

ABSTRACT

Title of Dissertation: **LEARNING WITH LESS EFFORT:
EFFICIENT TRAINING AND GENERALIZATION
IN (MULTI-)ROBOT SYSTEMS**

Peihong Yu
Doctor of Philosophy, 2025

Dissertation Directed by: **Professor Pratap Tokekar**
Department of Computer Science

Multi-robot systems are becoming essential in applications ranging from warehouse automation to search and rescue, offering advantages in speed, coverage, and capability compared to single robots. However, getting multiple robots to learn and coordinate effectively remains challenging — training robots to work together requires extensive effort and often fails to generalize beyond training conditions. This dissertation addresses two fundamental challenges in multi-robot learning: reducing the training effort required and improving generalization to reduce policy retraining. First, we propose methods to leverage more easily obtainable data to facilitate the learning — using human-drawn sketches as an alternative to costly teleoperated demonstrations for manipulation tasks, and utilizing individual robot demonstrations instead of harder-to-obtain joint multi-robot demