our predator prey example. This makes maps that were classified as “easy” by default benchmark (e.g., “8m” in Figure 5.14) very hard to solve for methods like VDN and QMIX for $p = 1.0$ (equally weighting the lives of ally and enemy units).

Prior work on SC2 has established that VDN/QMIX can solve nearly all SMAC maps in the absence of any punishment (i.e., negative reward scaling $p = 0$) and we confirm this also holds for low punishments (see results in last section for $p = 0.5$). However, this puts little weight on the lives of your own allies and cautious generals may want to learn a more conservative policy with higher punishment for losing their ally units. Similarly to the predator-prey example, Figure 5.14 shows that increasing the punishment to $p = 1$ (equally weighting the lives of allies and enemies) leads VDN/QMIX and other SOTA MARL methods like QTRAN, WQMIX and QPLEX to fail in many maps, whereas our novel method UneVEn, in particular with greedy action selection, reliably outperforms baselines on all tested SMAC maps.

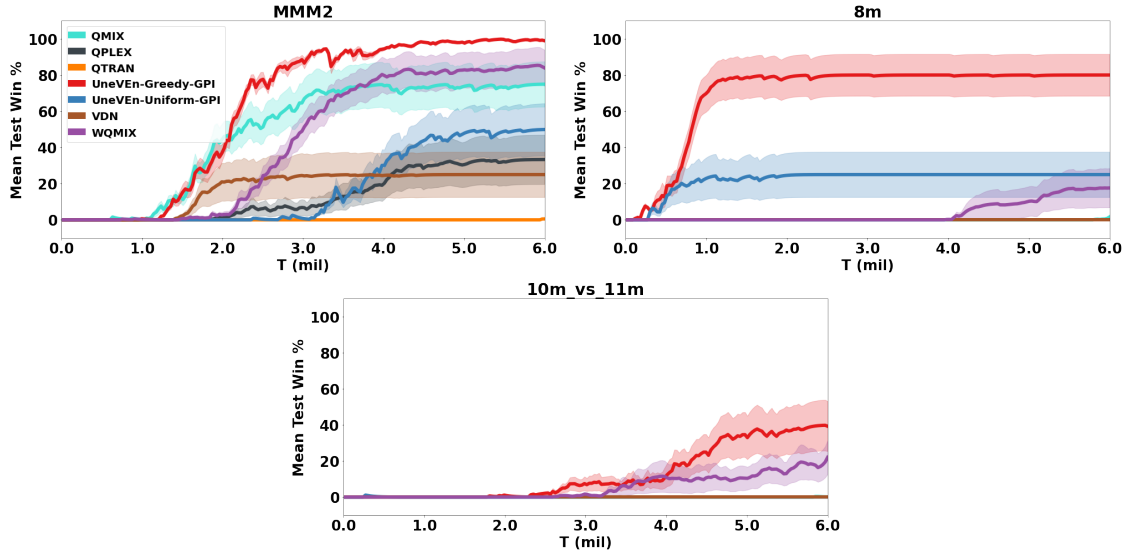


Figure 5.14: Comparison between UneVEn and SOTA MARL baselines on SMAC maps with $p = 1.0$.

5.10 Conclusion, Limitations and Future Work

This chapter presents a novel approach decomposing multi-agent universal successor features (MAUSFs) into local agent-specific SFs, which enable a decentralized version of the GPI to maximize over a combinatorial space of agent policies. We propose UneVEn, which leverages the generalization power of MAUSFs to perform action selection based on simpler related tasks to address the suboptimality of the target task’s monotonic joint action value function in current SOTA methods. Our experiments show that UneVEn significantly outperforms VDN, QMIX and other state-of-the-art value-based MARL methods on challenging tasks exhibiting severe RO.

Limitations and Future Work

Rather than choosing related tasks at the start of each episode, a more dynamic selection of related tasks can be implemented at each time step within an episode, specifically for the greedy GPI action selection scheme. This approach could decrease the number of tasks sampled in the current training setup, introducing greater diversity in sampled tasks throughout the episode. However, a reduction in sampled tasks might also result in more challenging unsolved tasks being chosen, potentially increasing the variance in learning.

UneVEn relies on the assumption that both the basis functions and the reward-weights of the target task are known which restricts the applicability of the method. Barreto et al. [2020] show that both can be learned using multi-task regression. However, depending on the coverage of the training data, usually collected with a random policy, only a subset of tasks might be present in the training data. As a result, regression-based approximation of the basis function and reward weights might only be able to capture a sub-set of the basis functions which only represent a subset of representable reward functions, and lead to high variance during learning. One way to mitigate this is to leverage semantic intrinsic motivation discussed in Chapter 3, or other maximum entropy based policies to collect diverse meaningful data. Another interesting direction is the one proposed by Hansen

et al. [2019] which learns controllable features in a variational framework that can be leveraged to provide enhanced generalization and fast task inference through the successor features framework.

Additionally, our SFs based method requires learning in d dimensions instead of learning scalar values for each action in the case of Q -learning, which decreases the scalability of our method for domains where d is very large. Future work may explore more efficient neural network architectures with hyper networks [Ha et al., 2016] can be leveraged to handle larger dimensional features. Additionally, the paradigm of UneVEn with related task based action selection can be directly applied with universal value functions (UVFs) [Schaul et al., 2015a] to enable other state-of-the-art nonlinear mixing techniques like QMIX [Rashid et al., 2020b] and QPLEX [Wang et al., 2020a] to overcome RO, at the cost of losing ability to perform GPI.

UneVEn Exploration with UVFs

UneVEn with its action-selection schemes can also be realized with UVFs of the form $Q_{tot}(s, \mathbf{u}, \mathbf{w})$ which is represented as a monotonic mixing (QMIX, VDN) of decentralized agent utility functions $Q^a(\tau^a, u^a, \mathbf{w}; \theta)$. One major disadvantage while using UVFs is that we can no longer efficiently compute a GPI policy by instant policy evaluation of joint-action value functions of related and target tasks on an unseen task. Therefore the action-selection schemes select actions based on task $\mathbf{k}$ using:

$$\mathbf{u}_t = \{u_t^a = \arg \max_{u \in \mathcal{U}} Q^a(\tau_t^a, u, \mathbf{k}; \theta)\}_{a \in A},$$

where $\mathbf{k} \in \nu \cup \{\mathbf{w}\}$ with the highest Q -value for **greedy**, and $\mathbf{k} \in \text{Uniform}(\nu)$ for **uniform** action-selection schemes. Setting $\mathbf{k} = \mathbf{w}$ becomes similar to standard VDN and QMIX as actions are always taken based on the target task.

6

Role Based Exploration

Contents

6.1	Introduction	102
6.2	RODE Learning Framework	104
6.2.1	Determining Role Action Spaces by Learning Action Representations	105
6.2.2	Learning the Role Selector and Role Policies	107
6.3	Related Work	110
6.3.1	Hierarchical MARL	110
6.3.2	Dealing with Large Discrete Action Spaces	111
6.3.3	Role-Based Learning	112
6.4	Experiments	113
6.4.1	Action Representations	114
6.4.2	Performance and Ablation Study	115
6.4.3	Case Study : Role Dynamics	118
6.4.4	Effect of Role Interval	120
6.4.5	Effect of Role Selector Architecture	121
6.4.6	Policy Transfer	122
6.4.7	Role Representations	124
6.5	Analysis on Clustering Algorithms	124
6.5.1	Influence of Number of Clusters k	125
6.5.2	Other Clustering Algorithms	126
6.6	Conclusion and Future Work	127

6.1 Introduction

In the previous chapter, we showcased how relative overgeneralization can affect learning control policies for cooperative multi-agent systems. In this chapter, we focus on yet another challenge i.e. exploration in joint action space which increases exponentially with the number of agents. Joint action learning [Claus and Boutilier, 1998] learns centralized policies conditioned on the full state, but this global information is often unavailable during execution due to partial observability or communication constraints. Independent learning [Tan, 1993] avoids this problem by learning decentralized policies but suffers from non-stationarity during learning as it treats other learning agents as part of the environment.

The framework of centralized training with decentralized execution (CTDE) [Foerster et al., 2016, Gupta et al., 2017, Rashid et al., 2018] combines the advantages of these two paradigms. Decentralized policies are learned in a centralized manner so that they can share information, parameters, etc., without restriction during training. Although CTDE algorithms can solve many multi-agent problems [Mahajan et al., 2019, Das et al., 2019, Wang et al., 2020c], during training they must search in the joint action-observation space, which grows exponentially with the number of agents. This makes it difficult to learn efficiently when the number of agents is large [Samvelyan et al., 2019].

Humans cooperate in a more effective way. When dealing with complex tasks, instead of directly conducting a collective search in the full action-observation space, they typically decompose the task and let sub-groups of individuals learn to solve different sub-tasks [Smith, 1937, Butler, 2012]. Once the task is decomposed, the complexity of cooperative learning can be effectively reduced because individuals can focus on restricted sub-problems, each of which often involves a smaller action-observation space. Such potential scalability motivates the use of roles in multi-agent tasks, in which each role is associated with a certain sub-task and a corresponding policy.

The key question in realizing such scalable learning is how to come up with a set of roles to effectively decompose the task. Previous work typically predefines the task decomposition and roles [Pavón and Gómez-Sanz, 2003, Cossentino et al., 2005, Spanoudakis and Moraitis, 2010, Bonjean et al., 2014]. However, this requires prior knowledge that might not be available in practice and may prevent the learning methods from transferring to different environments. Therefore, to be practical, it is crucial for role-based methods to automatically learn an appropriate set of roles. However, learning roles from scratch might not be easier than learning without roles, as directly finding an optimal decomposition suffers from the same problem as other CTDE learning methods – searching in the large joint space with substantial exploration [Wang et al., 2020b].

To solve this problem, we propose a novel framework for learning ROles to DEcompose (RODE) multi-agent tasks. Our key insight is that, instead of learning roles from scratch, role discovery is easier if we first decompose joint action spaces according to action functionality. Intuitively, when cooperating with other agents, only a subset of actions that can fulfill a certain functionality is needed under certain observations. For example, in football games, the player who does not possess the ball only needs to explore how to **move** or **sprint** when attacking. In practice, we propose to first learn effect-based action representations and cluster actions into role action spaces according to their effects on the environment and other agents. Then, with knowledge of effects of available actions, we train a role selector that determines corresponding role observation spaces.

This design forms a bi-level learning framework. At the top level, a role selector coordinates role assignments in a smaller role space and at a lower temporal resolution. At the low level, role policies explore strategies in reduced primitive action-observation spaces. In this way, the learning complexity is significantly reduced by decomposing a multi-agent cooperation problem, both temporally and spatially, into several short-horizon learning problems with fewer agents. To further improve learning efficiency on the sub-problems, we condition role policies on

the learned effect-based action representations, which improves generalizability of role policies across actions.

We test RODE on StarCraft II micromanagement environments [Samvelyan et al., 2019]. Results on this benchmark show that RODE establishes a new state of the art. Particularly, RODE has the best performance on 9 out of all 14 maps, including all 5 super hard maps and most hard maps. Visualizations of learned action representations, factored action spaces, and dynamics of role selections shed further light on the superior performance of RODE. We also demonstrate that conditioning the role selector and role policies on action representations enables learned RODE policies to be transferred to tasks with different numbers of actions and agents, including tasks with three times as many agents.

6.2 RODE Learning Framework

In this section, we introduce the RODE learning framework. We consider fully cooperative multi-agent tasks that can be modelled as a Dec-POMDP [Oliehoek et al., 2016] consisting of a tuple $G = \langle I, S, A, P, R, \Omega, O, n, \gamma \rangle$, where I is the finite set of n agents, $\gamma \in [0, 1)$ is the discount factor, and $s \in S$ is the true state of the environment. At each timestep, each agent i receives an observation $o_i \in \Omega$ drawn according to the observation function $O(s, i)$ and selects an action $a_i \in A$, forming a joint action $\mathbf{a} \in A^n$, leading to a next state s' according to the transition function $P(s'|s, \mathbf{a})$, and observing a reward $r = R(s, \mathbf{a})$ shared by all agents. Each agent has local action-observation history $\tau_i \in \mathbf{T} \equiv (\Omega \times A)^*$.

Our idea is to learn to decompose a multi-agent cooperative task into a set of sub-tasks, each of which has a much smaller action-observation space. Each sub-task is associated with a role, and agents taking the same role collectively learn a role policy for solving the sub-task by sharing their learning. Formally, we propose the following definition of sub-tasks and roles.

Definition 2 (Role and Sub-Task). Given a cooperative task $G=\langle I, S, A, P, R, \Omega, O, n, \gamma \rangle$, let Ψ be a set of roles. A role $\rho_j \in \Psi$ is a tuple $\langle g_j, \pi_{\rho_j} \rangle$ where $g_j = \langle I_j, S, A_j, P, R, \Omega_j, O, \gamma \rangle$ is a sub-task and $I_j \subset I$, $\cup_j I_j = I$, and $I_j \cap I_k = \emptyset$, $j \neq k$. A_j is the action space of role j , $A_j \subset A$, $\cup_j A_j = A$, and we allow action spaces of different roles to overlap: $|A_j \cap A_k| \geq 0$, $j \neq k$. $\pi_{\rho_j} : \mathcal{T} \times A_j \rightarrow [0, 1]$ is a role policy for the sub-task g_j associated with the role, where $\mathcal{T}$ is the local action observation history.

In our definition, a given set of roles specifies not only a factorization of the task but also policies for sub-tasks. This role-based formulation is effectively a kind of dynamic grouping mechanism where agents are grouped by roles. The benefit is that we can learn policies for roles instead of agents, providing scalability with the number of agents and reducing search space of target policies. In our framework, an agent decides which role to take by a *role selector* β that is shared among agents. Here, $\beta : \mathcal{T} \rightarrow \Psi$ is a deterministic function that is conditioned on local action-observation history. Our aim is to learn a set of roles Ψ^* that maximizes the expected global return

$$Q^\Psi(s_t, \mathbf{a}_t) = \mathbb{E}_{s_{t+1:\infty}, \mathbf{a}_{t+1:\infty}} \left[\sum_{i=0}^{\infty} \gamma^i r_{t+i} | s_t, \mathbf{a}_t, \Psi \right].$$

Learning roles involves two questions: learning sub-tasks and role policies. For learning sub-tasks, we need to learn the role selector, β , and to determine role action spaces, A_j . These two components determine the observation and action spaces of roles, respectively. We now introduce the framework to learn these components, which is shown in Fig. 6.1.

6.2.1 Determining Role Action Spaces by Learning Action Representations

Restricting the action space of each role can significantly reduce the policy search space. The key to our approach is to factor the action space according to actions' properties and thereby let roles focus on actions with similar effects, such as

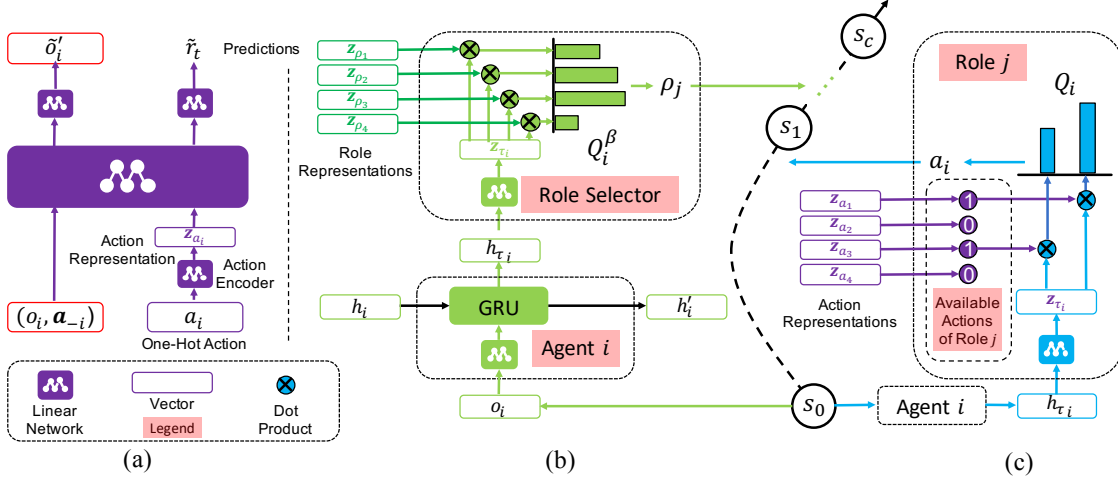


Figure 6.1: RODE framework. (a) The forward model for learning action representations. (b) Role selector architecture. (c) Role action spaces and role policy structure.

attacking enemy units of the same type in StarCraft II. To this end, we learn action representations that can reflect the effects of actions on the environment and other agents. The effect of an action can be measured by the induced reward and the change in local observations. Therefore, we formulate an objective for learning action representations that incentivizes including enough information such that the next observations and rewards can be predicted when given the actions of other agents and current observations. In practice, we use the predictive model shown in Fig. 6.1(a) to learn an action encoder:

$$f_e(\cdot; \theta_e) : \mathbb{R}^{|A|} \rightarrow \mathbb{R}^d.$$

This encoder, parameterized by θ_e , maps one-hot actions to a d -dimensional representation space. We denote the representation of an action a in this space by $\mathbf{z}_a$, *i.e.*,

$$\mathbf{z}_a = f_e(a; \theta_e),$$

which is then used to predict the next observation o_i' and the global reward r , given the current observation o_i of an agent i , and the one-hot actions of other agents, $\mathbf{a}_{-i}$. This model can be interpreted as a forward model, which is trained

by minimizing the following loss function:

$$\mathcal{L}_e(\theta_e, \xi_e) = \mathbb{E}_{(o, a, r, o') \sim \mathcal{D}} \left[\sum_i \|p_o(\mathbf{z}_{a_i}, o_i, \mathbf{a}_{-i}) - o'_i\|_2^2 + \lambda_e \sum_i (p_r(\mathbf{z}_{a_i}, o_i, \mathbf{a}_{-i}) - r)^2 \right], \quad (6.1)$$

where p_o and p_r are predictors for observations and rewards, respectively, and parameterized by ξ_e . λ_e is a scaling factor, $\mathcal{D}$ is an replay buffer, and the sum is carried out over all the agents. One can also use the action-observation history of the agent to learn action representations, however, in our work, we only used local observations.

In the beginning, we initialize K roles, each with the full action space. After collecting samples and training the predictive model for t_e timesteps, we cluster the actions according to their latent representations and set the action space of each role to contain one of the clusters. The intuition is that actions which lead to similar effects on the environment and other agents will be clustered together, defining the roles and their action spaces, each resembling a sub-task. Outliers are added to all the clusters to avoid action sets consisting of only one action. After this update, training begins and action representations and action spaces of each role are kept fixed during training. In this chapter, we simply use k -means clustering based on Euclidean distances between action representations, but it can be easily extended to other clustering methods. In Section 6.5, we investigate the influence of different k values and how to determine the number of clusters automatically. In practice, we find that the forward model converges quickly. For example, a well-formed representation space can be learned using $50K$ samples on StarCraft II micromanagement tasks, while the training usually lasts for $2M$ timesteps.

6.2.2 Learning the Role Selector and Role Policies

Learning action representations to cluster actions based on their effects leads to a factorization of the action space, where each subset of actions can fulfill a certain functionality. To make full use of such a factorization, we use a bi-level hierarchical structure to coordinate the selection of roles and primitive actions. At the top

level, a role selector assigns a role to each agent every c timesteps. After a role is assigned, an agent explores in the corresponding restricted role action space to learn the role policy. We now describe how to design efficient, transferable, and lightweight structures for the role selector and role policies.

Role Selector

For the role selector, we can simply use a conventional Q -network whose input is local action-observation history and output is Q -values for each role. However, this structure may not be efficient because it ignores information about action space of different roles. Intuitively, selecting a role is selecting a subset of actions to execute for the next c timesteps, so the Q -value of selecting a role is closely related to its role action space. Therefore, we propose to base the role selector on the average representation of available actions. Fig. 6.1(b) shows the proposed role selector structure. Specifically, we call

$$\mathbf{z}_{\rho_j} = \frac{1}{|A_j|} \sum_{a_k \in A_j} \mathbf{z}_{a_k}, \quad (6.2)$$

the representation of role ρ_j , where A_j is its restricted action space. To select roles, agents share a linear layer and a GRU, parameterized by θ_{τ_β} , to encode the local action-observation history τ into a fixed-length vector h_τ . The role selector, a fully connected network $f_\beta(h_\tau; \theta_\beta)$ with parameters θ_β , then maps h_τ to $\mathbf{z}_\tau \in \mathbb{R}^d$, and we estimate the expected return of agent i selecting role ρ_j as:

$$Q_i^\beta(\tau_i, \rho_j) = \mathbf{z}_{\tau_i}^\top \mathbf{z}_{\rho_j}. \quad (6.3)$$

Agents select their roles concurrently, which may lead to unintended behaviors. For example, all agents choosing to attack a certain unit type in StarCraft II causes overkill. To better coordinate role assignments, we use a mixing network to estimate global Q -values, Q_{tot}^β , using per-agent utility Q_i^β so that the role selector can be trained using global rewards. In this chapter, we use the mixing network introduced by QMIX [Rashid et al., 2018] for its monotonic approximation. The parameters

of the mixing network are conditioned on the global state s and are generated by a hypernetwork [Ha et al., 2016] parameterized by ξ_β . Then we minimize the following TD loss to update the role selector f_β :

$$\mathcal{L}_\beta(\theta_{\tau_\beta}, \theta_\beta, \xi_\beta) = \mathbb{E}_{\mathcal{D}} \left[\left(\sum_{t'=0}^{c-1} r_{t+t'} + \gamma \max_{\rho'} \bar{Q}_{tot}^\beta(s_{t+c}, \rho') - Q_{tot}^\beta(s_t, \rho_t) \right)^2 \right], \quad (6.4)$$

where $\bar{Q}_{tot}^\beta$ is a target network, $\rho = \langle \rho_1, \rho_2, \dots, \rho_n \rangle$ is the joint role of all agents, and the expectation is estimated with uniform samples from a replay buffer $\mathcal{D}$.

Role Policies

After a role is assigned, an agent follows it for the next c timesteps, during which it can only take actions in the corresponding role action space. Each role ρ_j has a role policy $\pi_{\rho_j} : \mathcal{T} \times \mathcal{A}_j \rightarrow [0, 1]$ that is defined in the restricted action space. Similar to the role selector, the most straightforward approach to learning role policies is with a conventional deep Q -network that directly estimates the Q -values of each action. However, basing Q -values on action representations makes full use of the effect information of actions and can generalize better across actions. Therefore, we use the framework shown in Fig. 6.1(c) for learning role policies. Specifically, agents again use a shared linear layer and a GRU which together encode a local action-observation history τ into a vector h_τ . We denote the parameters of these two networks by θ_{τ_ρ} . Each role policy is a fully connected network $f_{\rho_j}(h_\tau; \theta_{\rho_j})$ with parameters θ_{ρ_j} , whose input is h_τ . f_{ρ_j} maps h_τ to $\mathbf{z}_\tau$, which is a vector in $\mathbb{R}^d$. Using action representations $\mathbf{z}_{a_k}$ and $\mathbf{z}_\tau$, we estimate the value of agent i choosing a primitive action a_k as:

$$Q_i(\tau_i, a_k) = \mathbf{z}_{\tau_i}^T \mathbf{z}_{a_k}. \quad (6.5)$$

For learning the Q_i using global rewards, we again feed local Q -values into a QMIX-style mixing network to estimate the global action-value, $Q_{tot}(s, \mathbf{a})$. The parameters of the mixing network are denoted by ξ_ρ . This formulation gives the TD loss for learning role policies:

$$\mathcal{L}_\rho(\theta_{\tau_\rho}, \theta_\rho, \xi_\rho) = \mathbb{E}_{\mathcal{D}} \left[\left(r + \gamma \max_{\mathbf{a}'} \bar{Q}_{tot}(s', \mathbf{a}') - Q_{tot}(s, \mathbf{a}) \right)^2 \right]. \quad (6.6)$$

Here, $\bar{Q}_{tot}$ is a target network, θ_ρ is the parameters of all role policies, and the expectation is estimated with uniform samples from the same replay buffer $\mathcal{D}$ as the one for the role selector. Moreover, two mixing networks for training the role selector and role policies are only used during training. By virtue of effect-based action representations, our framework is lightweight – in practice, we use a single linear layer without activation functions for role policies and a two-layer fully-connected network for the role selector. Basing the role selector and role policies on action representations also enable RODE policies to be rapidly transferred to new tasks with different numbers of agents and actions, which we discuss in detail in Sec. 6.4.6.

6.3 Related Work

6.3.1 Hierarchical MARL

To realize efficient role-based learning, RODE adopts a hierarchical decision-making structure. Hierarchical reinforcement learning (HRL) [Sutton et al., 1999b, Al-Emran, 2015] has been extensively studied to address the sparse reward problem and to facilitate transfer learning. Single-agent HRL focuses on learning temporal decomposition of tasks, either by learning sub-goals [Nachum et al., 2018b,a, Levy et al., 2018, Ghosh et al., 2018, Sukhbaatar et al., 2018, Dwiel et al., 2019, Nair and Finn, 2019, Nasiriany et al., 2019, Dilokthanakul et al., 2019] or by discovering reusable skills [Daniel et al., 2012, Gregor et al., 2016, Warde-Farley et al., 2018, Shankar and Gupta, 2020, Thomas et al., 2018, Sharma et al., 2020]. In multi-agent settings, many challenges, such as efficient communication [Ossenkopf et al., 2019] and labor division [Wang et al., 2020b] in large systems, necessitate hierarchical learning along the second dimension – over agents [Zhang et al., 2010]. Ahilan and Dayan [2019] propose a hierarchy in FeUdal [Dayan and Hinton, 1993, Vezhnevets et al., 2017] style for cooperative multi-agent tasks, where managers and workers are predefined and workers are expected to achieve the goal generated by managers. Similar hierarchical organizations among agents have been exploited to control traffic

lights [Jin and Ma, 2018]. Lee et al. [2019b], Yang et al. [2019] propose to use bi-level hierarchies to train (or discover) and coordinate individual skills. Vezhnevets et al. [2020] extend hierarchical MARL to Markov Games, where a high-level policy chooses strategic responses to opponents. RODE proposes to learn multi-agent coordination skills in restricted action spaces and is thus complementary to these previous works.

6.3.2 Dealing with Large Discrete Action Spaces

In single-agent settings, being able to reason with a large number of discrete actions is essential to bringing reinforcement learning to a larger class of problems, such as natural language processing [He et al., 2016] and recommender systems [Ayush et al., 2020]. Sallans and Hinton [2004], Pazis and Parr [2011] represent actions using binary code to reduce the size of large action spaces. Cui and Khardon [2016, 2018] address scalability issues of planning problems by conducting a gradient-based search on a symbolic representation of the state-action value function. Sharma et al. [2017] show that factoring actions into their primary categories can improve performance of deep RL algorithms on Atari 2600 games [Bellemare et al., 2013]. All aforementioned methods assume that a handcrafted binary decomposition of raw actions is provided. Another interesting way to deal with large discrete state-action spaces is by estimating equivalence classes over reward trajectories and learn on quotient MDPs [Mahajan and Tulabandhula, 2017a,b].

Yet another line of research uses continuous action representations to discover underlying structures of large discrete action spaces, thereby easing training. Van Hasselt and Wiering [2009] use policy gradients with continuous actions and select the nearest discrete action. Dulac-Arnold et al. [2015] extend this work to large-scale problems but use predefined action embeddings. Tennenholtz and Mannor [2019], Chandak et al. [2019] avoid pre-definitions by extracting action representations from expert demonstrations or without prior knowledge. Action representations have been used to solve natural language processing problems [He et al., 2016], enable generalization to unseen [Ayush et al., 2020] or under-explored [Chandak

et al., 2019] actions in RL, learn linear dynamics that help exploration [Kim et al., 2019], and transfer policy across different tasks [Chen et al., 2019]. By contrast, we use action representations to factor multi-agent tasks.

Besides learning action representations, Czarnecki et al. [2018], Murali et al. [2018], Farquhar et al. [2020] propose to use curriculum learning over action spaces of increasing sizes to scale learning to large spaces. Although the large action space problem is more demanding in multi-agent settings, how to organize effective learning in them is largely unstudied – most algorithms directly search in the full action spaces, including previous works [Wang et al., 2020d] that consider the effects (semantics) of actions like RODE. This is a key motivation for RODE, where search spaces are reduced by an effect-based action space factorization and a role-based hierarchical structure designed for efficient learning in these factored action spaces.

6.3.3 Role-Based Learning

Our approach is an attempt to integrate roles into deep multi-agent reinforcement learning. Many natural systems feature emergent roles, such as ants [Gordon, 1996], bees [Jeanson et al., 2005], and humans [Butler, 2012]. In these systems, roles are closely related to the division of labor and is critical to labor efficiency. These benefits inspired multi-agent system designers, who try to reduce the design complexity by decomposing the task and specializing agents with the same role to certain sub-tasks [Wooldridge et al., 2000, Omicini, 2000, Padgham and Winikoff, 2002, Pavón and Gómez-Sanz, 2003, Cossentino et al., 2005, Zhu and Zhou, 2008, Spanoudakis and Moraitis, 2010, DeLoach and Garcia-Ojeda, 2010, Bonjean et al., 2014]. However, roles and the associated responsibilities (or subtask-specific rewards [Sun et al., 2020]) are predefined using prior knowledge in these systems [Lhaksmana et al., 2018]. Although pre-definition can be efficient in tasks with a clear structure, such as software engineering [Bresciani et al., 2004], it hurts generalization and requires prior knowledge that may not be available in practice. To solve this problem, Wilson et al. [2010] use Bayesian inference to learn a set of roles and

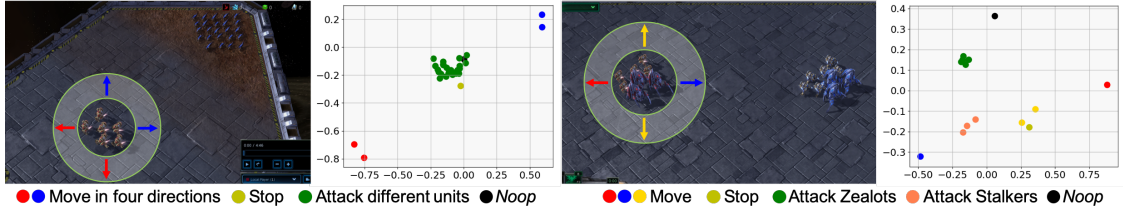


Figure 6.2: Action representations learned by our predictive model on *corridor* (left) and *3s5z_vs_3s6z* (right). Blue arrows indicate the directions moving towards enemies while red ones show the directions moving away from enemies.

Wang et al. [2020b] design a specialization objective to encourage the emergence of roles. However, these methods search the optimal task decomposition in the full state-action space, resulting in inefficient learning in hard-exploration tasks. RODE avoids this shortcoming by first decomposing joint action spaces according to action effects, which makes role discovery much easier.

6.4 Experiments

We design experiments to answer the following questions:

- Can the proposed predictive model learn action representations that reflect effects of actions? (Sec. 6.4.1)
- Can RODE improve learning efficiency? If so, which component contributes the most to the performance gains? (Sec. 6.4.2)
- Can the role selector learn interpretable high-level cooperative behaviors? (Section. 6.4.3)
- Can RODE support rapid transfer to environments with different numbers of actions and agents? (Sec. 6.4.6)

We choose the StarCraft II micromanagement (SMAC) benchmark [Samvelyan et al., 2019] as the testbed for its rich environments and high complexity of control. The SMAC benchmark requires learning policies in a large action space. Agents

can **move** in four cardinal directions, **stop**, take **noop** (do nothing), or select an enemy to **attack** at each timestep. Therefore, if there are n_e enemies in the map, the action space for each ally unit consists of $n_e + 6$ discrete actions. Demonstrative videos can be viewed at <https://sites.google.com/view/rode-marl>.

6.4.1 Action Representations

For learning action representations, on all maps, we set the scaling factor λ_e to 10 and collect samples and train the predictive model for $50K$ timesteps using the loss function described by Eq. 6.1. The model is trained every episode using a batch of 32 episodes. In Fig. 6.2, we show the learned action representations on two maps, **3s5z_vs_3s6z** and **corridor**.

The map **corridor** features homogeneous agents and enemies, with 6 Zealots faced with 24 enemy Zerglings. In this task, all attack actions have similar effects because the enemies are homogeneous. Moreover, due to the symmetry of the map, moving northward and eastward exert similar influence on the environment – moving the agent towards enemies. Likewise, moving southward and westward have the same effect because both actions move agents away from enemies. The learned action encoder captures these underlying structures in the action space (Fig. 6.2 left). We can observe three clear clusters in the latent action representation space, which correspond to the aforementioned three types of actions, respectively. On the map **3s5z_vs_3s6z**, 3 Stalkers and 5 Zealots try to beat 3 enemy Stalkers and 6 Enemy Zealots. Unlike **corridor**, the enemy team consists of heterogeneous units, and attacking Stalkers and Zealots have distinct effects. Our predictive model highlights this difference – the attack actions are clustered into two groups (Fig. 6.2 right). Moreover, the learned action encoder also captures the dissimilarity between moving northward/southward and moving in other directions – these two move actions do not bring agents near to or far away from enemies. Although we only show results on two maps, each with one random seed, similar action clusters can be observed on all the scenarios and are robust across different seeds. Therefore,

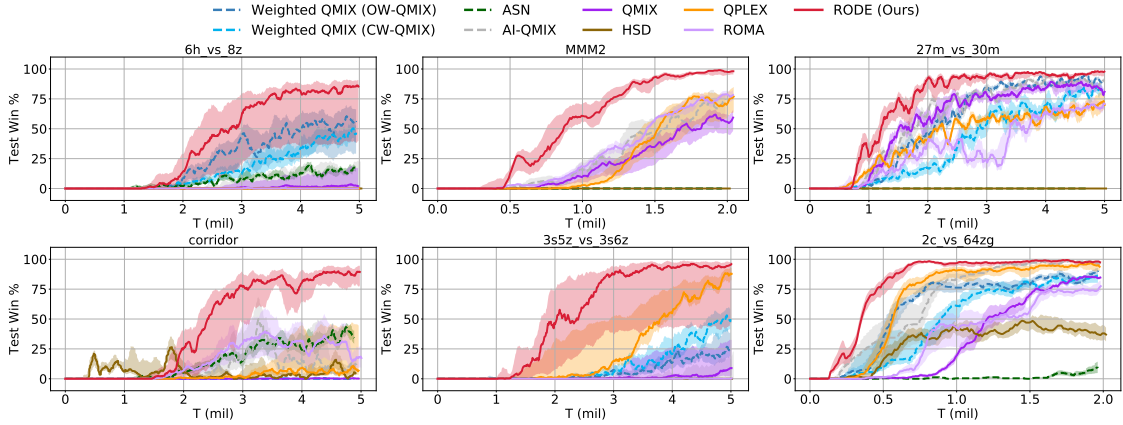


Figure 6.3: Performance comparison with baselines on *all super hard* maps and one *hard* map (2c_vs_64zg). In Appendix C.2.1, we show results on the whole benchmark.

we conclude that our predictive model effectively learns action representations that reflect the effects of actions.

6.4.2 Performance and Ablation Study

For evaluation, all experiments in this section are carried out with 8 different random seeds, and the median performance as well as the 25-75% percentiles are shown. The detailed setting of hyper-parameters is described in Appendix C.1. We benchmark our method on all 14 scenarios in order to evaluate its performance across the entire SMAC suite, and the results are shown in Fig. 6.4. We compare with current state-of-the-art value-based MARL algorithms (QMIX [Rashid et al., 2018], QPLEX [Wang et al., 2020a], Weighted QMIX [Rashid et al., 2020a], and AI-QMIX [Iqbal et al., 2020]), hierarchical MARL method (HSD, Yang et al. [2019]), role-based MARL method (ROMA, Wang et al. [2020b]), and MARL method based on action semantics (ASN, Wang et al. [2020d]). RODE outperforms all the baselines by at least 1/32 on **9** of all **14** scenarios after $2M$ training steps.

Maps of the SMAC benchmark have been classified as *easy*, *hard*, and *super hard*. Hard and super hard maps are typically hard-exploration tasks. Since RODE is designed to help exploration by decomposing the joint action space, we are especially interested in the performance of our method on these maps. Fig. 6.3 shows the

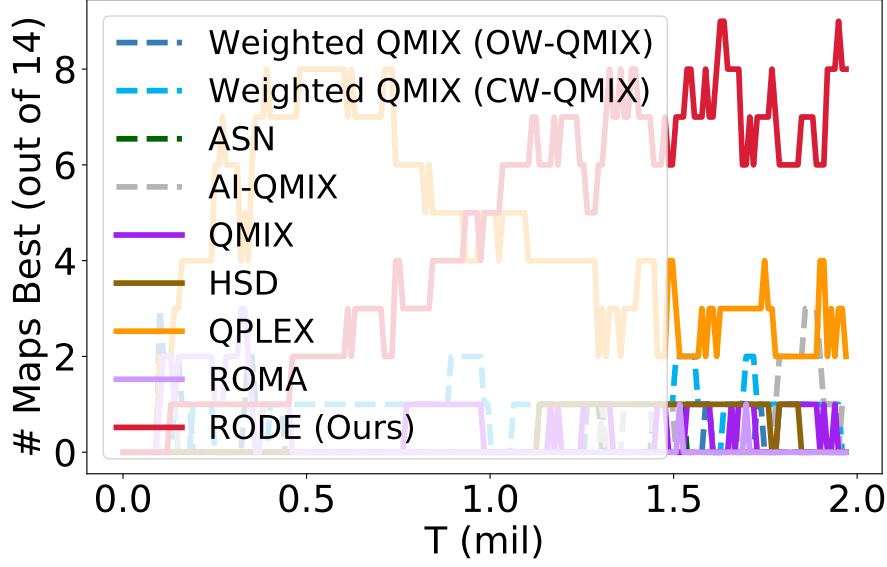


Figure 6.4: The number of scenarios (out of all 14 scenarios) in which the algorithm’s median test win % is the highest by at least $1/32$.

performance of RODE on all the super hard scenarios and one hard scenario. In Appendix C.2.1, we further show learning curves on the whole benchmark. We can see on all super hard maps and most hard maps, RODE has the best performance. Moreover, RODE outperforms baselines by the largest margin on the maps that require more exploration: `3s5z_vs_3s6z`, `corridor`, and `6h_vs_8z`. In contrast, on 5 easy maps, RODE typically needs more samples to learn a successful strategy. We hypothesize that this is because these maps do not require substantial exploration, and the exploratory benefit brought by restricted action spaces is not obvious.

Ablations

To understand the superior performance of RODE, we carry out ablation studies to test the contribution of its four main components:

- (A) Restricted role action spaces;
- (B) Using action effects to cluster actions;
- (C) The integration of action representations into role policies and the role selector;
- (D) The hierarchical learning structure.

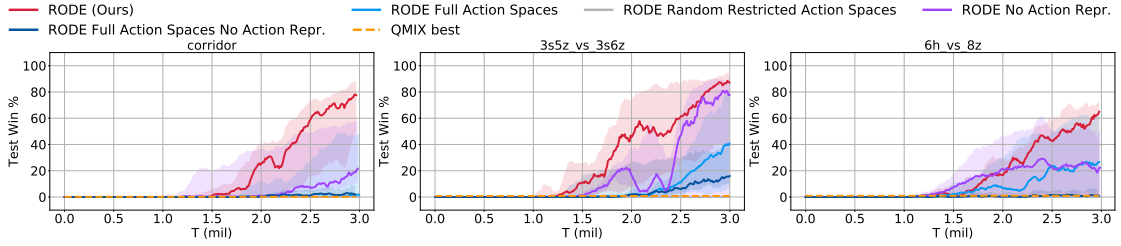


Figure 6.5: Ablation studies regarding each component of RODE. The best performance that can be achieved by QMIX is shown as horizontal dashed lines.

To test component A and B, when updating role action spaces after $50K$ samples, we let each role action space contains all actions or a random subset of actions, respectively, while keeping other parts in the framework unchanged. For component B, we make sure the union of role action spaces equals the entire action space. To test component C, we can simply use conventional Q -networks for learning role policies and the role selector. The testing of component D is a bit different because we cannot ablate it separately while still using different roles. Therefore, we test it by ablating components A and C – using conventional deep Q -networks for role policies and the role selector and allowing role policies to choose from all primitive actions.

The results on the three most difficult scenarios from the benchmark are shown in Fig. 6.5. Generally speaking, performance of RODE using conventional deep Q -networks (*RODE No Action Repr.*) is close to that of RODE, and is significantly better than the other two ablations where role action spaces are not restricted. Therefore, using action representations in policy learning can help improve learning but the superior performance of RODE is mostly due to the restriction of the role action spaces. RODE with full role action spaces and conventional Q -networks (*RODE Full Action Spaces No Action Repr.*) has similar performance to QMIX, which demonstrates that the hierarchical structure itself does not contribute much to the performance gains. RODE with random restricted role action spaces also cannot outperform QMIX, highlighting the importance of using action effects to decompose joint action spaces.

In summary, by virtue of restricted role action spaces and the integration of action effect information into policies, RODE achieves a new state of the art on the SMAC benchmark. In the next section (6.4.3), we provide a case study to better understand why restricting role action spaces can improve performance. We study the effects of the interval between role selections in Section 6.4.4.

6.4.3 Case Study : Role Dynamics

In this section, we provide a case study to explore the dynamics of the role selector and give an example to explain why restricting action spaces of roles can improve learning performance. We carry out our study on the map `corridor` from the SMAC benchmark. We choose this map because it presents a hard-exploration task. An optimal strategy for this scenario requires active state space exploration for the allied Zealots to learn to move to edges or to the choke point on the map, so as to not be surrounded by the enemy army. Previous state-of-the-art algorithms all learn a suboptimal strategy, where agents merely damage the enemy units for rewards, instead of first moving to edges and then attacking. In Fig. 6.6, we show the roles assigned by the learned role selector at different timesteps in an episode. The first row shows game snapshots, and the second row presents the corresponding role assignments. Here, the action encoder gives three clusters of actions (see Fig. 6.2left) – `move eastward & northward`, `attack`, and `move westward & southward`. These clusters of actions form the action spaces of three roles. The color of circles indicates the role of an ally unit.

We see that all agents select *Role 2* in the beginning (Fig. 6.6 (a)) to help them move to the edges, which is the good strategy as discussed above. The agents also learn an effective cooperative policy – one of the agents (agent A) attracts most of the enemies (enemy group A) so that its teammates have an advantage when facing fewer enemies (enemy group B). After killing agent A, enemy group A goes through the choke point and arrives their destination. In Fig. 6.6 (b), the alive agents start to attack this group of enemies. However, the enemies are still

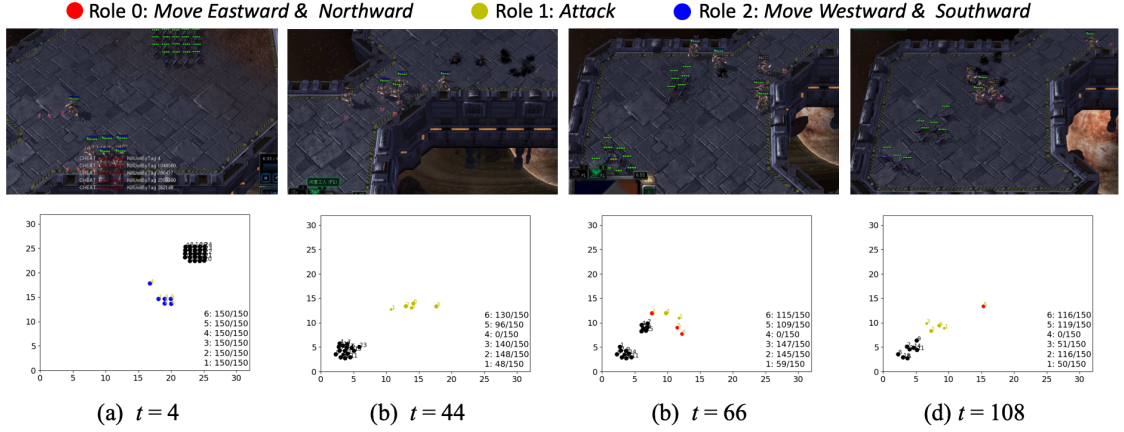


Figure 6.6: Visualization of roles in one episode on corridor. The first row shows game snapshots. In the second row, black circles are enemies and circles in other colors are ally agents, with different colors indicating different assigned roles. The size of circles is proportional to the remaining health points of units. The remaining and maximum health points are also shown in the lower right corner of the figures.

very powerful. To win the game, ally agents learn to attract part of enemies and kill them. They achieve this by alternating between *Role 0* (**retract** in this case) and *Role 1* (**attack**), as shown in Fig. 6.6 (b-d).

Why do restricted action spaces work?

Our effect-based action clustering decomposes the full action spaces into smaller role action spaces, each of which focuses on a smaller subset of actions having similar effects. As each role learns on a smaller action space which has similar actions, it enables agents to efficiently explore and results in each role being able to learn different strategies, like spread and retracting. Moreover, exploration in the much smaller role space makes the coordination of these sub-strategies easily.

Exploring in the hierarchically decomposed action spaces also provides a bias in exploration space. For example, on corridor, *Role 0* and *Role 2* motivate agents to explore the state space in a certain direction which help them learn several important strategies on this map: 1) Agents first move to the edges (*Role 0*) to avoid being surrounded by the enemies (Fig. 6.6 (a)), and 2) Agents alternate between attacking (*Role 1*) and retracting (*Role 2*) to attract and kill part of the enemies, gaining

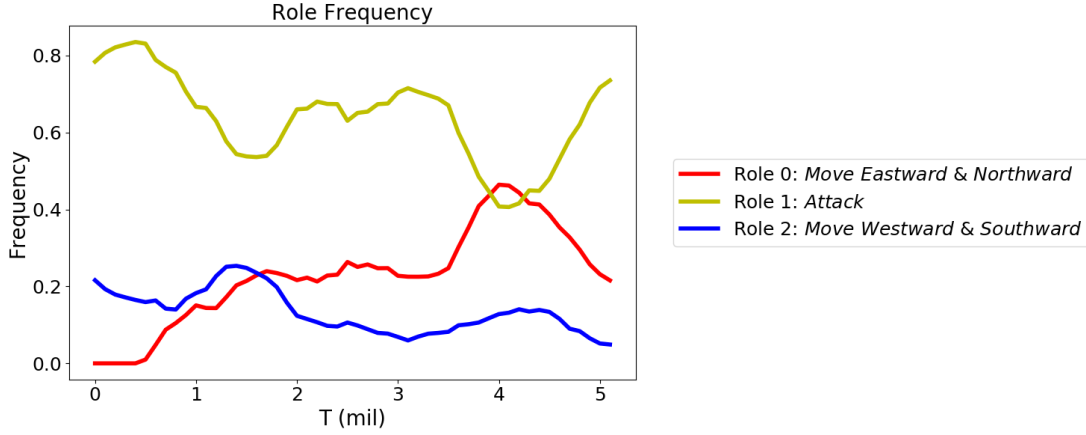


Figure 6.7: Frequencies of taking each role during training on *corridor*. Correspond to Fig. 6.6. The frequencies of two roles which help exploration (*Role 0* and *Role 2*) first increase and then decrease.

an advantage in numbers (Fig. 6.6 (b-d)). The fact that *Role 0* and *Role 2* help exploration is also supported by Fig. 6.7 – their frequencies first improve as the agents explore more, and then decrease when cooperation strategies gradually converge.

On other maps, restricted role action spaces may help agents solve the task in other ways. For example, on the map *3s5z_vs_3s6z*, agents learn more efficiently because the action encoder decomposes the task into two sub-tasks, one focusing on attacking Zealots and another one focusing on attacking Stalkers. Learning policies for these restricted sub-problems significantly reduces the learning complexity. In summary, action effects provide effective information to decompose joint action spaces. Clustering actions according to their effects makes role discovery much easier, and learning cooperative strategies in smaller action spaces makes learning more tractable.

6.4.4 Effect of Role Interval

In our framework, the role selector assigns roles to each agent every c timesteps. We call c the role interval. The role interval decides how frequently the action spaces change and may have a critical influence on the performance. To get a better sense of this influence, we change the role interval while keeping other parts unchanged and test RODE on several environments. In Fig. 6.8, we show the results on 2 easy

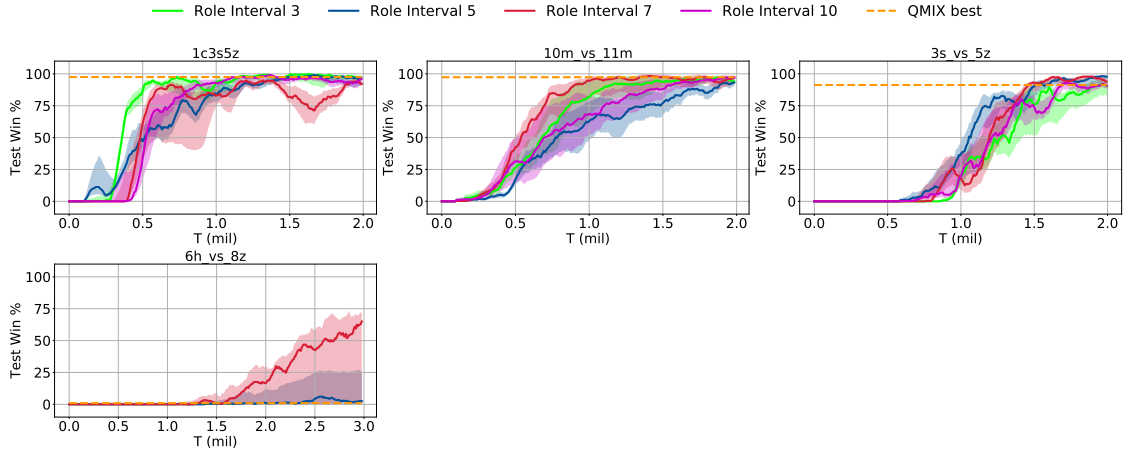


Figure 6.8: Influence of the role interval (the number of timesteps between two consecutive role selections).

maps (10m_vs_11m and 1c3s5z), 1 hard map (3s_vs_5z), and 1 super hard map (6h_vs_8z). Generally speaking, the role interval has a significant influence on performance, but 5 or 7 can typically generate satisfactory results. In this chapter, we use a role interval of 5 on most of the maps.

6.4.5 Effect of Role Selector Architecture

The role selector is supposed to learn a decomposition of observation space. However, we base this component on local action-observation history. In this section, we discuss the influence of using GRUs in the role selector. In Fig. 6.9, we compare RODE to *RODE with a fully-connected role selector* conditioned on local observations. As expected, a recurrent role selector performs better than a selector based on local observations. Even though this conclusion is not surprising, we are interested in those environments where a recurrent role selector can not perform better. For map 10m_vs_11m, together with the results presented in Fig. 6.8, we see that RODE with a role interval of 7 has similar performance to a fully-connected role selector, but underperforms when the role interval is set to other values. This indicates that the role interval has a sophisticated interaction with exploration, optimization, and learning stability. We leave the study of this as future work. In the previous section, we give more examples showing the influence of the role interval.

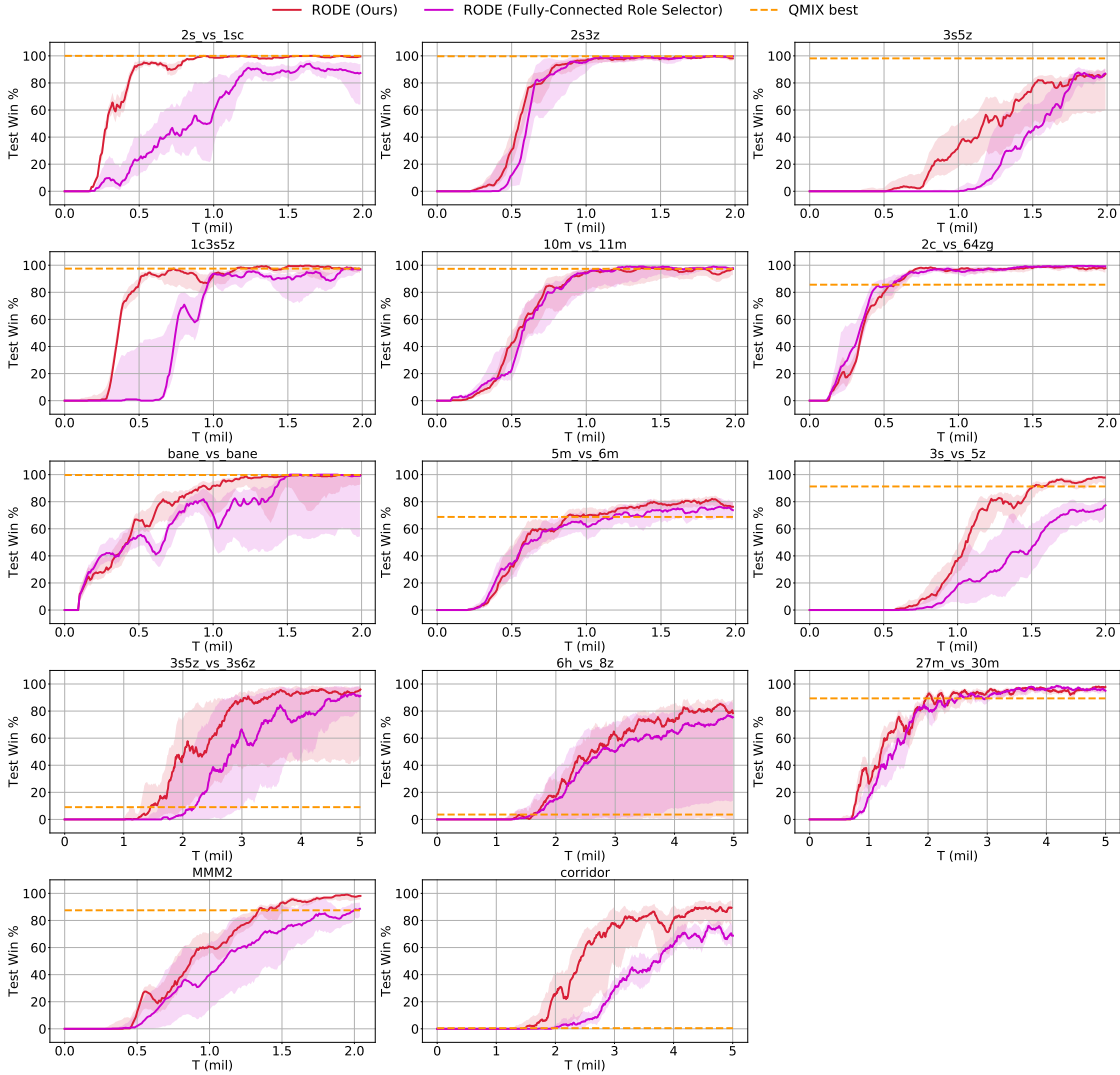


Figure 6.9: Comparisons between RODE with a recurrent role selector (RODE) and a fully-connected role selector conditioned on local observations (RODE (Fully-Connected Role Selector)).

6.4.6 Policy Transfer

A bonus of basing role policies on action representations is that we can transfer the learned policies to tasks with new actions that have similar effects to old ones. To demonstrate this, given a new task, we first collect some samples (50k) and train an action encoder. In this way, we identify the old actions that have similar functionality to the new actions. Then we use the average representation of these old actions as representations for new actions and add new actions to role action spaces to which the similar old actions belong. Since role action spaces are still

# Enemies	24	26	28	30	32	34	36
# Allies							
6	99.2	85.0	25.8	0.0	0.8	0.0	0.0
8	75.0	68.3	6.7	0.0	0.0	0.0	0.0
10	92.5	75.0	24.2	8.3	2.5	1.7	0.0
12	97.5	75.0	74.2	29.2	17.5	9.2	1.7
14	97.5	60.0	69.2	42.5	38.3	22.5	5.8
16	100	55.0	63.3	42.5	48.3	38.3	32.5
18	100	56.7	71.7	54.2	45.8	45.0	50.0

Figure 6.10: Win rates on unseen maps of the policy learned on `corridor`, where 6 ally Zealots face 24 Zerglings. We do not train policies on new maps.

expected to contain actions with similar effects, which can be reflected by old role representations, we do not re-calculate the role representations on the new task. Moreover, in our framework, it is roles that are interacting with the environment, and agents only need to be assigned with roles. Therefore, our learned policies can also be transferred to tasks with different numbers of agents.

We test the transferability of our method on the SMAC benchmark. To keep the length of observations fixed, we sort allies and enemies by their relative distances to an agent and include the information of nearest N_a allies and N_e enemies in the visible range. In Fig. 6.10, we show win rates of the policy learned from map `corridor` on different maps without further policy training. In the original task, 6 Zealots face 24 enemy Zerglings. We set N_a to 5 and N_e to 24. We increase the number of agents and enemies, and the number of actions increases as the number of enemies grows. The transferability of RODE is attested by observing that the

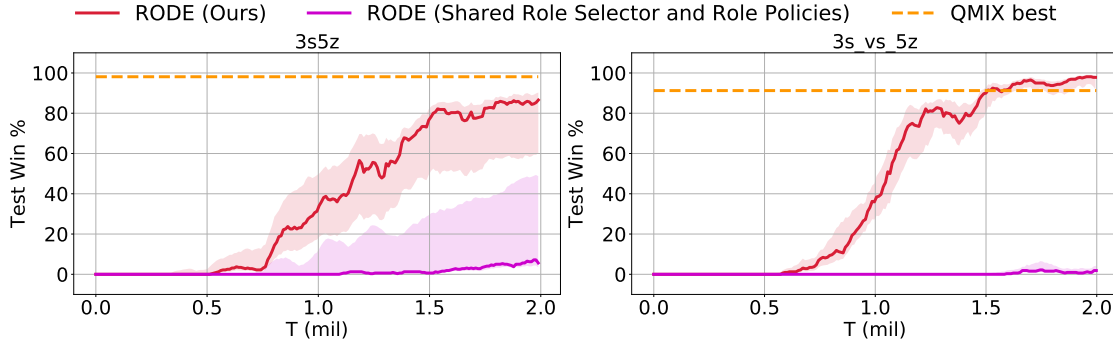


Figure 6.11: Comparisons between RODE and RODE with shared role policies and role selector. The separation of the role selector and role policies is important to make the average of action representations well represent the roles.

learned policy can still win 50% of the games on unseen maps with three times the number of agents. More details of this experiment is described in Appendix C.2.2. Besides rapid transfer, iteratively training the transferred policies on larger tasks may be a promising future direction to scaling MARL to large-scale problems.

6.4.7 Role Representations

In our framework, we use the average representation of actions in the role action space to represent a role. One concern is that whether such representations can well represent the roles, given that both role policies and the role selector are based on action representations. We find that the separation of Q networks for the role selector and role policies (including the GRUs for processing local observations and the network f_β (or f_{ρ_j})) is important to ensure such role representations have sufficient expressivity. If we share the GRUs and f_β (or f_{ρ_j}), RODE struggles even on easy maps (Fig. 6.11).

6.5 Analysis on Clustering Algorithms

In our framework, we use k -means to cluster action representations to determine the role action spaces. k -means algorithm requires predefining the number of clusters k . In this section, we first investigate the influence of k values on the performance and then discuss how to determine the number of clusters with minimum prior knowledge.

6.5.1 Influence of Number of Clusters k

We test the influence of k values on 7 maps, including 2 easy maps (**3s5z** and **10m_vs_11m**), 2 hard maps (**2c_vs_64zg** and **MMM2**), and 3 super hard maps (**3s5z_vs_3s6z**, **6h_vs_8z**, and **corridor**). In Fig. 6.12, we show the performance of RODE with k -values of 3, 5, and 7. Generally speaking, a k -value of 3 or 5 can work well across these maps. For outliers, we add them to all other clusters to avoid action sets consisting of only one action, which we find can largely stabilize training across the benchmark. The number of clusters k were chosen based on SMAC domain knowledge. Therefore, we set $k = 3$ on maps with homogeneous enemies (potentially encoding roles with movement towards, away from enemies, and attack) and $k = 5$ on maps with heterogeneous enemies (potentially encoding roles with movement and different types of enemies). Therefore, the user priors on clusters may be usefully imposed by incorporating expert knowledge about the domain and the specific tasks to leverage the clustering process for improved performance. Despite the potential unreliability of k -means clustering with random initialization, we observed that the performance of RODE remained consistent across 8 different seeds, exhibiting low variance, as seen in Fig. 6.12.

To further understand how different values of k affect the learning performance, we take one super hard map, **6h_vs_8z**, as an example. When $k=3$ or 5, the clustering method (after adding outliers to all other clusters) gives three role action spaces – **Move Eastward & Northward**, **Attack**, and **Move Westward & Southward**. Similar to the case of **corridor** as discussed in Section 6.4.3, these roles can improve exploration and thus boost the performance. However, when $k=7$, all move actions become outliers. The result is that most role action spaces become the original full action space. Such a decomposition over action space does not reduce the search space, and thus the performance is close to that of QMIX. To avoid such complicated interactions between the selection of k values and the way of dealing

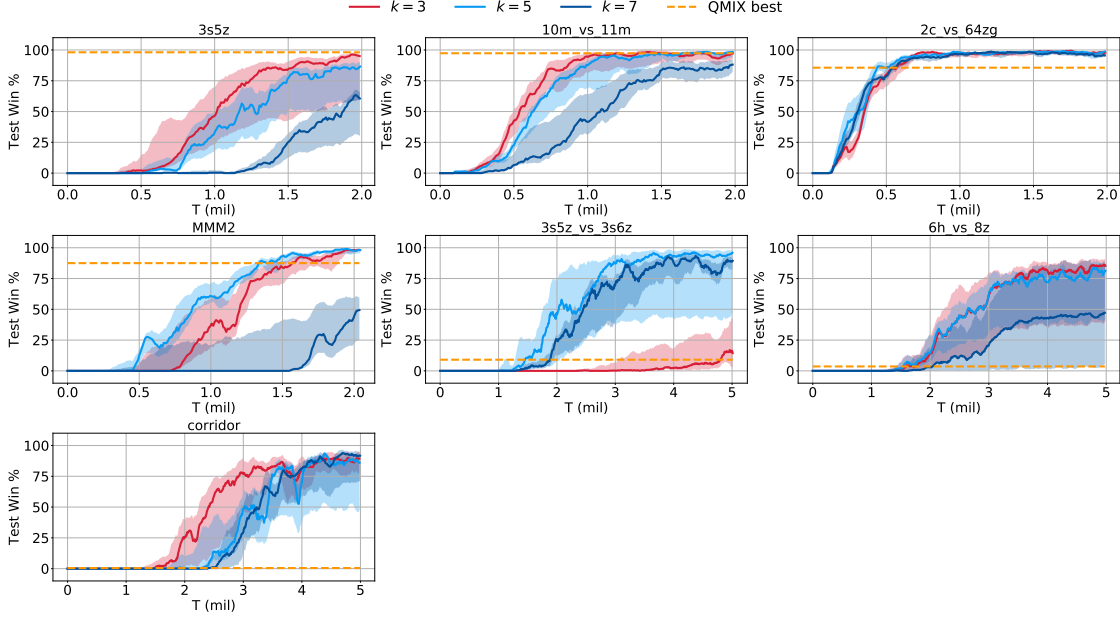


Figure 6.12: RODE with different k values.

with outliers, we propose to use X -means when clustering action representations, as we discuss in detail in the next section.

6.5.2 Other Clustering Algorithms

In this section, we investigate how to avoid predefining k values. X -means [Pelleg et al., 2000] improves k -means by automatically determining the number of clusters. We use *minimum noiseless description length* as the splitting criterion for X -means and test RODE using X -means on all the 14 maps. The results are shown in Fig. 6.13. We see that the results are comparable to k -means with predefined k values. We thus recommend X -means when facing a new task, while for tasks that users know the number of roles, k -means can be more accurate.

Additionally, we can use density-based clustering algorithms to avoid the predefinition of k , like DBSCAN [Ester et al., 1996]. In practice, we find that DBSCAN can generate similar role action spaces to X -means. Since the clustering algorithm is used only to split the action spaces, RODE using DBSCAN clustering is expected to have similar learning performance to X -means.

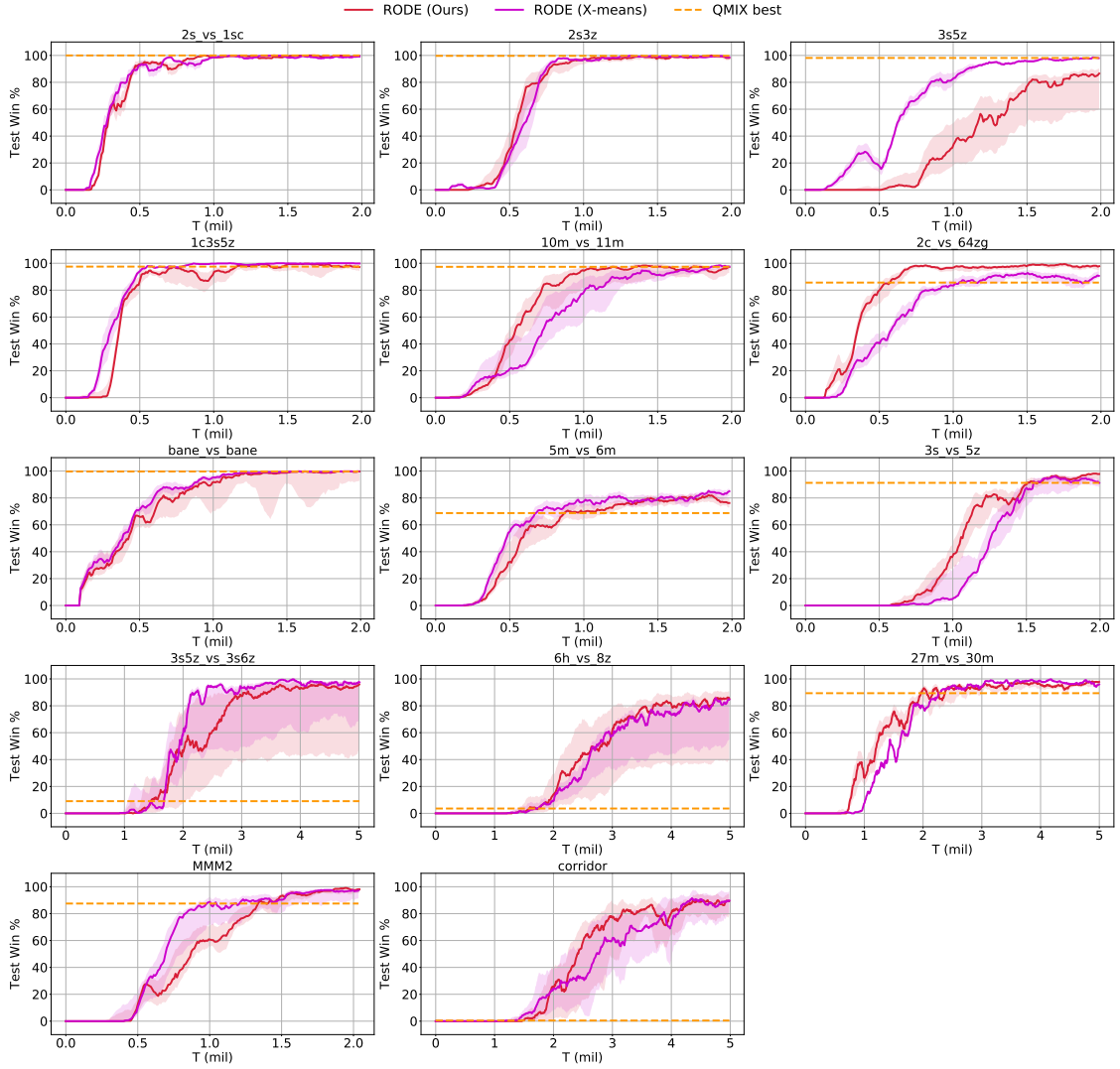


Figure 6.13: Comparisons between RODE using X-means clustering and RODE using k -means clustering, where $k = 3$ for maps with homogeneous enemies and $k = 5$ for maps with heterogeneous enemies. X-means does not require predefining the number of clusters.

6.6 Conclusion and Future Work

Coming up with a set of roles that can effectively decompose the task is a long standing problem preventing role-based learning from realizing scalability. Instead of learning roles from scratch, in this chapter, we find that role discovery becomes much easier if we first decompose joint action spaces according to action effects. With a specially designed hierarchical learning framework, we achieve efficient learning over these factored action spaces. We believe that the scalability and

transferability provided by our method are crucial in building flexible and general-purpose multi-agent systems.

Future Work

Role-based learning can be a very promising direction for future research in other complex cooperative MARL domains such as Google Research Football [GRF, Kurach et al., 2020]. In GRF, which simulates the dynamics of a football game, agents must learn individual skills and how to cooperate as part of a team effectively. Role-based learning can significantly enhance this process. By assigning specific roles to different agents — akin to positions in a real football team — each agent can specialize in specific skills and behaviors that are optimal for their role. For instance, a ‘defender’ agent would focus on learning strategies to block opponent advances, while a ‘striker’ agent would concentrate on scoring tactics.

Part II(B) : Cooperative Policy-Based Multi-Agent RL

Abstract

We next focus on discussing policy-based cooperative MARL methods. Although value-based MARL methods have significantly contributed to progressing the state-of-the-art in cooperative settings, they present stability and convergence challenges [Wang et al., 2021], especially in continuous action spaces. There has been a lot of progress in developing multi-agent policy gradient methods (MAPG) using the CTDE paradigm, where centralized critics are learned alongside individual agent policies with access to only local trajectories. Despite the recent progress with MADDPG [Lowe et al., 2017], COMA [Foerster et al., 2018] and DOP [Wang et al., 2020e], multi-agent policy gradients still underperform value-based methods by a good margin on the popular multi-agent SMAC benchmark suite [Samvelyan et al., 2019]. In this chapter, we demonstrate that, despite its various theoretical shortcomings, Independent PPO (IPPO), a form of independent learning in which each agent simply estimates its local value function, can perform just as well as or better than state-of-the-art joint learning approaches on SMAC with little hyperparameter tuning. We also compare IPPO to several variants; the results suggest that IPPO’s strong performance may be due to its robustness to some forms of environment non-stationarity. These results suggest it might be fruitful to improve independent learning approaches such as IPPO further.

7

Independent PPO for SMAC

Contents

7.1	Introduction	133
7.2	Related Work	136
7.3	Background	137
7.3.1	Trust Region Policy Optimization	138
7.3.2	Independent Actor-Critic	138
7.4	Independent PPO	139
7.4.1	Value Clipping	140
7.4.2	Learning Architecture	141
7.5	Empirical Results	141
7.5.1	IPPO Performance	142
7.5.2	Role of Centralised Value Functions	143
7.5.3	Ablation Studies	143
7.6	Discussion	145
7.7	Conclusion and Future Work	148

7.1 Introduction

Our discussion in previous chapters focused on cooperative multi-agent reinforcement learning (MARL), which has shown considerable promise in solving tasks that can be described as a Dec-POMDP [Oliehoek et al., 2016]. Agents optimize a single scalar team reward signal in a partially observable environment while choosing

actions based only on their own local action-observation histories [Panait and Luke, 2005, Busoniu et al., 2008, Tuyls and Weiss, 2012].

Independent learning (IL) decomposes an n -agent MARL problem into n decentralized single-agent problems where all other agents are treated as part of the environment and learning policies that condition only on an agent’s local observation history. While easy to distribute and decentralize by construction, IL suffers from a variety of theoretical limitations that may result in learning instabilities or suboptimal performance observed in practice [Tan, 1993, Foerster et al., 2018]. Firstly, the presence of other learning and exploring agents renders the resulting environment non-stationary from the given agent’s perspective, forfeiting convergence guarantees. Secondly, independent learners are not always able to distinguish environment stochasticity from another agent’s exploration, making them unable to learn optimal policies in some environments [Claus and Boutilier, 1998].

In fact, decentralized policies need not be learned in a decentralized fashion. For safety and efficiency reasons, MARL training frequently takes place centrally in a laboratory or in simulation, allowing agents access to each other’s observations during training as well as otherwise unobservable extra-state information. Centralized training allows training of a single joint policy for all agents that conditions on the joint observations and extra state information. While centralized joint learning reduces or removes issues surrounding partial observability and environment non-stationarity, it must cope with joint action spaces that grow exponentially with respect to the number of agents. Importantly, vanilla joint policies are not inherently decentralisable, and naive policy distillation approaches are often ineffective [Böhmer et al., 2019]. Joint learning does not immediately address the multi-agent credit assignment problem either, i.e., an agent inferring how much their own actions contribute to the team’s success or failure.

As discussed in the past few chapters, recent research has focused on algorithms that can exploit the benefits of combining centralized training with decentralized execution [Oliehoek et al., 2016, CTDE] while mitigating some of the pathologies

of vanilla centralized joint learning. A particularly successful line of research uses value factorization of the joint Q -value function in order to reduce the size of the joint action space [VDN, QMIX, FacMADDPG, Sunehag et al., 2018, Rashid et al., 2020b, de Witt et al., 2020]. However, from Chapter 5, we know that value factorization is prone to a learning pathology called relative overgeneralization [RO, Wei and Luke, 2016], where policies erroneously converge to a suboptimal joint action. Relative overgeneralization commonly arises when multiple agents must coordinate their actions but receive negative rewards if only a subset of them do so. In this case, random occurrences of such successful coordination events are a needle in a haystack, and monotonic value factorizations may fail to represent the non-monotonic team reward function [Böhmer et al., 2020]. A variety of recent approaches employ joint optimization of, or transfer learning between, factored and unrestricted joint value functions in order to overcome the representational limitations of factored value functions, with varying success [Böhmer et al., 2019, Son et al., 2019, Rashid et al., 2020a].

In this chapter, we present empirical evidence for Independent PPO (IPPO), a multi-agent variant of *proximal policy optimization* [Schulman et al., 2017b], that shows IPPO matches or outperforms state-of-the-art MARL CTDE algorithms such as QMIX [Rashid et al., 2020b] or MAVEN [Mahajan et al., 2019] on multiple hard maps on SMAC [Samvelyan et al., 2019] with the original reward function. Please refer to Section 5.9.3 for a detailed discussion on different reward functions in SMAC. We hypothesize that algorithmic choices made by PPO, such as policy clipping, may help mitigate some forms of environment non-stationarity. We empirically show that the effect of policy clipping in IPPO cannot be emulated by lowering the effective learning rate alone. Similarly, we hypothesize that coordination limitations found in certain matrix games [Claus and Boutilier, 1998] rarely occur in complex extended games, thus allowing independent learning to perform well in such settings while avoiding representational limitations introduced in factored value function approaches. Similar to Chapter 5, we note that most SMAC scenarios with the

original reward function (i.e., `reward-only-positive = True`) are not conducive to the study of relative overgeneralization. For example, in an SMAC environment with positive-only rewards, agents do not have to strictly coordinate at a single time step to attack enemy units as the reward function is strictly enemy-oriented, and there is no team penalty for dying ally units.

7.2 Related Work

Actor-critic algorithms may benefit from learning centralized joint critics alongside decentralized policies [Central-V, Foerster et al., 2018]. As the critics are not needed during execution, this approach is inherently decentralizable. COMA [Foerster et al., 2018] extends this approach with a counterfactual multi-agent critic baseline based on temporal difference errors [Tumer and Agogino, 2007] in order to facilitate multi-agent credit assignment. Value Decomposition Networks [VDN, Sunehag et al., 2018] decompose joint state-action value functions into sums of decentralized utility functions that can be used during greedy execution. QMIX [Rashid et al., 2020b] extends this additive decomposition to arbitrary centralized monotonic mixing networks. Value factorization has also been successfully transferred to continuous action spaces and combined with actor-critic approaches [COMIX, FacMADDPG, de Witt et al., 2020].

Independent learning (IL) dates back to the early days of multi-agent reinforcement learning [IQL, Tan, 1993]. While initially tabular, IL algorithms using neural networks as function approximators were subsequently developed [IAC, Deep IQL, Rashid et al., 2020b, Foerster et al., 2018]. The question of whether independent agents are better at learning cooperative behavior than multiple *complete observing* agents [Whitehead, 1991] was first comprehensively investigated by Tan [1993], who concludes that while sharing policies or experience was generally advantageous, sharing extra sensory information could also negatively interfere with learning, e.g., by expanding the state space.

Trust region optimization for reinforcement learning was popularised by TRPO [Schulman et al., 2017a], which implements iterative guaranteed monotonic improvements. PPO [Schulman et al., 2017b] preserves many of TRPO’s empirical benefits while trading in theoretical guarantees for computational speed. PPO’s policy update regularisation using clipped probability ratios has been subject to much scrutiny in single-agent settings: Truly PPO [Wang et al., 2020f] suggests modifications in order to ensure guaranteed monotonic improvements with little computational overhead. Recent work confirms that PPO’s performance may crucially depend on the choice of policy surrogate objective [Hsu et al., 2020]. In addition, code-level design choices have been shown to significantly impact PPO performance in practice [Engstrom et al., 2020].

While popular for single-agent tasks, PPO has only recently been applied to decentralized cooperative multi-agent tasks. Concurrent work proposes MAPPO [Yu et al., 2022], an actor-critic multi-agent algorithm based on PPO. Like IPPO, MAPPO employs weight sharing between each agent’s critics. In contrast to IPPO, MAPPO uses a centralized value function that conditions on the full state (or concatenation of agent observations, if the full state is unavailable). In this chapter, we mainly focus on the difference in local observation based critics vs critics conditioned on joint-observations or state. We show that IPPO substantially outperforms MAPPO on a variety of hard SMAC maps and even beats other state-of-the-art algorithms, such as QMIX, on some of these, with only modest hyperparameter tuning.

7.3 Background

Similar to previous chapters, we consider a fully cooperative multi-agent task formalized by Dec-POMDPs [Oliehoek et al., 2016] under the CTDE framework. We next discuss the popular single-agent Trust Region Policy Optimization (TRPO) [Schulman et al., 2017a] method and Independent Actor-Critic (IAC) [Foerster et al., 2018].

7.3.1 Trust Region Policy Optimization

TRPO [Schulman et al., 2017a] is a class of policy-gradient methods that restricts the update of policies to within the trust region of the behavior policy by enforcing a KL divergence constraint on policy update at each iteration. Formally, TRPO optimizes the following:

$$\max_{\theta} \quad \mathbb{E}_{s_t, u_t} \left[\frac{\pi_{\theta}(u_t|s_t)}{\pi_{\theta_{old}}(u_t|s_t)} A(s_t, u_t) \right] \quad (7.1)$$

$$\text{subject to} \quad \mathbb{E}_{s_t, u_t} [\text{KL}(\pi_{\theta_{old}}, \pi_{\theta})] \leq \delta, \quad (7.2)$$

where θ_{old} are the policy parameters before the update and $A(s_t, u_t)$ is an approximation of advantage function. The formulation is computationally expensive as it requires the computation of multiple Hessian-vector products for nonlinear conjugate gradients when approximating the KL constraint. To resolve this, Proximal Policy Optimization [PPO, Schulman et al., 2017b] approximates the trust region constraints by policy ratio clippings, i.e. the policy loss becomes:

$$\mathcal{L}(\theta) = \mathbb{E}_{s_t, u_t \sim \pi_{\theta_{old}}} \left[\min \left(\frac{\pi_{\theta}(u_t|s_t)}{\pi_{\theta_{old}}(u_t|s_t)} A(s_t, u_t), \right. \right. \\ \left. \left. \text{clip} \left(\frac{\pi_{\theta}(u_t|s_t)}{\pi_{\theta_{old}}(u_t|s_t)}, 1 - \epsilon, 1 + \epsilon \right) A(s_t, u_t) \right) \right]. \quad (7.3)$$

PPO works by restricting the new policy update from *diverging far from the old policy* by controlling the hyperparameter ϵ . When the advantage is positive, the loss does not allow the objective to increase beyond $(1 + \epsilon)A(s_t, u_t)$. Similarly, in the case of negative advantage, the objective is clipped at $(1 - \epsilon)A(s_t, u_t)$. Therefore, the hyperparameter ϵ denotes the extent to which the new policy can deviate from the previous one, establishing a trust region.

7.3.2 Independent Actor-Critic

Independent Actor-Critic (IAC) leverages single-agent actor-critic methods such as A2C [Konda and Tsitsiklis, 2000] to learn independent policies for each agent. It is very common for these independent agent policies to share NN parameters for

efficiency purposes [Foerster et al., 2018]. Each agent can still have different policies as the input can be concatenated with agent IDs. Terry et al. [2020] provides a detailed discussion on the relative performance of different agent indication methods in deep MARL. The gradient of policy π^a parameterized with shared NN parameters θ and an action-value critic Q^a parameterized with shared NN parameters ϕ is:

$$\begin{aligned}\nabla\eta(\theta) &= \mathbb{E}_{\pi^a} [Q^a(o^a, u^a; \phi) \nabla_{\theta} \log \pi^a(u^a|o^a; \theta)] \\ \theta &\leftarrow \theta + \alpha \nabla\eta(\theta).\end{aligned}$$

We can also use the observation value function $V^a(o^a)$ as a critic instead of the action-value function shown above. Since the critic is only used during training, it can condition on available additional information during training. It can be either fully *decentralized* i.e. conditioned only on local observation (or observation history) of an agent, or *centralized* i.e. additionally conditioned on either joint observations (or joint observation histories) of all agents, or state information, or combination of these. Lyu et al. [2023] showcase that having additional state information may not always be beneficial for learning as it may add noise and lead to higher variance in the policy gradient during training. Please refer to Lyu et al. [2023] for a detailed discussion and theoretical analysis of centralized critics.

7.4 Independent PPO

In this chapter, we use PPO to learn decentralized policies π^a for agents with individual policy clipping where each agent’s independent policy updates are clipped based on the objective defined in Equation 7.3. We consider a variant of the advantage function based on independent learning, where each agent a learns a local observation-based critic $V_{\phi}(o_t^a)$ parameterized by ϕ using Generalized Advantage Estimation (GAE) [Schulman et al., 2015] with discount factor $\gamma = 0.99$ and $\lambda = 0.95$. The network parameters ϕ, θ are shared across critics and actors, respectively. We

also add an entropy regularization term [Mnih et al., 2016] to the final policy loss. For each agent a , we have its advantage estimation as follows:

$$A_t^a = \sum_{l=0}^h (\gamma\lambda)^l \delta_{t+l}^a, \quad (7.4)$$

where $\delta_t^a = r_t(o_t^a, u_t^a) + \gamma V_\phi(o_{t+1}^a) - V_\phi(o_t^a)$ is the TD error at time step t and we vary h as shown in Table 7.2 (marked as steps num). We use the team reward $r_t(s_t, \mathbf{u}_t)$ to approximate $r_t(o_t^a, u_t^a)$. The final policy loss for each agent a becomes:

$$\mathcal{L}^a(\theta) = \mathbb{E}_{o_t^a, u_t^a} \left[\min \left(\frac{\pi_\theta(u_t^a | o_t^a)}{\pi_{\theta_{old}}(u_t^a | o_t^a)} A_t^a, \text{clip} \left(\frac{\pi_\theta(u_t^a | o_t^a)}{\pi_{\theta_{old}}(u_t^a | o_t^a)}, 1 - \epsilon, 1 + \epsilon \right) A_t^a \right) \right]. \quad (7.5)$$

7.4.1 Value Clipping

In addition to clipping the policy updates, our method also uses value clipping to restrict the update of the critic function for each agent a to be smaller than ϵ as proposed by [Schulman et al., 2015, GAE] using:

$$\mathcal{L}^a(\phi) = \mathbb{E}_{o_t^a} \left[\min \left\{ \left(V_\phi(o_t^a) - \hat{V}_t^a \right)^2, \right. \right. \\ \left. \left. \left(V_{\phi_{old}}(o_t^a) + \text{clip} \left(V_\phi(o_t^a) - V_{\phi_{old}}(o_t^a), -\epsilon, +\epsilon \right) - \hat{V}_t^a \right)^2 \right\} \right], \quad (7.6)$$

where ϕ_{old} are old parameters before the update and $\hat{V}_t^a = \sum_{l=0}^{\infty} \gamma^l r_{t+l}$. The update equation restricts the update of the value function to within the trust region and, therefore, helps us to avoid overfitting to the most recent batch of data. For each agent, the overall learning loss becomes:

$$\mathcal{L}(\theta, \phi) = \sum_{a=1}^n \mathcal{L}^a(\theta) + \lambda_{\text{critic}} \mathcal{L}^a(\phi) + \lambda_{\text{entropy}} \mathcal{H}(\pi^a), \quad (7.7)$$

where $\mathcal{H}(\pi^a)$ denotes the entropy of policy π^a , λ_{critic} and λ_{entropy} vary as shown in Table 7.2. In Section 7.5.3, we compare IPPO to a baseline in which both policy and value clipping are ablated, yielding a variant of IAC [Foerster et al., 2018].

7.4.2 Learning Architecture

We use a variance scaling initializer with truncated normal distribution [He et al., 2015] with $\text{scale} = 2.0$ to initialize the parameters of our policy and value function NNs that have been shown to work well with ReLU activations. The input to our NN comprises of stacked observations for the past few times steps (marked as frames in Table 7.2), which is passed through three conv1d convolution layers with varying number of filters (marked as net arch in Table 7.2) and fixed $\text{kernel_size} = 3$, $\text{strides} = (2, 1, 1)$, $\text{padding} = (\text{same}, \text{valid}, \text{valid})$ for each conv layer, followed by two MLP layers with (256, 128) units and ReLU activations (see Table 7.2 for more details). We use gradient clipping to restrict the norm of the gradient to be less than 0.5 and normalize the advantage by subtracting the mean and dividing by the standard deviation once before training - we find that advantage normalization by minibatch yields worse performance. We use a discount factor of $\gamma = 0.99$, a learning rate of 10^{-4} , and clipping parameter $\epsilon = 0.2$.

7.5 Empirical Results

In this section, we evaluate Independent PPO (IPPO) on 16 maps from the StarCraft Multi-Agent Challenge [SMAC, Samvelyan et al., 2019] maps, performing only mild hyperparameter tuning for individual maps (see Table 7.2). SMAC consists of a diverse set of StarCraft II [SCII, Vinyals et al., 2017] unit micromanagement tasks of varying difficulty, where a collaborative team of SCII units needs to defeat an enemy team of units controlled by the built-in AI. SCII units consist of a balanced set of both melee and long-range attack units (plus healing units, *medivacs*), and winning strategies often entail precisely coordinated unit movements strategies, including *kiting* [Rashid et al., 2020b], in order to gain positional advantages. For all our experiments, we use game version 4.6 and select the hardest AI difficulty level.

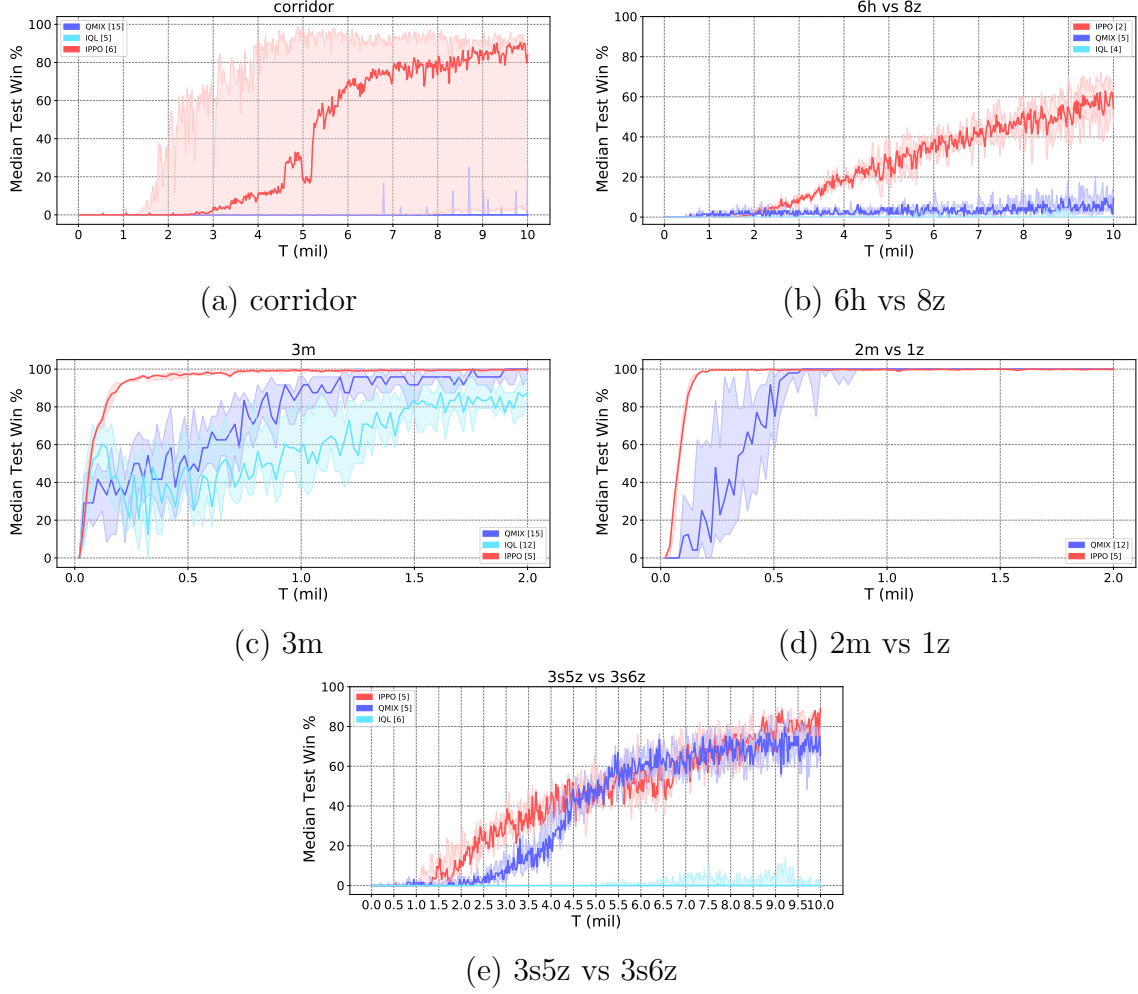


Figure 7.1: Results on select SMAC maps, comparing IPPO to QMIX and IQL. Uncertainty regions depict $[0.25, 0.75]$ confidence intervals of the median test win rate.

7.5.1 IPPO Performance

We benchmark IPPO and a number of ablations on a number of SMAC maps (see Figure 7.1). Despite its simplicity, we find that IPPO significantly outperforms two strong MARL baselines that exploit centralized state during learning, namely MAPPO (see Table 7.1) and QMIX on three difficult maps, namely *3s5z vs 3s6z*, *corridor* and *6h vs 8z*. On *corridor*, IPPO even exceeds the particularly strong performance of MAVEN [Mahajan et al., 2019]. In addition, IPPO outperforms QMIX on easy SMAC maps *2m vs 1z* and *3m* and is competitive on a few more (see Appendix D.1). IPPO is also able to learn maps renowned for their difficulty,

Map	corridor	6h_vs_8z	3s5z_vs_3s6z	3m	2m_vs_1z
IPPO	80	60	90	100	100
MAPPO	0	8.95	80	99.875	100

Table 7.1: Median win test rates on select SMAC maps at 10M steps, comparing IPPO to MAPPO results reported by Yu et al. [2022].

such as *3s5z vs 3s6z* and *6h vs 8z*. We also find that IPPO performance is superior to that of IQL, the other independent learning algorithm tested [Rashid et al., 2020b], and it is also generally more stable across a large variety of maps. Further results may be found in Appendix D.1.

7.5.2 Role of Centralised Value Functions

IPPO exploits centralized training solely through sharing network parameters between agent critics. However, SMAC additionally provides full-state information that lifts partial observability of the environment during learning. While at first glance, conditioning the critic on full-state information should help simply due to it containing more information, already early empirical research in independent vs centralized learning concludes that, perhaps counter-intuitively, this is not necessarily the case as the larger size of the full state and/or it including information irrelevant to the task may adversely impact learning [Lyu et al., 2023]. Nevertheless, previous work on SMAC has reported higher performance when the centralized state was being exploited during learning [Rashid et al., 2020b, Yu et al., 2022]. A direct comparison (see Figure 7.2) shows that replacing IPPO’s local critics with parameter-sharing critics conditioning on full state information performs substantially worse on a selection of hard SMAC maps, calibrating our implementation with centralized value function performance stated in concurrent work [MAPPO, Yu et al., 2022].

7.5.3 Ablation Studies

To understand better why IPPO performs outperforms other independent learning algorithms, including Independent Actor-Critic (IAC) [Foerster et al., 2018], we

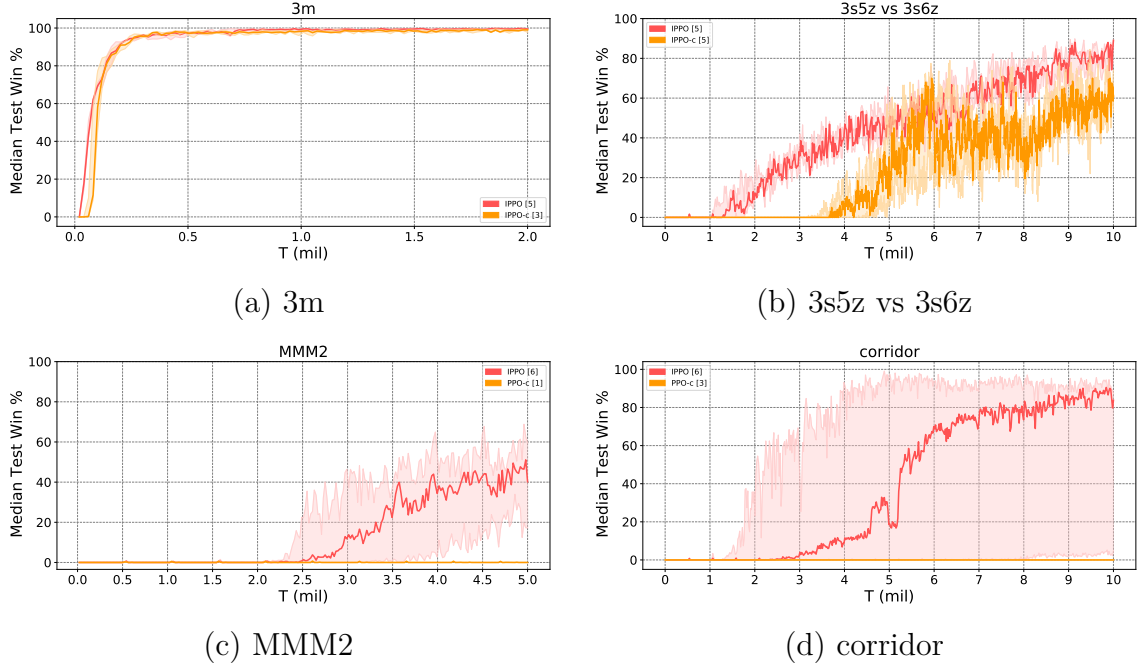


Figure 7.2: Results on select SMAC maps comparing IPPO and MAPPO.

investigate the performance of a number of ablations (see Figure 7.3). We find that IPPO with neither policy nor value clipping, which corresponds to a variant of IAC [Foerster et al., 2018], performs poorly across all six SMAC maps studied. Furthermore, we find that policy clipping is essential to performance.

In conjunction with policy clipping, we find that value clipping improves performance on some maps (e.g., *corridor* and *MMM2*). The selective usefulness of value function clipping is in line with empirical observations in single-agent settings, which suggest that value function clipping may only be advantageous if the critic estimator suffers from high variance [Kostrikov].

An alternative explanation for our findings is that IPPO’s policy clipping objective reduces the effective policy learning rate, thus stabilizing learning. We test this hypothesis by reducing the learning rate for IAC. This does not yield the anticipated performance gains relative to IPPO (see Figure 7.3).

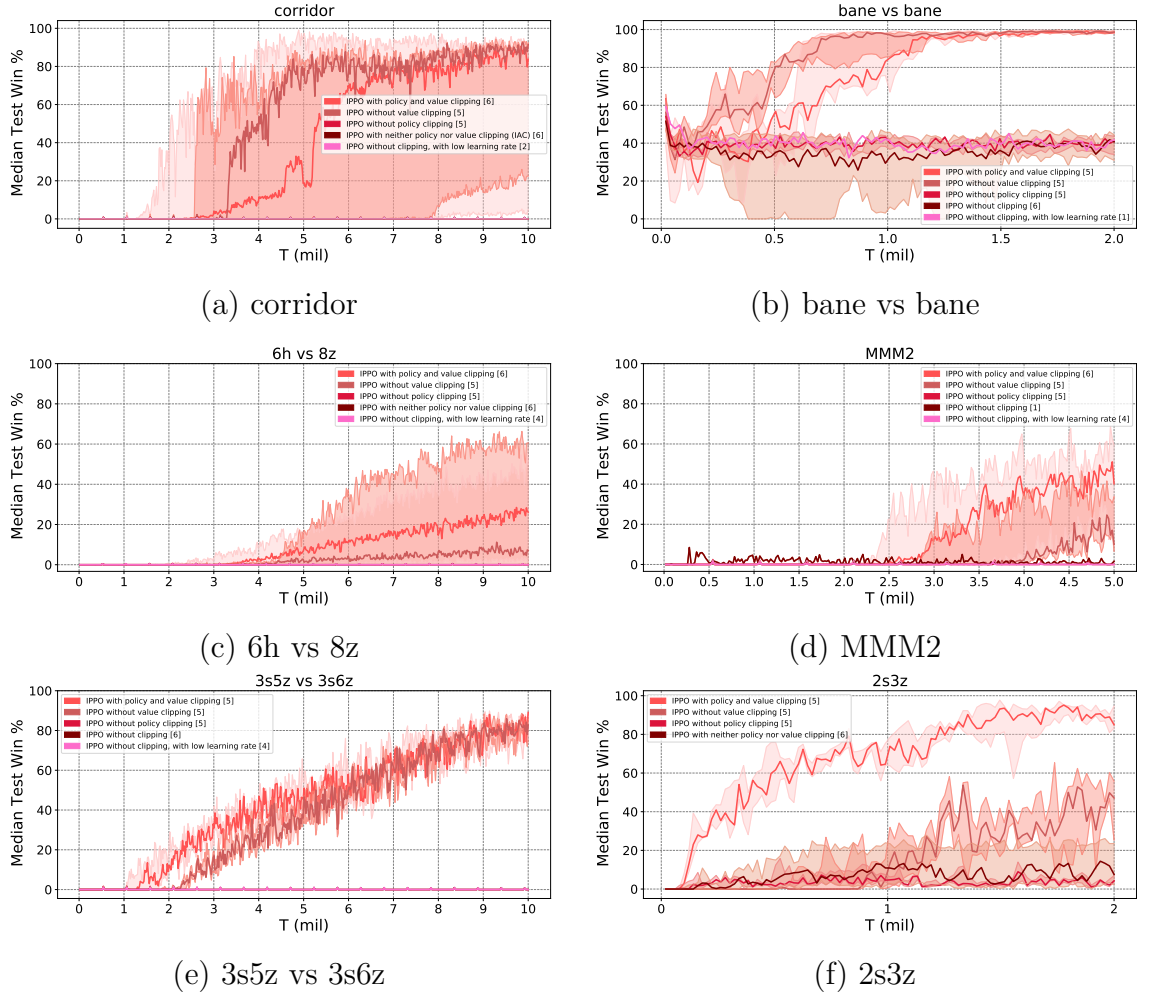


Figure 7.3: Ablation study for IPPO with different combinations of policy and value clipping.

7.6 Discussion

Our empirical results give rise to a number of interesting insights about the utility of IPPO in relation to other state-of-the-art MARL algorithms, in particular those relying on value factorization methods, as well as the role of SMAC in future MARL research.

Centralized State: IPPO outperforms a number of algorithms using a centralized state, including QMIX, MAPPO, and MAVEN, as well as both independent learning algorithms IAC and IQL, on a number of both hard and easy SMAC maps. Lyu et al. [2023] also note that exploiting state information during centralized training

map name	critic coef	entropy coef	frames	lr	mini epochs	mini batch	norm input	steps num	type	net arch
10m vs 11m	2	0.005	4	1e-4	4	2560	True	128	cnn	[256, 512, 1024]
1c3s5z	2	0.005	4	1e-4	1	4096	False	128	cnn	[64, 128, 256]
27 vs 30m	2	0.005	4	1e-4	1	2560	False	64	cnn	[64, 128, 256]
2c vs 64zg	2	0.005	4	1e-4	4	512	False	64	cnn	[64, 128, 256]
2m vs 1z	1	0.005	1	1e-4	4	1024	False	128	mlp	[256, 128]
2s3z	1	0.001	1	5e-4	4	1536	True	128	cnn	[64, 128, 256]
2s vs 1sc	2	0.005	4	1e-4	1	2048	False	128	cnn	[64, 128, 256]
3m	1	0.001	1	5e-4	4	1536	True	128	mlp	[256, 128]
3s5z	2	0.005	4	1e-4	1	4096	False	128	cnn	[64, 128, 256]
3s5z vs 3s6z	2	0.005	4	1e-4	1	4096	False	128	cnn	[64, 128, 256]
3s vs 5z	2	0.005	4	1e-4	1	1536	False	128	cnn	[64, 128, 256]
5m vs 6m	2	0.005	4	1e-4	1	2560	False	128	cnn	[64, 128, 256]
6h vs 8z	2	0.005	4	1e-4	1	3072	False	128	cnn	[64, 128, 256]
MMM	2	0.005	4	1e-4	1	2560	False	64	cnn	[64, 128, 256]
MMM2	2	0.005	4	1e-4	1	2560	False	64	cnn	[64, 128, 256]
bane vs bane	2	0.005	4	1e-4	1	3072	False	128	cnn	[64, 128, 256]
corridor	2	0.005	4	1e-4	1	3072	False	128	cnn	[64, 128, 256]

Table 7.2: IPPO hyperparameters across different SMAC maps. Fixed hyperparameters: $\gamma = 0.99$, $\lambda = 0.95$, $e_{clip} = 0.2$, $grad\ norm = 0.5$ and $\epsilon = 0.2$. We do not employ any learning rate schedule, and always normalize the advantage function. We fix the number of actors to 8. *Net arch* indicates successive channels across layers for *conv1d* architecture, and number of activations for *mlp*. All convolutions have stride 3.

might lead to poor performance due to additional noise and higher variance.

Policy Clipping: PPO’s optimization objective, in particular policy clipping, is crucial to performance in cooperative deep multi-agent reinforcement learning on popular benchmark environment SMAC. According to our empirical ablation studies, this effect cannot be explained through a decrease in the effective learning rate alone. Given IPPO’s improved learning stability over IAC and IQL, it seems that its (approximate) surrogate objective might mitigate certain forms of environment non-stationarity that other independent learning algorithms are prone to, e.g., by suppressing updates catastrophic to performance.

Recently, Sun et al. [2022] have theoretically demonstrated that the effectiveness of IPPO is primarily due to applying a trust region constraint, which involves

bounding the ratios of *independent* policies in relation to the number of agents. They first point out that bounding the *joint* ratios, i.e., computing ratios between joint policies, enjoys monotonic policy improvement guarantees. The authors then show that enforcing the trust region constraint over all decentralized policies in centralized training (via parameter sharing) tackles non-stationarity challenges in learning decentralized policies and can enjoy the same performance guarantees as joint ratios. Additionally, they observe that the number of agents significantly influences the extent of the clipping range. This is because the number of agents collectively plays a crucial role in defining the size of the centralized trust region.

SMAC: We show that, at least on some hard SMAC maps, the currently best performance is achieved by an algorithm that does not require the learning of either joint or centralized value functions and does not exploit central state information in any other way. This implies that the fundamental obstacles to independent learning methods posed by certain pathological matrix games [Claus and Boutilier, 1998, Mahajan et al., 2019] are not present in such SMAC tasks. A possible explanation is that the sequential nature of SMAC tasks allows the decomposition of tricky simultaneous coordination tasks into a temporal sequence of easier coordination tasks. Similar to Chapter 5, we note that the common learning pathology of relative overgeneralization does not seem to be present in these SMAC tasks with the original positive-only reward function. On the other hand, as noted from Section 5.9.2 and Figure D.1, IPPO and value factorization algorithms like QMIX, are in principle prone to relative overgeneralization on predator-prey tasks with high penalties where simultaneous coordination is required. We hypothesize that in a SMAC environment with positive-only rewards, agents do not have to strictly coordinate at a single time step to attack enemy units as the reward function is strictly enemy-oriented, and there is no team penalty for dying ally units. In fact, Ellis et al. [2022] demonstrates that SMAC lacks stochasticity and partial observability to require complex policies which need to condition on the observations.

They show that a policy that relies solely on the time step can successfully attain significant winning percentages in numerous SMAC situations.

7.7 Conclusion and Future Work

In this chapter, we studied the empirical performance of Independent PPO (IPPO), a multi-agent variant of the popular PPO algorithm. Despite the purported limitations of independent learning for cooperative MARL tasks, we found that IPPO performs competitively on a range of state-of-the-art benchmark tasks, outperforming state-of-the-art value factorization methods on some maps. A number of ablation studies indicated that PPO’s policy clipping objective is crucial to performance and that the value of central state information in SMAC is unclear. Furthermore, our results raise the question of whether relative overgeneralization really matters in practice: despite the diversity of its scenarios, it does not seem to be an obstacle to independent learning in SMAC with the original reward function. As a consequence, we suggest that the MARL community revisit the question of whether relative overgeneralization really matters in practice.

Future Work

One possible direction of future research to improve policy-based MARL approaches is to condition the learned policy on richer information than just local observation of the agent. As discussed many times in this thesis, under CTDE settings, extra-state information is often available during training. However, at execution, each agent only has access to its own observations, and the action distribution of the agents must be factored as a product of their individual distributions. Therefore, in policy gradient approaches, the separation of observations at run time creates an issue of information that can only be used during training, i.e., privileged information (PI).

Future methods can train agents to predict the global state information from local action-observation histories during training. However, this may not be feasible to learn, as the global state contains a lot of irrelevant information about all agents

that would be difficult to reproduce for the agent. Therefore, agents can benefit from learning and embedding the agent-specific state representations that contain relevant information about other agents during training, and policies can be conditioned on such agent-specific representations. To enable decentralized execution, we can infer learned agent-specific state representation from their local action-observation trajectories using mutual information. As a result, agents can now condition on more information than just local histories during execution. This method should improve sample efficiency since agents do not need to reason about the entire joint state, which may contain irrelevant information.

8

Afterword

Contents

8.1 Broader Impact	152
-------------------------------------	------------

In this thesis, we addressed challenges around how to develop agents that can learn to coordinate with other agents under strict penalties for miscoordination and efficiently explore their environments in single-agent, multi-agent, and procedurally-generated sparse reward settings. The proposed methods improve the sample efficiency of deep (multi-agent) reinforcement learning methods and enable zero-shot generalization to harder tasks.

Chapter 3 leverages semantic representations of the CLIP model for novelty-based exploration in RL via intrinsic rewards. Central to this is the idea that language abstractions incorporated within foundation models such as CLIP can be useful for semantic exploration in RL, even in grid-based environments. We showcased that CLIP-based intrinsic rewards achieve higher performance and significantly faster learning speed and sample efficiency on multiple tasks in a procedurally generated grid-world environment.

Chapter 5 tackles the issue of relative overgeneralization (RO) in multi-agent

tasks requiring significant coordination from many agents at a single time step. We proposed UneVEn, which improves MARL exploration in such tasks by leveraging the knowledge from tasks related to the target task but simpler and easier to solve, thereby increasing the set of solvable tasks until we reach our target task. Our experiments show that UneVEn significantly outperforms state-of-the-art (SOTA) value-based MARL methods on tasks exhibiting severe RO. We also shed light on the different reward functions in Starcraft Multi-Agent Challenge (SMAC) and show that the default positive reward function in SMAC does not exhibit RO.

Chapter 6 proposes RODE, which improves multi-agent exploration by clustering the action space into different roles such that each role corresponds to a different sub-task and learning a hierarchical policy to select the roles. Each role-based policy only needs to be explored in a restricted action space, making it easier to learn. For example, football players have many roles, such as defending, attacking, goalkeeping, etc. These roles are transferable across tasks, and RODE beats all other SOTA methods on SMAC tasks, which requires agents to learn complicated cooperation strategies.

Lastly, Chapter 7 focused on investigating independent PPO for SMAC tasks and showcased that even independent learning methods can outperform SOTA MARL methods on a number of SC2 tasks with the default positive reward function. This reaffirms our experiments in Chapter 5 and sheds light on the limitations of the default reward function in SMAC. Ellis et al. [2022] demonstrates that SMAC lacks stochasticity and partial observability to require complex policies which condition on the observation. They propose SMACv2, which is procedurally generated and requires generalization to unseen settings.

8.1 Broader Impact

As this thesis concludes, reflecting on the broader implications of our findings within the realm of Reinforcement Learning (RL) is essential. Enabling efficient semantic exploration through natural language can enhance the sample efficiency of RL

methods and open up possibilities for more intuitive and human-like AI interactions in various domains. An example is complex visual gaming scenarios, where an agent must navigate and strategize in a dynamic, intricate virtual world. By leveraging natural language to understand and interact with the game environment, these agents can make more informed decisions, adapt to changing situations, and learn to play the game more effectively and efficiently. Such advancements could revolutionize game development, enabling rapid prototyping and more immersive gaming experiences by allowing developers to incorporate AI agents that can learn and adapt in real-time. Similarly, developing agents capable of efficiently collaborating can be helpful for many domains, such as autonomous driving cars to improve traffic management and safety, enabling more efficient coordination among robots in manufacturing and logistics, or mitigating large-scale global issues like climate change where multiple agents (like sensors, models, and decision-making systems) need to work together to monitor, predict, and mitigate environmental impacts. We hope this research benefits the RL community and inspires further research and ideas.

Appendices



Language Guided Exploration

Contents

A.1 Additional Results	157
----------------------------------	-----

A.1 Additional Results

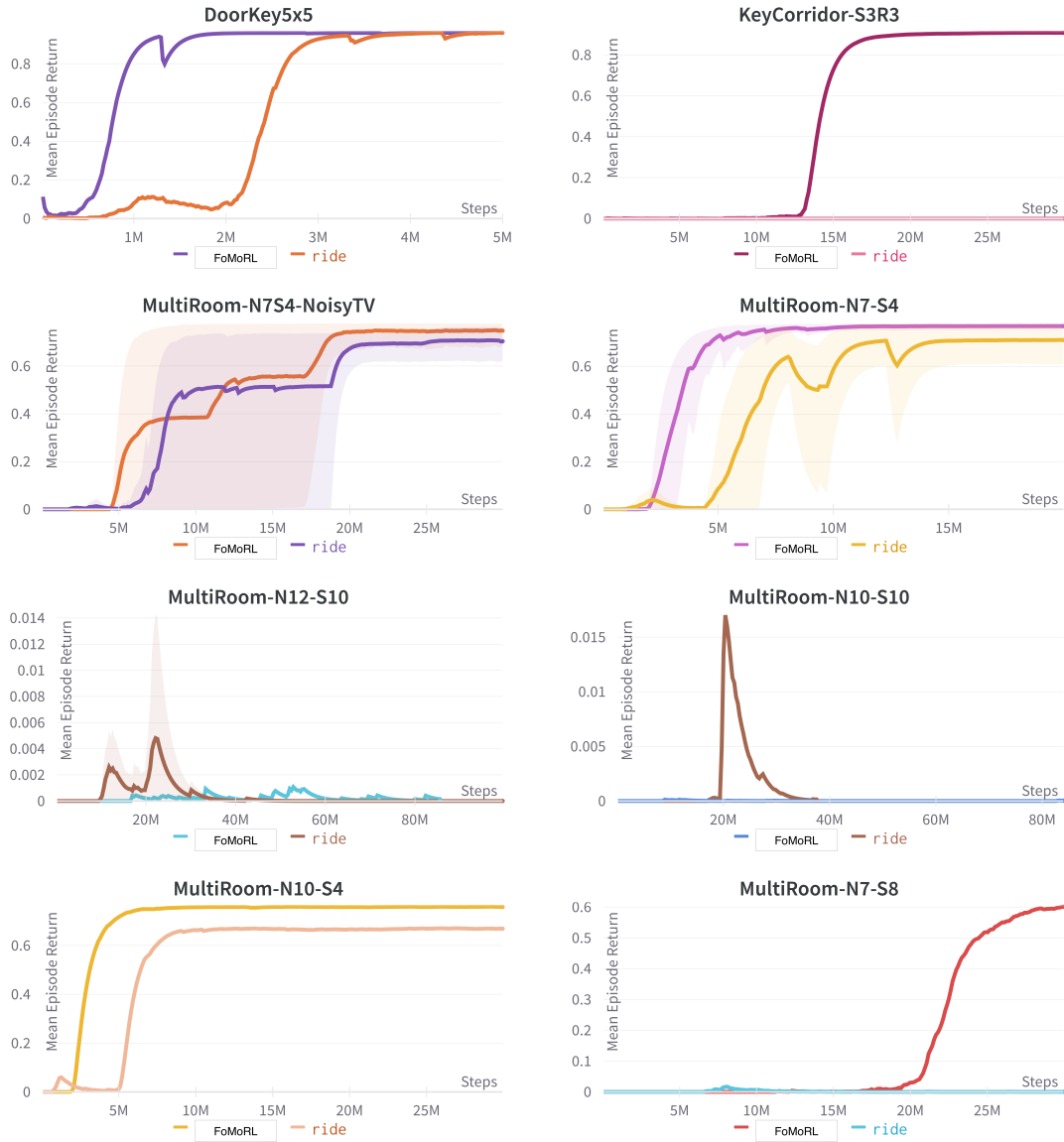


Figure A.1: Learning curves of RIDE and FoMoRL on a variety of hard exploration problems in MiniGrid. Both methods receive partially observable RGB images as input to the policy and intrinsic reward module.

B

Universal Value Exploration

Contents

B.1	Formal derivation of the illustrative example in Section 5.4	159
B.2	Training algorithm	161
B.3	Implementation Details	161
B.3.1	Hyper parameters	161
B.3.2	NN Architecture	163
B.4	Additional Results	164
B.4.1	Predator-Prey	164
B.4.2	StarCraft II Micromanagement	165

B.1 Formal derivation of the illustrative example in Section 5.4

A VDN decomposition of the joint state-action-value function in the simplified predator-prey task shown in Figure 5.1 is:

$$\begin{aligned} Q^\pi(s, u^1, u^2) &= \mathbb{E} \left[r(s, u^1, u^2) + \gamma V^\pi(s') \middle| \begin{array}{l} u^2 \sim \pi^2(\cdot | s) \\ s' \sim P(\cdot | s, u^1, u^2) \end{array} \right] \\ &\approx Q^1(s, u^1) + Q^2(s, u^2) =: Q_{\text{tot}}(s, u^1, u^2). \end{aligned}$$

Here $V^\pi(s')$ denotes the expected return of state $s' \in \mathcal{S}$. However, in this analysis we are mainly interested in the behavior at the beginning of training,

and so we assume in the following that the estimated future value is close to zero, i.e. $V^\pi(s') \approx 0, \forall s' \in \mathcal{S}$. The VDN decomposition allows to derive a decentralized policy $\pi(u^1, u^2|s) := \pi^1(u^1|s)\pi^2(u^2|s)$, where π^i is usually an ϵ -greedy policy based on agent i 's utility $Q^i(s, u^i)$, $i \in \{1, 2\}$. Similar to Rashid et al. [2020a], we analyze here the corresponding VDN projection operator:

$$\Pi_{\text{vdn}}^\pi[Q^\pi] := \arg \min_{Q^{\text{tot}}} \sum_{u^1 \in \mathcal{U}^1} \sum_{u^2 \in \mathcal{U}^2} \pi(u^1, u^2|s) \left(Q^{\text{tot}}(s, u^1, u^2) - Q^\pi(s, u^1, u^2) \right)^2.$$

Setting the gradient of the above mean-squared loss to zero yields for Q^1 (and similarly for Q^2):

$$Q^1(s, u^1) \stackrel{!}{=} \sum_{u^2 \in \mathcal{U}^2} \pi^2(u^2|s) \left(\underbrace{r(s, u^1, u^2) + \gamma \mathbb{E}[V^\pi(s') | s' \sim P(\cdot|s, u^1, u^2)]}_{Q^\pi(s, u^1, u^2)} - Q^2(s, u^2) \right).$$

In the following we use $\bar{V}_{u^1} := \max_{u^2} \mathbb{E}[V^\pi(s') | s' \sim P(\cdot|s, u^1, u^2)]$. Relative overgeneralization [Panait et al., 2006a] occurs when an agent's utility (for example agent 1's) of the catch action $u^1 = C$ falls below the movement actions $u^1 \in \{L, R\}$:

$$\begin{aligned} Q^1(s, C) &< Q^1(s, L/R) \\ \Leftrightarrow \sum_{u^2 \in \mathcal{U}^2} \pi^2(u^2|s) (r(s, C, u^2) - Q^2(s, u^2)) &< \sum_{u^2 \in \mathcal{U}^2} \pi^2(u^2|s) (Q^\pi(s, L/R, u^2) - Q^2(s, u^2)) \\ \Rightarrow r \pi^2(C|s) - p \pi^2(L|s) - p \pi^2(R|s) &< -p \pi^2(C|s) + \gamma(1 - \pi^2(C|s)) \bar{V}_{L/R} \\ \Leftrightarrow r \pi^2(C|s) &< p(\pi^2(L|s) + \pi^2(R|s) - \pi^2(C|s)) + \gamma(1 - \pi^2(C|s)) \bar{V}_{L/R} \\ \Leftrightarrow r \pi^2(C|s) &< p(1 - 2\pi^2(C|s)) + \gamma(1 - \pi^2(C|s)) \bar{V}_{L/R} \\ \Leftrightarrow r &< p \left(\frac{1}{\pi^2(C|s)} - 2 \right) + \gamma(1 - \pi^2(C|s)) \bar{V}_{L/R} \end{aligned}$$

This demonstrates that, at the beginning of training with $\bar{V}_{L/R} = 0$, relative overgeneralization occurs at $p > r$ for $\pi^2(C|s) = \frac{1}{3}$, at $p > \frac{3}{2}r$ for $\pi^2(C|s) = \frac{3}{8}$, at $p > 3r$ for $\pi^2(C|s) = \frac{3}{7}$, and at *never* at $\pi^2(C|s) > \frac{1}{2}$, as shown in Figure 5.2. Note that the value $\bar{V}_{L/R}$ contributes a constant w.r.t. punishment p , that is scaled by $1 - \pi^2(C|s)$, and that positive values make the problem harder (require lower

p). An initial relative overgeneralization will make agents choose the wrong greedy actions, which lowers $\pi^2(C|s)$ even further and therefore solidifies the pathology when ϵ gets annealed. Empirical thresholds in experiments can differ significantly, though, as random sampling increases the chance to over/underestimate $\pi^2(C|s)$ during the exploration phase and the estimated value $V^\pi(s')$ has a non-trivial influence. Furthermore, the above thresholds cannot be transferred directly to the experiments conducted in Section 5.9, which are much more challenging: (i) it requires 3 agents to catch a prey, (ii) agents have to explore 5 actions, (iii) agents can execute catch actions when they are alone with the prey, and (iv) catch actions do not end the episode, increasing the influence of empirically estimated $V^\pi(s')$. Empirically we observe that VDN fails to solve tasks somewhere between $p = 0.004r$ and $p = 0.008r$. The presented analysis provides therefore only an illustrative example how UneVEn can help to overcome relative overgeneralization.

B.2 Training algorithm

Algorithm 1 and 2 presents the training of MAUSFs with UneVEn. Our method is able to learn on all tasks (target $\mathbf{w}$ and sampled $\mathbf{z}$) simultaneously in a sample efficient manner using the same feature $\phi_t \equiv \phi(s_t, \mathbf{u}_t)$ due to the linearly decomposed reward function (Equation 5.1).

B.3 Implementation Details

B.3.1 Hyper parameters

All algorithms are implemented in the PyMARL framework [Samvelyan et al., 2019]. All our experiments use ϵ -greedy scheme where ϵ is decayed from $\epsilon = 1$ to $\epsilon = 0.05$ over $\{250k, 500k\}$ time steps. All our tasks use a discount factor of $\gamma = 0.99$. We freeze the trained policy every $30k$ timesteps and run 20 evaluation episodes with $\epsilon = 0$. We use learning rate of 0.0005 with soft target updates for all experiments. We use a target network similar to Mnih et al. [2015] with “soft”

Algorithm 1 Training MAUSFs with UneVEn

```

1: Require  $\epsilon, \alpha, \beta$  target task  $\mathbf{w}$ , set of agents  $\mathcal{A}$ , standard deviation  $\sigma$ 
2: Initialize the local-agent SF network  $\psi^a(\tau^a, u^a, \mathbf{z}; \theta)$  and replay buffer  $\mathcal{M}$ 
3: for fixed number of epochs do
4:    $\nu \sim \mathcal{N}(\mathbf{w}, \sigma \mathbf{I}_d)$ ;  $\mathbf{o}_0 \leftarrow \text{RESETENV}()$ 
5:    $t = 0$ ;  $\mathcal{M} \leftarrow \text{NEWEPISODE}(\mathcal{M}, \nu, \mathbf{o}_0)$ 
6:   while not terminated do
7:     if Bernoulli( $\epsilon$ )=1 then  $\mathbf{u}_t \leftarrow \text{Uniform}(\mathcal{U})$ 
8:     else  $\mathbf{u}_t \leftarrow \text{UNEVEN}(\tau_t, \nu)$ 
9:      $\langle \mathbf{o}_{t+1}, \phi_t \rangle \leftarrow \text{ENVSTEP}(\mathbf{u}_t)$ 
10:     $\mathcal{M} \leftarrow \text{ADDTRANSITION}(\mathcal{M}, \mathbf{u}_t, \mathbf{o}_{t+1}, \phi_t)$ 
11:     $t \leftarrow t + 1$ 
12:   end while
13:    $\mathcal{L} \leftarrow 0$ ;  $\mathcal{B} \leftarrow \text{SAMPLEMINIBATCH}(\mathcal{M})$ 
14:   for all  $\{\tau_t, \mathbf{u}_t, \phi_t, \tau_{t+1}, \nu\} \in \mathcal{B}$  do
15:     for all  $\mathbf{z} \in \nu \cup \{\mathbf{w}\}$  do
16:        $\mathbf{u}'_z \leftarrow \left\{ \arg \max_{u \in \mathcal{U}} \psi^a(\tau_{t+1}^a, u, \mathbf{z}; \theta)^\top \mathbf{z} \right\}_{a \in \mathcal{A}}$ 
17:        $\mathcal{L} \leftarrow \mathcal{L} + \left\| \phi_t + \gamma \psi_{\text{tot}}(\tau_{t+1}, \mathbf{u}'_z, \mathbf{z}; \theta^-) - \psi_{\text{tot}}(\tau_t, \mathbf{u}_t, \mathbf{z}; \theta) \right\|_2^2$ 
18:     end for
19:   end for
20:    $\theta \leftarrow \text{OPTIMIZE}(\theta, \nabla_\theta \mathcal{L})$ 
21:    $\theta^- \leftarrow (1 - \beta) \theta^- + \beta \theta$ 
22: end for

```

Algorithm 2 UNEVEN(τ_t, ν)

```

1: if Bernoulli( $\alpha$ ) = 1 or Scheme is Target then
2:    $\mathcal{C}_2 \leftarrow \{\mathbf{w}\}$ 
3: else
4:   if Scheme is Uniform then
5:      $\mathcal{C}_2 \leftarrow \nu \sim \text{Uniform}(\nu)$ 
6:   else if Scheme is Greedy then
7:      $\mathcal{C}_2 \leftarrow \nu \cup \{\mathbf{w}\}$ 
8:   end if
9: end if
10: if Use_GPI_Policy is True then
11:    $\mathcal{C}_1 \leftarrow \nu \cup \{\mathbf{w}\}$ 
12:    $\mathbf{u}_t \leftarrow \left\{ u_t^a = \arg \max_{u \in \mathcal{U}} \max_{\mathbf{k} \in \mathcal{C}_2} \max_{\mathbf{z} \in \mathcal{C}_1} \psi^a(\tau_t^a, u, \mathbf{z}; \theta)^\top \mathbf{k} \right\}_{a \in \mathcal{A}}$ 
13: else
14:    $\mathbf{u}_t \leftarrow \left\{ u_t^a = \arg \max_{u \in \mathcal{U}} \max_{\mathbf{k} \in \mathcal{C}_2} \psi^a(\tau_t^a, u, \mathbf{k}; \theta)^\top \mathbf{k} \right\}_{a \in \mathcal{A}}$ 
15: end if
16: return  $\mathbf{u}_t$ 

```

target updates, rather than directly copying the weights: $\theta^- \leftarrow \beta * \theta + (1 - \beta) * \theta^-$, where θ are the current network parameters. We use $\beta = 0.005$ for PP and m -step experiments and $\beta = 0.05$ for SC2 experiments. This means that the target values are constrained to change slowly, greatly improving the stability of learning. All algorithms were trained with RMSprop optimizer by one gradient step on loss computed on a batch of 32 episodes sampled from a replay buffer containing last 1000 episodes (for SC2, we use last 3000 episodes). We also used gradient clipping to restrict the norm of the gradient to be ≤ 10 .

The probability α of action selection based on target task in UneVEn with uniform and greedy action selection schemes increases from $\alpha = 0.3$ to $\alpha = 1.0$ over $\{250k, 500k\}$ time steps. For sampling related tasks using normal distribution, we use $\mathcal{N}(\mathbf{w}, \sigma \mathbf{I}_d)$ centered around target task $\mathbf{w}$ with $\sigma \in \{0.1, 0.2\}$. At the beginning of each episode, we sample *six* related tasks, therefore $|\nu| = 6$ (for SC2, we use $|\nu| = 3$).

B.3.2 NN Architecture

Each agent’s local observation o_t^a are concatenated with agent’s last action u_{t-1}^a , and then passed through a fully-connected (FC) layers of 128 neurons (for SC2, we use 1024 neurons), followed by ReLU activation, a GRU [Chung et al., 2014], and another FC of the same dimensionality to generate a action-observation history summary for the agent. Each agent’s task vector $\mathbf{z} \in \nu \cup \{\mathbf{w}\}$ is passed through a FC layer of 128 neurons (for SC2, we use 1024 neurons) followed by ReLU activation to generate an internal task embedding. The history and task embedding are concatenated together and passed through two hidden FC-256 layers (for SC2, FC-2048 layer) and ReLU activations to generate the outputs for each action. For methods with non-linear mixing such as QMIX [Rashid et al., 2020b], WQMIX [Rashid et al., 2020a], and MAVEN [Mahajan et al., 2019], we adopt the same hypernetworks from the original paper and test with either a single or double hypernet layers of dim 64 utilizing an ELU non-linearity. For all baseline methods, we use the code shared publicly by the corresponding authors on Github.

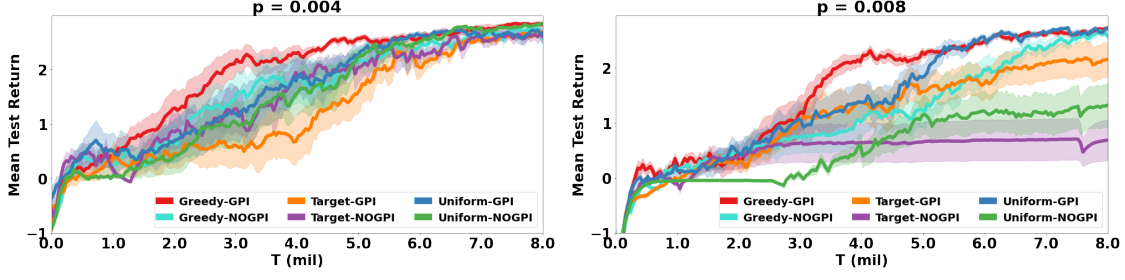


Figure B.1: Additional Ablation results: Comparison between different action selection of UneVEn for $p \in \{0.004, 0.008\}$.

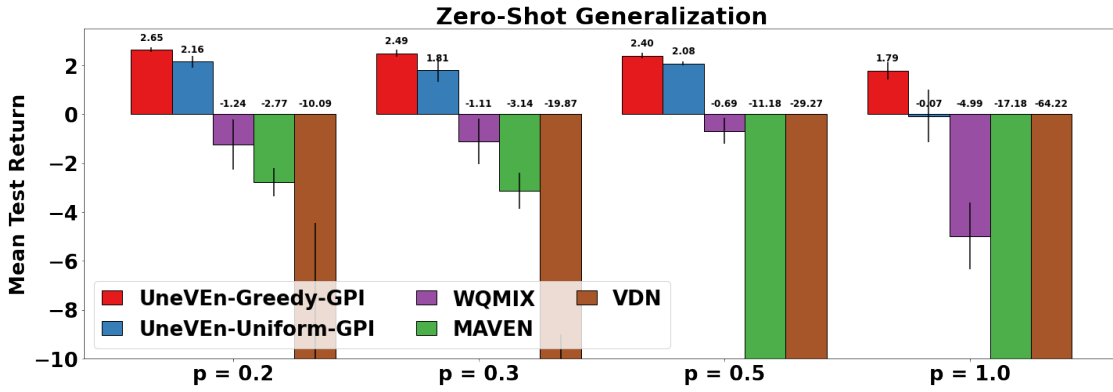


Figure B.2: Additional Zero-shot generalization results for $p \in \{0.2, 0.3, 0.5, 1.0\}$.

B.4 Additional Results

B.4.1 Predator-Prey

Figure B.1 presents additional ablation results for comparison between UneVEn with different action selection schemes for $p \in \{0.004, 0.008\}$. Figure B.2 presents additional zero-shot generalization results for policies trained on target task with penalty $p = 0.004$ tested on tasks with penalty $p \in \{0.2, 0.3, 0.5, 1.0\}$. For UneVEn-Greedy-GPI, we can observe that the average number of miscoordinated capture attempts per episode actually drops with p and converges around 1.2, i.e., for return R_p , average mistakes per episode is $\frac{3-R_p}{p} = \{1.75, 1.7, 1.2, 1.2\}$ for $p \in \{0.2, 0.3, 0.5, 1.0\}$.

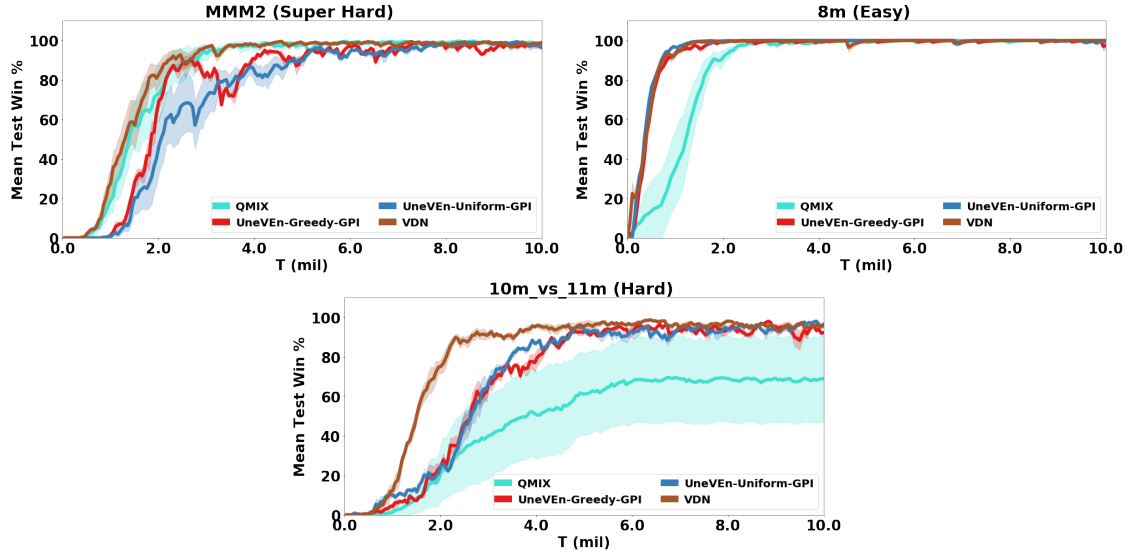


Figure B.3: Comparison between UneVEN, VDN and QMIX on SMAC maps with penalty $p = 0.5$.

B.4.2 StarCraft II Micromanagement

Figure B.3 presents the mean test win rate for SMAC maps with `reward-only-positive = False` and low penalty of $p = 0.5$. Both VDN and QMIX achieve almost 100% win rate on these maps and the additional complexity of learning MAUSFs in our approach results in slightly slower convergence. However, UneVEN with both GPI schemes matches the performance as VDN and QMIX in all maps.



Role Based Exploration

Contents

C.1	Architecture, Hyperparameters, and Infrastructure . .	167
C.2	Experimental Details	169
C.2.1	Benchmarking on StarCraft II Micromanagement Tasks	169
C.2.2	Policy Transfer	170

C.1 Architecture, Hyperparameters, and Infrastructure

In this thesis, we use simple network structures for the role selector and role policies. Each role policy is a simple linear network without hidden layers or activation functions, and the role selector is a two-layer feed-forward fully-connected network with a 64-dimensional hidden layer. Agents share a trajectory encoding network consisting of two layers, a fully-connected layer followed by a GRU layer with a 64-dimensional hidden state. For all experiments, the length of action representations, d , is set to 20. The outputs of role policies and the outputs of role selector are each fed into their own separate QMIX-style mixing networks [Rashid et al., 2020b] to estimate the global action values. The two mixing networks use the

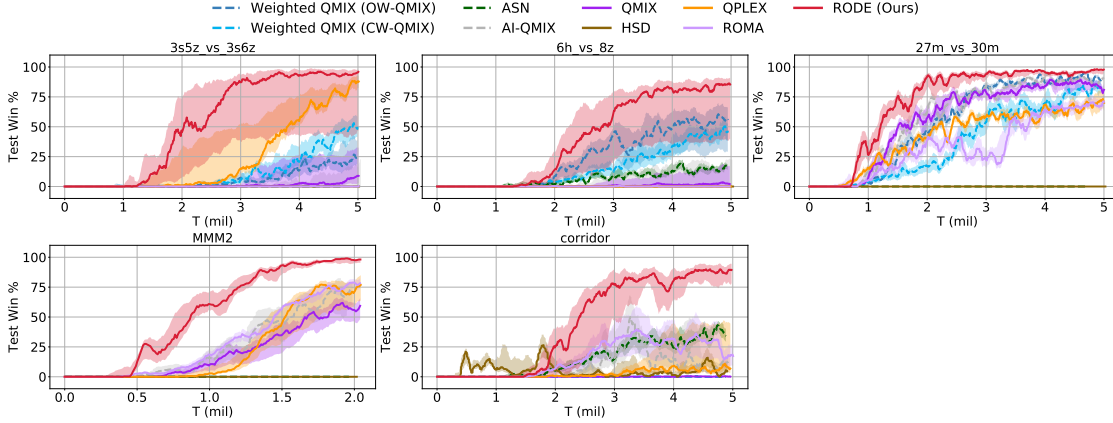


Figure C.1: Comparisons between RODE and baselines on all **super hard** maps.

same architecture, containing a 32-dimensional hidden layer with ReLU activation. Parameters of the mixing networks are generated by hypernetworks conditioning on global states. These settings are the same to QMIX [Rashid et al., 2020b]. Moreover, it can be easily extended to other mixing mechanisms, such as QPLEX [Wang et al., 2020a], which may further improve the performance of RODE.

For all experiments, the optimization is conducted using RMSprop with a learning rate of 5×10^{-4} , α of 0.99, and with no momentum or weight decay. For exploration, we use ϵ -greedy with ϵ annealed linearly from 1.0 to 0.05 over 50K time steps and kept constant for the rest of the training. For three hard exploration maps—3s5z_vs_3s6z, 6h_vs_8z, and 27m_vs_30m—we extend the epsilon annealing time to 500K, for both RODE and all the baselines and ablations. Batches of 32 episodes are sampled from the replay buffer, and the role selector and role policies are trained end-to-end on fully unrolled episodes. All experiments on the SMAC benchmark use the default reward and observation settings of the SMAC benchmark [Samvelyan et al., 2019]. Experiments are carried out on NVIDIA GTX 2080 Ti GPU.

We use k -means clustering when determining role action spaces. The number of clusters, k , is treated as a hyperparameter. Specifically, on maps with homogeneous enemies, we set k to 3, and on maps with heterogeneous enemies, we set k to 5. If the task only involves one enemy, k is set to 2. We can avoid this hyperparameter

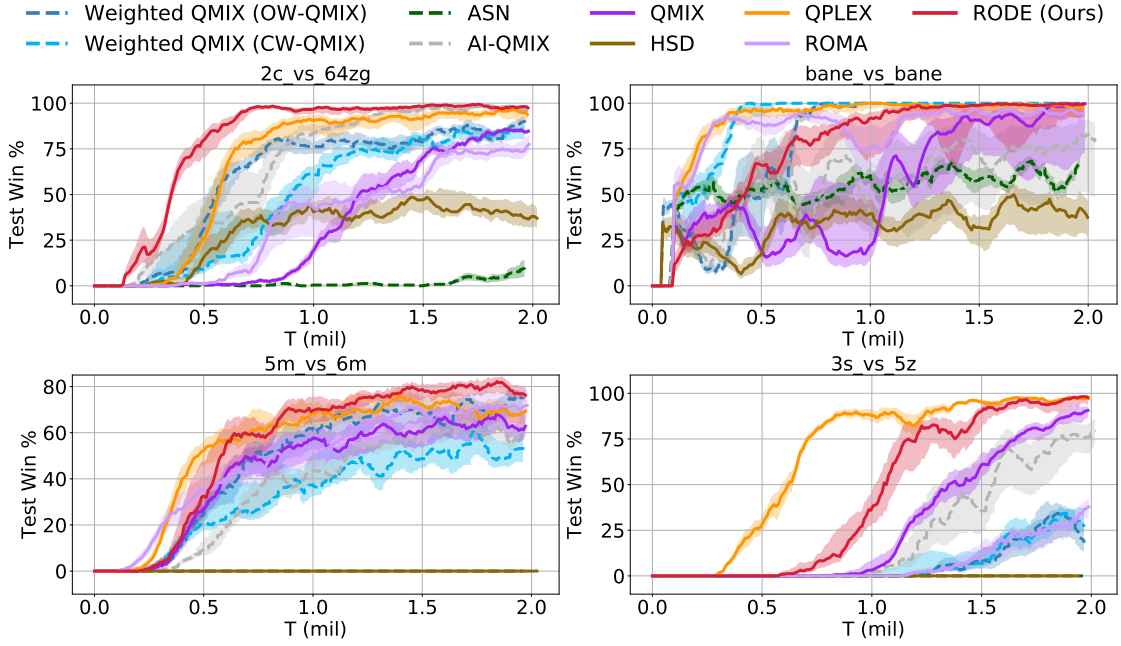


Figure C.2: Comparisons between RODE and baselines on all **hard** maps.

by using more advanced clustering approaches. For all baseline algorithms, we use the codes provided by their authors where the hyperparameters have been fine-tuned on the SMAC benchmark.

C.2 Experimental Details

In this section, we provide more experimental results supplementary to those presented in Sec. 6.4.2. We also discuss the details of experimental settings of rapid policy transfer and analyze the results.

C.2.1 Benchmarking on StarCraft II Micromanagement Tasks

In Sec. 6.4.2, we show the overall performance of RODE on the SMAC benchmark. In this section, we further show the performance of our method on all maps. All experiments in this section are carried out with 8 random seeds. Median win rate, together with 25-75% percentiles, is shown. The SMAC benchmark [Samvelyan et al., 2019] contains 14 maps that have been classified as easy, hard, and super hard. In

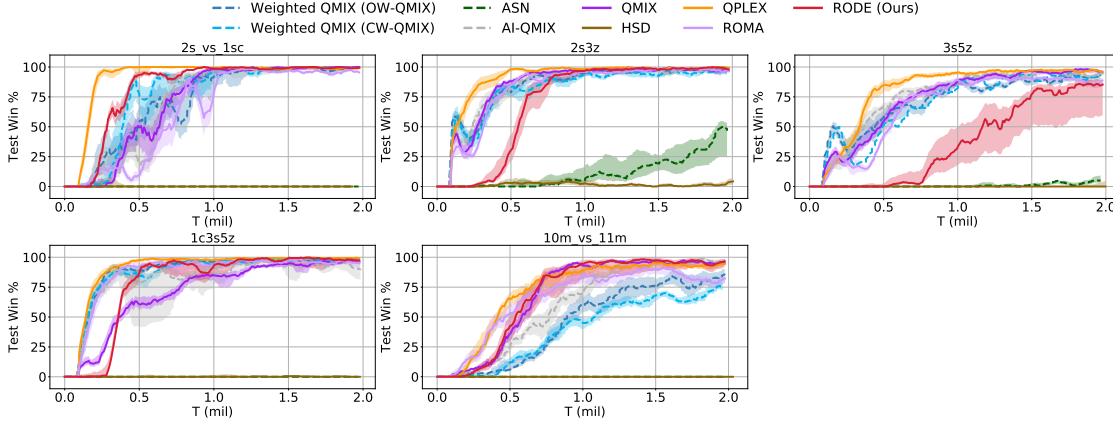


Figure C.3: Comparisons between RODE and baselines on all **easy** maps.

Fig. C.1, we compare the performance of RODE with baseline algorithms on all super hard maps. We can see that RODE outperforms all the baselines by a large margin, especially on those requiring the most exploration: **3s5z_vs_3s6z**, **corridor**, and **6h_vs_8z**. These results demonstrate that RODE can help exploration and solve complex tasks, in line with our expectations of it.

On hard maps, RODE still has superior performance compared to baselines, as shown in Fig. C.2. However, on easy maps (Fig. C.3), RODE tends to use more samples to achieve similar performance to baselines. We hypothesize that this is because easy maps do not require substantial exploration – ally agents can win by merely engaging in fight and damaging the enemy units for reward. Under these circumstances, RODE still explores a lot, which prevents it from performing better. In summary, RODE establishes a new state of the art on the SMAC benchmark by outperforming all baselines on 9 out of 14 scenarios. Notably, its superiority is particularly obvious on all of the hard and super hard maps.

C.2.2 Policy Transfer

In this section, we describe the setting of the policy transfer experiment introduced in Sec. 6.4.6. Policy transfer experiments are based on the assumption that new actions have similar functionalities to some old actions. We train an action encoder on the new task to cluster the actions. In this way, we can distinguish old actions

$a_{i_1}, \dots, a_{i_m}$ that have similar effects to a new action a_{n_i} . We then use $\frac{1}{m} \sum_{j=1}^m \mathbf{z}_{a_{i_j}}$ as the representation for a_{n_i} , where $\mathbf{z}_{a_{i_j}}$ is the representation for a_{i_j} on the original task, and we add a_{n_i} to role action spaces to which $a_{i_1}, \dots, a_{i_k}$ belong. For each new action, we repeat these operations. The number of roles does not change during this process. This process enables the role selector and role policies learned in original task to be rapidly transferred to a similar task.

To test our idea, we train RODE on the map `corridor`, where 6 ally Zealots face 24 enemy Zerglings, for 5 million timesteps and rapidly transfer the learned policy to similar maps by increasing the number of agents and enemies. The number of actions also increases because a new `attack` action is added for every new enemy unit. The results are shown in Fig. 6.10.

D

Independent PPO for SMAC

Contents

D.1 Additional Results	173
----------------------------------	-----

D.1 Additional Results

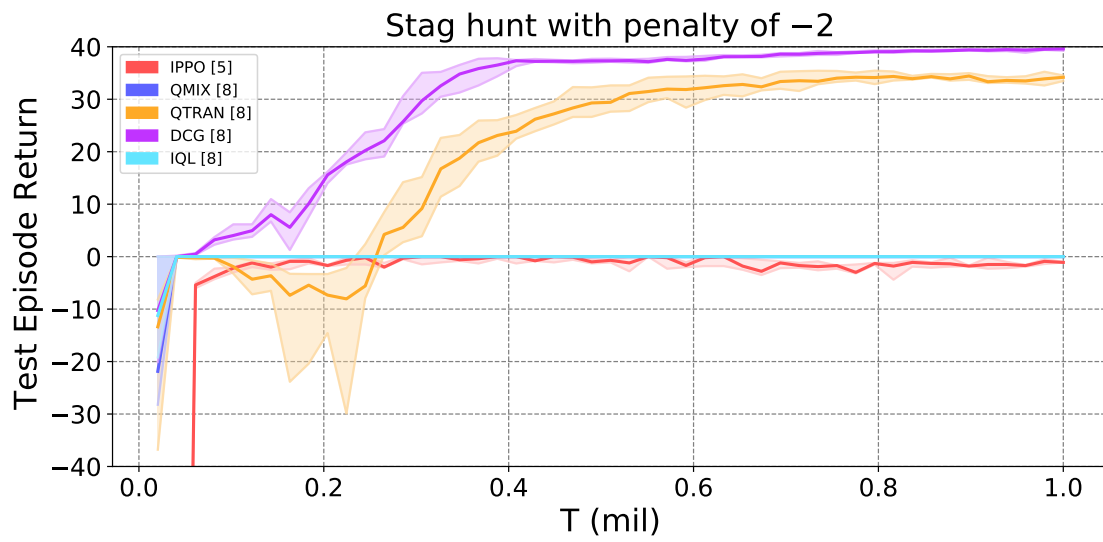


Figure D.1: Relative overgeneralization, as experienced by select MARL algorithms on Stag Hunt with penalty -2 (see Böhmer et al. [2020]). Note that DCG is not fully decentralized during execution.

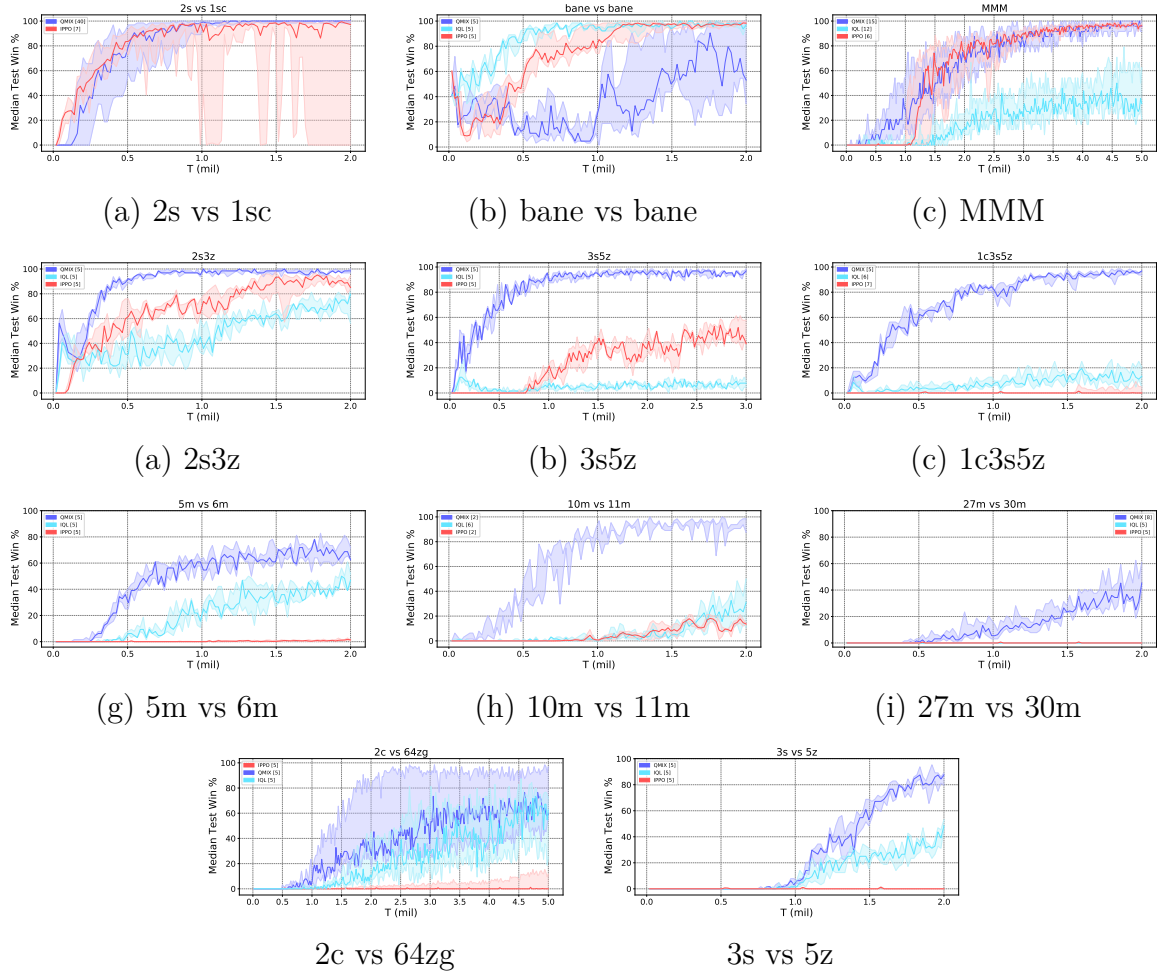


Figure D.2: Results on select SMAC maps comparing IPPO and IPPO-C with QMIX, VDN and IQL.

Bibliography

- Y. Abbasi-Yadkori and C. Szepesvári. Bayesian optimal control of smoothly parameterized systems. In *UAI*, pages 1–11, 2015. 28
- S. Ahilan and P. Dayan. Feudal multi-agent hierarchies for cooperative reinforcement learning. *arXiv preprint arXiv:1901.08492*, 2019. 110
- M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. 30
- M. Al-Emran. Hierarchical reinforcement learning: a survey. *International journal of computing and digital systems*, 4(02), 2015. 110
- C. Amato. (a partial survey of) decentralized, cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2405.06161*, 2024. 13
- S. Amin, M. Gomrokchi, H. Satija, H. van Hoof, and D. Precup. A survey of exploration methods in reinforcement learning. *arXiv preprint arXiv:2109.00157*, 2021. 25
- P. K. Anasosalu Vasu, H. Pouransari, F. Faghri, and O. Tuzel. Clip with quality captions: A strong pretraining for vision tasks. *arXiv e-prints*, pages arXiv–2405, 2024. 45
- M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017. 29
- J. Anlauf and M. Biehl. The adatron: an adaptive perceptron algorithm. *Europhysics Letters*, 10(7):687, 1989. 2
- A. Aubret, L. Matignon, and S. Hassas. A survey on intrinsic motivation in reinforcement learning. *arXiv preprint arXiv:1908.06976*, 2019. 23
- J. Ayush, S. Andrew, and J. L. Joseph. Generalization to new actions in reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, 2020. 111
- A. P. Badia, P. Sprechmann, A. Vitvitskyi, D. Guo, B. Piot, S. Kapturowski, O. Tieleman, M. Arjovsky, A. Pritzel, A. Bolt, et al. Never give up: Learning directed exploration strategies. *arXiv preprint arXiv:2002.06038*, 2020. 28, 47
- A. Barreto, W. Dabney, R. Munos, J. J. Hunt, T. Schaul, H. P. van Hasselt, and D. Silver. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, pages 4055–4065, 2017. 73, 74, 84

- A. Barreto, D. Borsa, J. Quan, T. Schaul, D. Silver, M. Hessel, D. Mankowitz, A. Zidek, and R. Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *International Conference on Machine Learning*, pages 501–510. PMLR, 2018. 73, 84
- A. Barreto, S. Hou, D. Borsa, D. Silver, and D. Precup. Fast reinforcement learning with generalized policy updates. *Proceedings of the National Academy of Sciences*, 2020. 80, 99
- D. Batra, A. Gokaslan, A. Kembhavi, O. Maksymets, R. Mottaghi, M. Savva, A. Toshev, and E. Wijmans. ObjectNav Revisited: On Evaluation of Embodied Agents Navigating to Objects. In *arXiv:2006.13171*, 2020. 30, 47
- R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman. Solving transition independent decentralized markov decision processes. *Journal of Artificial Intelligence Research*, 22: 423–455, 2004. 84
- M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013. 111
- M. G. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems (NIPS) 29*, pages 1471–1479, 2016. 23, 25, 26, 83
- M. G. Bellemare, W. Dabney, and R. Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pages 449–458. PMLR, 2017. 55, 60
- W. Böhmer, T. Rashid, and S. Whiteson. Exploration with unreliable intrinsic reward in multi-agent reinforcement learning. *CoRR*, abs/1906.02138, 2019. URL <http://arxiv.org/abs/1906.02138>. Presented at the ICML *Exploration in Reinforcement Learning* workshop. 61, 83
- W. Böhmer, V. Kurin, and S. Whiteson. Deep coordination graphs. In *International Conference on Machine Learning*, pages 980–991. PMLR, 2020. 17, 56, 59
- N. Bonjean, W. Mefteh, M. Gleizes, C. Maurel, and F. Migeon. Adelfe 2.0: Handbook on agent-oriented design processes, m. cossentino, v. hilaire, a. molesini, and v. seidita, 2014. 103, 112
- A. M. Borghi and F. Binkofski. *Words as social tools: An embodied view on abstract concepts*, volume 2. Springer, 2014. 33
- D. Borsa, A. Barreto, J. Quan, D. Mankowitz, R. Munos, H. van Hasselt, D. Silver, and T. Schaul. Universal successor features approximators. *arXiv preprint arXiv:1812.07626*, 2018. 71, 73, 74, 76, 84
- P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004. 112

- T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 30
- Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018. 24, 27, 28, 35, 37, 46
- Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros. Large-scale study of curiosity-driven learning. In *International Conference on Learning Representations (ICLR)*, 2019. 26, 61, 83
- L. Busoniu, R. Babuska, and B. De Schutter. A Comprehensive Survey of Multiagent Reinforcement Learning. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38:156–172, Apr. 2008. doi: 10.1109/TSMCC.2007.913919. 134
- E. Butler. *The condensed wealth of nations*. Centre for Independent Studies, 2012. 102, 112
- W. Böhmer, T. Rashid, and S. Whiteson. Exploration with Unreliable Intrinsic Reward in Multi-Agent Reinforcement Learning. *arXiv:1906.02138 [cs]*, June 2019. URL <http://arxiv.org/abs/1906.02138>. arXiv: 1906.02138. 134, 135
- W. Böhmer, V. Kurin, and S. Whiteson. Deep Coordination Graphs. *arXiv:1910.00091 [cs]*, June 2020. URL <http://arxiv.org/abs/1910.00091>. arXiv: 1910.00091. xviii, 135, 174
- Y. Cao, W. Yu, W. Ren, and G. Chen. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial informatics*, 9(1):427–438, 2012. 5
- Y. Chandak, G. Theodorou, J. Kostas, S. Jordan, and P. Thomas. Learning action representations for reinforcement learning. In *International Conference on Machine Learning*, pages 941–950, 2019. 111
- Y. Chen, Y. Chen, Y. Yang, Y. Li, J. Yin, and C. Fan. Learning action-transferable policy with action embedding. *arXiv preprint arXiv:1909.02291*, 2019. 112
- M. Chevalier-Boisvert, L. Willems, and S. Pal. Minimalistic gridworld environment for gymnasium. <https://github.com/Farama-Foundation/MiniGrid>, 2018. xv, 22, 25, 36
- L. Choshen, L. Fox, and Y. Loewenstein. Dora the explorer: Directed outreaching reinforcement action-selection. *arXiv preprint arXiv:1804.04012*, 2018. 26
- J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. 3, 15, 163
- C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI*, 1998(746-752):2, 1998. 52, 102, 134, 135, 147

- V. Conitzer and T. Sandholm. Awesome: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. *Machine Learning*, 67(1-2):23–43, 2007. 52
- M. Cossentino, S. Gaglio, L. Sabatucci, and V. Seidita. The passi and agile passi mas meta-models compared with a unifying proposal. In *International Central and Eastern European Conference on Multi-Agent Systems*, pages 183–192. Springer, 2005. 103, 112
- H. Cui and R. Khardon. Online symbolic gradient-based optimization for factored action mdps. In *IJCAI*, pages 3075–3081, 2016. 111
- H. Cui and R. Khardon. Lifted stochastic planning, belief propagation and marginal map. In *AAAI Workshops*, pages 658–664, 2018. 111
- W. Czarnecki, S. Jayakumar, M. Jaderberg, L. Hasenclever, Y. W. Teh, N. Heess, S. Osindero, and R. Pascanu. Mix & match agent curricula for reinforcement learning. In *International Conference on Machine Learning*, pages 1087–1095, 2018. 112
- C. Daniel, G. Neumann, and J. Peters. Hierarchical relative entropy policy search. In *Artificial Intelligence and Statistics*, pages 273–281, 2012. 110
- A. Das, T. Gervet, J. Romoff, D. Batra, D. Parikh, M. Rabbat, and J. Pineau. Tarmac: Targeted multi-agent communication. In *International Conference on Machine Learning*, pages 1538–1546, 2019. 102
- P. Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993. 73, 84
- P. Dayan and G. E. Hinton. Feudal reinforcement learning. In *Advances in neural information processing systems*, pages 271–278, 1993. 110
- P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005. 60
- C. A. S. de Witt, J. N. Foerster, G. Farquhar, P. H. Torr, W. Boehmer, and S. Whiteson. Multi-agent common knowledge reinforcement learning. *arXiv preprint arXiv:1810.11702*, 2018. 64
- C. S. de Witt, B. Peng, P.-A. Kamienny, P. Torr, W. Böhmer, and S. Whiteson. Deep multi-agent reinforcement learning for decentralized continuous cooperative control. *arXiv preprint arXiv:2003.06709*, 2020. 60, 64, 135, 136
- S. A. DeLoach and J. C. Garcia-Ojeda. O-mase: a customisable approach to designing and building complex, adaptive multi-agent systems. *International Journal of Agent-Oriented Software Engineering*, 4(3):244–280, 2010. 112
- N. Dilokthanakul, C. Kaplanis, N. Pawlowski, and M. Shanahan. Feature control as intrinsic motivation for hierarchical reinforcement learning. *IEEE transactions on neural networks and learning systems*, 30(11):3409–3418, 2019. 110

- M. Dotoli, A. Fay, M. Miśkowicz, and C. Seatzu. Advanced control in factory automation: a survey. *International Journal of Production Research*, 55(5):1243–1259, March 2017. doi: 10.1080/00207543.2016.117. URL <https://ideas.repec.org/a/taf/tprxxx/v55y2017i5p1243-1259.html>. 5
- Y. Du, L. Han, M. Fang, J. Liu, T. Dai, and D. Tao. Liir: Learning individual intrinsic reward in multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 4403–4414, 2019. 63
- G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, and B. Coppin. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015. 111
- Z. Dwiel, M. Candadai, M. Phielipp, and A. K. Bansal. Hierarchical policy learning is sensitive to goal space design. *arXiv preprint arXiv:1905.01537*, 2019. 110
- B. Ellis, S. Moalla, M. Samvelyan, M. Sun, A. Mahajan, J. N. Foerster, and S. Whiteson. Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2212.07489*, 2022. 147, 152
- L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry. Implementation Matters in Deep Policy Gradients: A Case Study on PPO and TRPO. *arXiv:2005.12729 [cs, stat]*, May 2020. URL <http://arxiv.org/abs/2005.12729>. arXiv: 2005.12729. 137
- L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pages 1407–1416. PMLR, 2018. 16, 24, 31, 35, 37
- M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, pages 226–231, 1996. 126
- B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018. 30, 64
- H. Fang, P. Xiong, L. Xu, and Y. Chen. Clip2video: Mastering video-text retrieval via image clip. *arXiv preprint arXiv:2106.11097*, 2021. 44
- G. Farquhar, L. Gustafson, Z. Lin, S. Whiteson, N. Usunier, and G. Synnaeve. Growing action spaces. In *Proceedings of the 37th International Conference on Machine Learning*, 2020. 112
- J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. Torr, P. Kohli, and S. Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1146–1155. JMLR. org, 2017. 54
- J. N. Foerster, Y. M. Assael, N. De Freitas, and S. Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *arXiv preprint arXiv:1605.06676*, 2016. 54, 56, 102

- J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson. Counterfactual multi-agent policy gradients. In *Thirty-second AAAI conference on artificial intelligence*, 2018. 17, 62, 131, 134, 136, 137, 139, 140, 143, 144
- M. Fortunato, M. G. Azar, B. Piot, J. Menick, M. Hessel, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, C. Blundell, and S. Legg. Noisy networks for exploration. In *International Conference on Learning Representations (ICLR)*, 2018. URL <https://openreview.net/forum?id=rywHCPkAW>. 26, 83
- J. Fu, J. Co-Reyes, and S. Levine. Ex2: Exploration with exemplar models for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017. 26
- Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016. 29
- Y. Gal, J. Hron, and A. Kendall. Concrete dropout. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3584–3593, 2017. 26, 83
- D. Ghosh, A. Gupta, and S. Levine. Learning actionable representations with goal conditioned policies. In *International Conference on Learning Representations*, 2018. 110
- C. A. Gomez-Uribe and N. Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, 6(4):1–19, 2015. 2, vii
- D. M. Gordon. The organization of work in social insect colonies. *Nature*, 380(6570): 121–124, 1996. 112
- K. Gregor, D. J. Rezende, and D. Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016. 30, 64, 110
- C. Guestrin, M. Lagoudakis, and R. Parr. Coordinated reinforcement learning. In *ICML*, volume 2, pages 227–234. Citeseer, 2002. 55, 57
- A. Gupta, R. Mendonca, Y. Liu, P. Abbeel, and S. Levine. Meta-reinforcement learning of structured exploration strategies. In *Advances in Neural Information Processing Systems*, pages 5302–5311, 2018a. 30
- J. K. Gupta, M. Egorov, and M. Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 66–83. Springer, 2017. 102
- T. Gupta, A. Kumar, and P. Paruchuri. Planning and learning for decentralized mdps with event driven rewards. In *AAAI*, 2018b. 85
- T. Gupta, A. Kumar, and P. Paruchuri. Successor features based multi-agent rl for event-based decentralized mdps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6054–6061, 2019. 84

- D. Ha, A. Dai, and Q. V. Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016. 59, 63, 100, 109
- T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018. 65
- N. Haber, D. Mrowca, S. Wang, L. F. Fei-Fei, and D. L. Yamins. Learning to play with intrinsically-motivated, self-aware agents. *Advances in neural information processing systems*, 31, 2018. 27
- S. Hansen, W. Dabney, A. Barreto, T. Van de Wiele, D. Warde-Farley, and V. Mnih. Fast task inference with variational intrinsic successor features. *arXiv preprint arXiv:1906.05030*, 2019. 84, 99
- M. Hassaballah and K. M. Hosny. Recent advances in computer vision. *Studies in computational intelligence*, 804:1–84, 2019. 3
- M. Hausknecht and P. Stone. Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposium Series*, 2015. 14, 55, 58
- J. He, J. Chen, X. He, J. Gao, L. Li, L. Deng, and M. Ostendorf. Deep reinforcement learning with a natural language action space. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1621–1630, 2016. 111
- K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 141
- P. Henderson, T. Doan, R. Islam, and D. Meger. Bayesian policy gradients via alpha divergence dropout inference. *arXiv preprint arXiv:1712.02037*, 2017. 29
- M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 54
- S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997. 15, 36
- R. Houthoofd, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel. Vime: Variational information maximizing exploration. *Advances in neural information processing systems*, 29, 2016. 26, 27
- C. C.-Y. Hsu, C. Mendler-Dünner, and M. Hardt. Revisiting Design Choices in Proximal Policy Optimization. *arXiv:2009.10897 [cs, stat]*, Sept. 2020. URL <http://arxiv.org/abs/2009.10897>. arXiv: 2009.10897. 137
- F.-H. Hsu. *Behind Deep Blue: Building the computer that defeated the world chess champion*. Princeton University Press, 2002. 2

- J. Hu, S. A. Harding, H. Wu, and S.-w. Liao. Qr-mix: Distributional value function factorisation for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2009.04197*, 2020. 60
- M. Hüttenrauch, A. Šošić, and G. Neumann. Guided deep reinforcement learning for swarm systems. *arXiv preprint arXiv:1709.06011*, 2017. 5
- S. Iqbal and F. Sha. Actor-attention-critic for multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 2961–2970. PMLR, 2019a. 63
- S. Iqbal and F. Sha. Coordinated exploration via intrinsic rewards for multi-agent reinforcement learning. *arXiv preprint arXiv:1905.12127*, 2019b. 65
- S. Iqbal, C. A. S. de Witt, B. Peng, W. Böhmer, S. Whiteson, and F. Sha. Ai-qmix: Attention and imagination for dynamic multi-agent reinforcement learning. *arXiv preprint arXiv:2006.04222*, 2020. 115
- E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016. 56, 57
- N. Jaques, A. Lazaridou, E. Hughes, Ç. Gülçehre, P. A. Ortega, D. Strouse, J. Z. Leibo, and N. de Freitas. Intrinsic social motivation via causal influence in multi-agent RL. *CoRR*, abs/1810.08647, 2018. URL <https://arxiv.org/abs/1810.08647>. 61, 83
- R. Jeanson, P. F. Kukuk, and J. H. Fewell. Emergence of division of labour in halictine bees: contributions of social interactions and behavioural variance. *Animal behaviour*, 70(5):1183–1193, 2005. 112
- J. Jiang, C. Dun, T. Huang, and Z. Lu. Graph convolutional reinforcement learning. *arXiv preprint arXiv:1810.09202*, 2018. 62
- J. Jin and X. Ma. Hierarchical multi-agent control of traffic lights based on collective learning. *Engineering applications of artificial intelligence*, 68:236–248, 2018. 111
- A. Khandelwal, L. Weihs, R. Mottaghi, and A. Kembhavi. Simple but effective: Clip embeddings for embodied ai. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14829–14838, 2022. 30, 44, 45
- H. Kim, J. Kim, Y. Jeong, S. Levine, and H. O. Song. Emi: Exploration with mutual information. In *International Conference on Machine Learning*, pages 3360–3369, 2019. 112
- A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023. 46
- A. S. Klyubin, D. Polani, and C. L. Nehaniv. Empowerment: A universal agent-centric measure of control. In *2005 IEEE congress on evolutionary computation*, volume 1, pages 128–135. IEEE, 2005. 26
- J. R. Kok and N. Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *Journal of Machine Learning Research*, 7:1789–1828, 2006. 56

- V. R. Konda and J. N. Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014. Citeseer, 2000. 15, 138
- I. Kostrikov. Value Function Clipping for PPO · Issue #136 · ikostrikov/pytorch-a2c-ppo-acktr-gail — github.com.
<https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail/issues/136>.
[Accessed 29-11-2023]. 144
- L. Kraemer and B. Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190:82–94, 2016. 17, 53
- F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519, 2001. 56
- T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, pages 3675–3683, 2016a. 29
- T. D. Kulkarni, A. Saeedi, S. Gautam, and S. J. Gershman. Deep successor reinforcement learning. *arXiv preprint arXiv:1606.02396*, 2016b. 84
- K. Kurach, A. Raichuk, P. Stańczyk, M. Zajac, O. Bachem, L. Espeholt, C. Riquelme, D. Vincent, M. Michalski, O. Bousquet, et al. Google research football: A novel reinforcement learning environment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4501–4510, 2020. 128
- M. Lauer and M. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *In Proceedings of the Seventeenth International Conference on Machine Learning*. Citeseer, 2000. 53
- D. Lee, S. Srinivasan, and F. Doshi-Velez. Truly batch apprenticeship learning with deep successor features. *arXiv preprint arXiv:1903.10077*, 2019a. 84
- Y. Lee, J. Yang, and J. J. Lim. Learning to coordinate manipulation skills via skill behavior diversification. In *International Conference on Learning Representations*, 2019b. 111
- L. Lehnert, S. Tellex, and M. L. Littman. Advantages and limitations of using successor features for transfer in reinforcement learning. *arXiv preprint arXiv:1708.00102*, 2017. 84
- J. Z. Leibo, V. Zambaldi, M. Lanctot, J. Marecki, and T. Graepel. Multi-agent reinforcement learning in sequential social dilemmas. *arXiv preprint arXiv:1702.03037*, 2017. 54
- J. Z. Leibo, E. Hughes, M. Lanctot, and T. Graepel. Autocurricula and the emergence of innovation from social interaction: A manifesto for multi-agent intelligence research. *CoRR*, abs/1903.00742, 2019. URL <http://arxiv.org/abs/1903.00742>. 83
- A. Levy, G. Konidaris, R. Platt, and K. Saenko. Learning multi-level hierarchies with hindsight. In *International Conference on Learning Representations*, 2018. 110

- K. M. Lhaksmana, Y. Murakami, and T. Ishida. Role-based modeling for designing agent behavior in self-organizing multi-agent systems. *International Journal of Software Engineering and Knowledge Engineering*, 28(01):79–96, 2018. 112
- T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In Y. Bengio and Y. LeCun, editors, *ICLR*, 2016. URL <http://arxiv.org/abs/1509.02971>. 57, 62
- A. T. Lin, M. J. DeBord, K. Estabridis, G. Hower, and S. Osher. Cesma: Centralized expert supervises multi-agents. *arXiv preprint arXiv:1902.02311*, 2019. 57
- L.-J. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3):293–321, 1992. 14, 79
- Z. Lipton, X. Li, J. Gao, L. Li, F. Ahmed, and L. Deng. Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 28
- R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems*, pages 6379–6390, 2017. 62, 131
- H. Luo, L. Ji, M. Zhong, Y. Chen, W. Lei, N. Duan, and T. Li. Clip4clip: An empirical study of clip for end to end video clip retrieval. *arXiv preprint arXiv:2104.08860*, 2021. 44
- X. Lyu and C. Amato. Likelihood quantile networks for coordinating multi-agent reinforcement learning. *arXiv preprint arXiv:1812.06319*, 2018. 55
- X. Lyu, A. Baisero, Y. Xiao, B. Daley, and C. Amato. On centralized critics in multi-agent reinforcement learning. *Journal of Artificial Intelligence Research*, 77: 295–354, 2023. 139, 143, 145
- C. Ma, D. R. Ashley, J. Wen, and Y. Bengio. Universal successor features for transfer reinforcement learning. *arXiv preprint arXiv:2001.04025*, 2020. 84
- M. C. Machado, M. G. Bellemare, and M. Bowling. Count-based exploration with the successor representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5125–5133, 2020. 34
- A. Mahajan and T. Tulabandhula. Symmetry detection and exploitation for function approximation in deep rl. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 1619–1621, 2017a. 83, 111
- A. Mahajan and T. Tulabandhula. Symmetry learning for function approximation in reinforcement learning. *arXiv preprint arXiv:1706.02999*, 2017b. 83, 111
- A. Mahajan, T. Rashid, M. Samvelyan, and S. Whiteson. Maven: Multi-agent variational exploration. *Advances in Neural Information Processing Systems*, 32:7613–7624, 2019. xvi, 59, 70, 85, 87, 88, 102, 135, 142, 147, 163

- A. Mahajan, M. Samvelyan, L. Mao, V. Makoviychuk, A. Garg, J. Kossai, S. Whiteson, Y. Zhu, and A. Anandkumar. Tesseract: Tensorised actors for multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 7301–7312. PMLR, 2021. 61
- A. Malysheva, T. T. Sung, C.-B. Sohn, D. Kudenko, and A. Shpilman. Deep multi-agent reinforcement learning with relevance graphs. *arXiv preprint arXiv:1811.12557*, 2018. 56
- J. Martin, S. N. Sasikumar, T. Everitt, and M. Hutter. Count-based exploration in feature space for reinforcement learning. *arXiv preprint arXiv:1706.08090*, 2017. 34
- L. Matignon, G. J. Laurent, and N. Le Fort-Piat. Hysteretic q-learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 64–69. IEEE, 2007. 53, 55
- L. Matignon, G. J. Laurent, and N. Le Fort-Piat. Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems. *Knowledge Engineering Review*, 27(1):1–31, 2012. 52
- O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7327–7334, 2022. 47
- L. S.-T. Memory. Long short-term memory. *Neural computation*, 9(8):1735–1780, 2010. 3
- V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. 2, vii, 14, 54, 58, 79, 161
- V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016. 16, 31, 140
- M. Mock, S. Edavettal, C. Langmead, and A. Russell. Ai can help to speed up drug discovery—but only if we give it the right data. *Nature*, 621(7979):467–470, 2023. 1
- I. Mordatch and P. Abbeel. Emergence of grounded compositional language in multi-agent populations. corr abs/1703.04908 (2017). *arXiv preprint arXiv:1703.04908*, 2017. 56
- J. Mu, V. Zhong, R. Raileanu, M. Jiang, N. Goodman, T. Rocktäschel, and E. Grefenstette. Improving intrinsic exploration with language abstractions. *arXiv preprint arXiv:2202.08938*, 2022. 30
- A. Murali, L. Pinto, D. Gandhi, and A. Gupta. Cassl: Curriculum accelerated self-supervised learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6453–6460. IEEE, 2018. 112

- O. Nachum, S. Gu, H. Lee, and S. Levine. Near-optimal representation learning for hierarchical reinforcement learning. In *International Conference on Learning Representations*, 2018a. 110
- O. Nachum, S. S. Gu, H. Lee, and S. Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 3303–3313, 2018b. 110
- R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella. Taming decentralized pomdps: Towards efficient policy computation for multiagent settings. In *IJCAI*, volume 3, pages 705–711, 2003. 52
- S. Nair and C. Finn. Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation. In *International Conference on Learning Representations*, 2019. 110
- S. Nasiriany, V. Pong, S. Lin, and S. Levine. Planning with goal-conditioned policies. In *Advances in Neural Information Processing Systems*, pages 14843–14854, 2019. 110
- Z. Nawrat. Introduction to ai-driven surgical robots. *Artif Intell Surg*, 3(2):90–7, 2023. 1
- B. O’Donoghue, I. Osband, R. Munos, and V. Mnih. The uncertainty Bellman equation and exploration. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 3836–3845, 2018. URL <http://proceedings.mlr.press/v80/o-donoghue18a.html>. 26, 83
- F. A. Oliehoek, C. Amato, et al. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016. 16, 104, 133, 134, 137
- A. Omicini. Soda: Societies and infrastructures in the analysis and design of agent-based systems. In *International Workshop on Agent-Oriented Software Engineering*, pages 185–193. Springer, 2000. 112
- S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2681–2690. JMLR. org, 2017. 54
- S. Omidshafiei, D.-K. Kim, M. Liu, G. Tesauero, M. Riemer, C. Amato, M. Campbell, and J. P. How. Learning to teach in cooperative multiagent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6128–6136, 2019. 57
- R. OpenAI. Gpt-4 technical report. *arXiv*, pages 2303–08774, 2023. 2
- M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 46
- I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep exploration via bootstrapped dqn. *Advances in neural information processing systems*, 29, 2016a. 29

- I. Osband, B. Van Roy, and Z. Wen. Generalization and exploration via randomized value functions. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning (ICML)*, pages 2377–2386, 2016b. 26, 83
- I. Osband, J. Aslanides, and A. Cassirer. Randomized prior functions for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 31, 2018. 29
- I. Osband, B. Van Roy, D. J. Russo, and Z. Wen. Deep exploration via randomized value functions. *Journal of Machine Learning Research*, 20(124):1–62, 2019. 29
- M. Ossenkopf, M. Jorgensen, and K. Geihs. When does communication learning need hierarchical multi-agent deep reinforcement learning. *Cybernetics and Systems*, 50(8): 672–692, 2019. 110
- P.-Y. Oudeyer and F. Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, page 6, 2009. 23
- P.-Y. Oudeyer, F. Kaplan, and V. V. Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2):265–286, 2007. 25
- L. Padgham and M. Winikoff. Prometheus: A methodology for developing intelligent agents. In *International Workshop on Agent-Oriented Software Engineering*, pages 174–185. Springer, 2002. 112
- S. Pageaud, V. Deslandres, V. Lehoux, and S. Hassas. Multiagent learning and coordination with clustered deep q-network. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2156–2158, 2019. 56
- G. Palmer, K. Tuyls, D. Bloembergen, and R. Savani. Lenient multi-agent deep reinforcement learning. *arXiv preprint arXiv:1707.04402*, 2017. 55
- G. Palmer, R. Savani, and K. Tuyls. Negative update intervals in deep multi-agent reinforcement learning. *arXiv preprint arXiv:1809.05096*, 2018. 55
- L. Panait and S. Luke. Cooperative Multi-Agent Learning: The State of the Art. *Autonomous Agents and Multi-Agent Systems*, 11:387–434, Nov. 2005. doi: 10.1007/s10458-005-2631-2. 134
- L. Panait, S. Luke, and R. P. Wiegand. Biasing coevolutionary search for optimal multiagent behaviors. *IEEE Transactions on Evolutionary Computation*, 10(6): 629–645, 2006a. 59, 70, 75, 160
- L. Panait, K. Sullivan, and S. Luke. Lenience towards teammates helps in cooperative multiagent learning. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi Agent Systems—AAMAS-2006*. Citeseer, 2006b. 55
- O. Patashnik, Z. Wu, E. Shechtman, D. Cohen-Or, and D. Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2085–2094, 2021. 44

- R. Patel and E. Pavlick. Mapping language models to grounded conceptual spaces. In *International Conference on Learning Representations*, 2021. 30
- D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017. 23, 25, 27, 83
- J. Pavón and J. Gómez-Sanz. Agent oriented software engineering with ingenias. In *International Central and Eastern European Conference on Multi-Agent Systems*, pages 394–403. Springer, 2003. 103, 112
- J. Papis and R. Parr. Generalized value functions for large action sets. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 1185–1192, 2011. 111
- D. Pelleg, A. W. Moore, et al. X-means: Extending k-means with efficient estimation of the number of clusters. In *International Conference of Machine Learning*, volume 1, pages 727–734, 2000. 126
- P. Peng, Y. Wen, Y. Yang, Q. Yuan, Z. Tang, H. Long, and J. Wang. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069*, 2017. 56
- M. Peña. Waypoint - The official Waymo blog: The Waymo Driver is already improving road safety — waymo.com. <https://waymo.com/blog/2023/07/the-waymo-driver-is-already-improving.html>, 2023. [Accessed 21-11-2023]. 1
- H. X. Pham, H. M. La, D. Feil-Seifer, and A. Nefian. Cooperative and distributed reinforcement learning of drones for field coverage. *arXiv preprint arXiv:1803.07250*, 2018. 5
- D. Precup. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, page 80, 2000. 63
- A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 30
- A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021. xv, 24, 32, 33
- R. Raileanu and T. Rocktäschel. Ride: Rewarding impact-driven exploration for procedurally-generated environments. *arXiv preprint arXiv:2002.12292*, 2020. 24, 25, 27, 28, 34, 37
- T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. N. Foerster, and S. Whiteson. QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 4292–4301, 2018. 58, 72, 102, 108, 115

- T. Rashid, G. Farquhar, B. Peng, and S. Whiteson. Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 33:10199–10210, 2020a. 60, 70, 80, 85, 91, 115, 135, 160, 163
- T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint arXiv:2003.08839*, 21(1):7234–7284, 2020b. 17, 58, 70, 78, 85, 100, 135, 136, 141, 143, 163, 167, 168
- M. Roderick, C. Grimm, and S. Tellex. Deep abstract Q-networks. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 131–138, 2018. 83
- G. A. Rummery and M. Niranjan. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994. 13
- A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015. 55
- B. Sallans and G. E. Hinton. Reinforcement learning with factored states and actions. *Journal of Machine Learning Research*, 5(Aug):1063–1088, 2004. 111
- M. Samvelyan, T. Rashid, C. S. De Witt, G. Farquhar, N. Nardelli, T. G. Rudner, C.-M. Hung, P. H. Torr, J. Foerster, and S. Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019. xvi, 6, 64, 71, 93, 94, 102, 104, 113, 131, 135, 141, 161, 168, 169
- T. Schaul, D. Horgan, K. Gregor, and D. Silver. Universal value function approximators. In *International conference on machine learning*, pages 1312–1320, 2015a. 73, 100
- T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015b. 54
- J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015. 16, 139, 140
- J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust Region Policy Optimization. *arXiv:1502.05477 [cs]*, Apr. 2017a. URL <http://arxiv.org/abs/1502.05477>. arXiv: 1502.05477. 137, 138
- J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017b. 135, 137, 138
- M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997. 56
- S. Shalev-Shwartz, S. Shammah, and A. Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016. 5

- T. Shankar and A. Gupta. Learning robot skills with temporal variational inference. In *Proceedings of the 37th International Conference on Machine Learning*. JMLR. org, 2020. 110
- A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations*, 2020. 110
- S. Sharma, A. Suresh, R. Ramesh, and B. Ravindran. Learning to factor policies and action-value functions: Factored action space representations for deep reinforcement learning. *arXiv preprint arXiv:1705.07269*, 2017. 111
- M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, pages 894–906. PMLR, 2022. 30
- D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. PMLR, 2014. 62
- D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016. 29
- D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 550(7676): 354–359, 2017. 2, vii
- P. K. Singh, P. K. D. Pramanik, A. K. Dey, and P. Choudhury. Recommender systems: an overview, research trends, and future directions. *International Journal of Business and Systems Research*, 15(1):14–52, 2021. 2, vii
- A. Smith. The wealth of nations [1776], 1937. 102
- K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International conference on machine learning*, pages 5887–5896. PMLR, 2019. 58, 59, 70, 85, 86, 135
- K. Son, S. Ahn, R. D. Reyes, J. Shin, and Y. Yi. Qtran++: Improved value transformation for cooperative multi-agent reinforcement learning, 2020. 60, 97
- N. Spanoudakis and P. Moraitis. Using aseme methodology for model-driven agent systems development. In *International Workshop on Agent-Oriented Software Engineering*, pages 106–127. Springer, 2010. 103, 112
- B. C. Stadie, S. Levine, and P. Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015. 25, 27
- M. Strens. A bayesian framework for reinforcement learning. In *ICML*, volume 2000, pages 943–950, 2000. 28

- S. Sukhbaatar, A. Szlam, and R. Fergus. Learning multiagent communication with backpropagation. *arXiv preprint arXiv:1605.07736*, 2016. 56
- S. Sukhbaatar, Z. Lin, I. Kostrikov, G. Synnaeve, A. Szlam, and R. Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*, 2017. 27
- S. Sukhbaatar, E. Denton, A. Szlam, and R. Fergus. Learning goal embeddings via self-play for hierarchical reinforcement learning. *arXiv preprint arXiv:1811.09083*, 2018. 110
- C. Sun, W. Liu, and L. Dong. Reinforcement learning with task decomposition for cooperative multiagent systems. *IEEE transactions on neural networks and learning systems*, 2020. 112
- M. Sun, S. Devlin, J. Beck, K. Hofmann, and S. Whiteson. Monotonic improvement guarantees under non-stationarity for decentralized ppo. *arXiv preprint arXiv:2202.00082*, 2022. 146
- P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. 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*, pages 2085–2087. International Foundation for Autonomous Agents and Multiagent Systems, 2018. 57, 70, 72, 85, 135, 136
- I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014. 3
- R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL <http://incompleteideas.net/book/the-book-2nd.html>. 13, 14
- R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour, et al. Policy gradient methods for reinforcement learning with function approximation. In *NIPs*, volume 99, pages 1057–1063. Citeseer, 1999a. 15
- R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2): 181–211, 1999b. 110
- A. C. Tam, N. C. Rabinowitz, A. K. Lampinen, N. A. Roy, S. C. Chan, D. Strouse, J. X. Wang, A. Banino, and F. Hill. Semantic exploration from language abstractions and pretrained representations. *arXiv preprint arXiv:2204.05080*, 2022. 23, 28
- A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4), 2017. 54
- M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993. 51, 85, 102, 134, 136

- H. Tang, R. Houthoofd, D. Foote, A. Stooke, O. Xi Chen, Y. Duan, J. Schulman, F. DeTurck, and P. Abbeel. #Exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)* 30, pages 2753–2762. 2017. 25, 26, 83
- Y. Tang and A. Kucukelbir. Variational deep q network. *arXiv preprint arXiv:1711.11225*, 2017. 29
- G. Tennenholtz and S. Mannor. The natural language of actions. In *International Conference on Machine Learning*, pages 6196–6205, 2019. 111
- J. K. Terry, N. Grammel, S. Son, and B. Black. Parameter sharing for heterogeneous agents in multi-agent reinforcement learning. *arXiv preprint arXiv:2005.13625*, 2020. 139
- G. Tesauro. Extending q-learning to general adaptive multi-agent systems. In *Advances in neural information processing systems*, page None. Citeseer, 2003. 52
- V. Thomas, E. Bengio, W. Fedus, J. Pondard, P. Beaudoin, H. Larochelle, J. Pineau, D. Precup, and Y. Bengio. Disentangling the independently controllable factors of variation by interacting with the world. *arXiv preprint arXiv:1802.09484*, 2018. 110
- K. Tumer and A. Agogino. Distributed agent-based air traffic flow management. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems, AAMAS '07*, pages 1–8, New York, NY, USA, May 2007. Association for Computing Machinery. ISBN 978-81-904262-7-5. doi: 10.1145/1329125.1329434. URL <https://doi.org/10.1145/1329125.1329434>. 136
- K. Tuyls and G. Weiss. Multiagent Learning: Basics, Challenges, and Prospects. *AI Magazine*, 33(3):41–41, Sept. 2012. ISSN 2371-9621. doi: 10.1609/aimag.v33i3.2426. URL <https://www.aaai.org/ojs/index.php/aimagazine/article/view/2426>. Number: 3. 134
- A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, 29, 2016. 26
- H. Van Hasselt and M. A. Wiering. Using continuous action spaces to solve discrete problems. In *2009 International Joint Conference on Neural Networks*, pages 1149–1156. IEEE, 2009. 111
- H. van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the 13th AAAI Conference on Artificial Intelligence*, pages 2094–2100, 2016. URL <https://arxiv.org/pdf/1509.06461.pdf>. 14, 54, 79
- B. Vasilev, T. Gupta, B. Peng, and S. Whiteson. Semi-on-policy training for sample efficient multi-agent policy gradients. *arXiv preprint arXiv:2104.13446*, 2021. 62
- A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017. 3, 60, 62

- A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1703.01161*, 2017. 110
- A. S. Vezhnevets, Y. Wu, R. Leblond, and J. Z. Leibo. Options as responses: Grounding behavioural hierarchies in multi-agent rl. In *Proceedings of the 37th International Conference on Machine Learning*. JMLR. org, 2020. 111
- O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser, et al. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017. 141
- J. Wang, Z. Ren, T. Liu, Y. Yu, and C. Zhang. Qplex: Duplex dueling multi-agent q-learning. *arXiv preprint arXiv:2008.01062*, 2020a. 60, 85, 100, 115, 168
- J. Wang, Z. Ren, B. Han, J. Ye, and C. Zhang. Towards understanding cooperative multi-agent q-learning with value factorization. *Advances in Neural Information Processing Systems*, 34:29142–29155, 2021. 131
- T. Wang, H. Dong, V. Lesser, and C. Zhang. Roma: Multi-agent reinforcement learning with emergent roles. In *Proceedings of the 37th International Conference on Machine Learning*, 2020b. 65, 103, 110, 113, 115
- T. Wang, J. Wang, Y. Wu, and C. Zhang. Influence-based multi-agent exploration. In *International Conference on Learning Representations*, 2020c. URL <https://openreview.net/forum?id=BJgy96EYvr>. 61, 83, 102
- W. Wang, T. Y. Y. Liu, J. Hao, X. Hao, Y. Hu, Y. Chen, C. Fan, and Y. Gao. Action semantics network: Considering the effects of actions in multiagent systems. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020d. 112, 115
- Y. Wang, B. Han, T. Wang, H. Dong, and C. Zhang. Off-policy multi-agent decomposed policy gradients. *arXiv preprint arXiv:2007.12322*, 2020e. 63, 131
- Y. Wang, H. He, C. Wen, and X. Tan. Truly Proximal Policy Optimization. *arXiv:1903.07940 [cs, stat]*, Jan. 2020f. URL <http://arxiv.org/abs/1903.07940>. arXiv: 1903.07940. 137
- Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003, 2016. 54, 60
- D. Warde-Farley, T. Van de Wiele, T. Kulkarni, C. Ionescu, S. Hansen, and V. Mnih. Unsupervised control through non-parametric discriminative rewards. In *International Conference on Learning Representations*, 2018. 110
- C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992. 13, 14
- L. Weaver and N. Tao. The optimal reward baseline for gradient-based reinforcement learning. *arXiv preprint arXiv:1301.2315*, 2013. 15

- E. Wei and S. Luke. Lenient Learning in Independent-Learner Stochastic Cooperative Games. *Journal of Machine Learning Research*, 17(84):1–42, 2016. ISSN 1533-7928. URL <http://jmlr.org/papers/v17/15-417.html>. 55, 96, 135
- E. Wei, D. Wicke, D. Freelan, and S. Luke. Multiagent soft q-learning. *arXiv preprint arXiv:1804.09817*, 2018. 59, 70, 71
- E. Wei, D. Wicke, and S. Luke. Multiagent adversarial inverse reinforcement learning. In *AAMAS, AAMAS '19*, pages 2265–2266, 2019. 70
- M. Weiser. The computer for the 21 st century. *Scientific american*, 265(3):94–105, 1991. 2
- C. Wen, X. Yao, Y. Wang, and X. Tan. Smix (λ): Enhancing centralized value functions for cooperative multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7301–7308, 2020. 60
- S. Whitehead. A Complexity Analysis of Cooperative Mechanisms in Reinforcement Learning. pages 607–613, Jan. 1991. 136
- A. Wilson, A. Fern, and P. Tadepalli. Bayesian policy search for multi-agent role discovery. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010. 112
- D. H. Wolpert and K. Tumer. Optimal payoff functions for members of collectives. In *Modeling complexity in economic and social systems*, pages 355–369. World Scientific, 2002. 62
- M. Wooldridge, N. R. Jennings, and D. Kinny. The gaia methodology for agent-oriented analysis and design. *Autonomous Agents and multi-agent systems*, 3(3):285–312, 2000. 112
- J. Yang, I. Borovikov, and H. Zha. Hierarchical cooperative multi-agent reinforcement learning with skill discovery. *arXiv preprint arXiv:1912.03558*, 2019. 64, 111, 115
- Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang. Mean field multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 5571–5580. PMLR, 2018. 61
- Y. Yang, J. Hao, G. Chen, H. Tang, Y. Chen, Y. Hu, C. Fan, and Z. Wei. Q-value path decomposition for deep multiagent reinforcement learning. In *International Conference on Machine Learning*, pages 10706–10715. PMLR, 2020a. 60
- Y. Yang, J. Hao, B. Liao, K. Shao, G. Chen, W. Liu, and H. Tang. Qatten: A general framework for cooperative multiagent reinforcement learning. *arXiv preprint arXiv:2002.03939*, 2020b. 60
- Y. Yang, Y. Wen, J. Wang, L. Chen, K. Shao, D. Mguni, and W. Zhang. Multi-agent determinantal q-learning. In *International Conference on Machine Learning*, pages 10757–10766. PMLR, 2020c. 65
- C. Yu, A. Velu, E. Vinitzky, J. Gao, Y. Wang, A. Bayen, and Y. Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022. xix, 137, 143

- L. Yu, J. Song, and S. Ermon. Multi-agent adversarial inverse reinforcement learning. In *International Conference on Machine Learning*, pages 7194–7201. PMLR, 2019. 72
- C. Zhang and V. Lesser. Coordinated multi-agent reinforcement learning in networked distributed pomdps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, 2011. 5
- C. Zhang, V. R. Lesser, and S. Abdallah. Self-organization for coordinating decentralized reinforcement learning. In *AAMAS*, volume 10, pages 739–746, 2010. 110
- Q. Zhang, M. K. Burrage, E. Lukaschuk, M. Shanmuganathan, I. A. Popescu, C. Nikolaidou, R. Mills, K. Werys, E. Hann, A. Barutcu, et al. Toward replacing late gadolinium enhancement with artificial intelligence virtual native enhancement for gadolinium-free cardiovascular magnetic resonance tissue characterization in hypertrophic cardiomyopathy. *Circulation*, 144(8):589–599, 2021. 1
- S. Zheng and Y. Yue. Structured exploration via hierarchical variational policy networks, 2018. URL <https://openreview.net/forum?id=HyunpgbR->. 61, 83
- Z. Zheng, J. Oh, and S. Singh. On learning intrinsic rewards for policy gradient methods. *arXiv preprint arXiv:1804.06459*, 2018. 63
- C. Zhou, Q. Li, C. Li, J. Yu, Y. Liu, G. Wang, K. Zhang, C. Ji, Q. Yan, L. He, et al. A comprehensive survey on pretrained foundation models: A history from bert to chatgpt. *arXiv preprint arXiv:2302.09419*, 2023. 25
- M. Zhou, Y. Chen, Y. Wen, Y. Yang, Y. Su, W. Zhang, D. Zhang, and J. Wang. Factorized q-learning for large-scale multi-agent systems. In *Proceedings of the First International Conference on Distributed Artificial Intelligence*, pages 1–7, 2019. 56
- M. Zhou, Z. Liu, P. Sui, Y. Li, and Y. Y. Chung. Learning implicit credit assignment for cooperative multi-agent reinforcement learning. 2020. 63
- H. Zhu and M. Zhou. Role-based multi-agent systems. In *Personalized Information Retrieval and Access: Concepts, Methods and Practices*, pages 254–285. Igi Global, 2008. 112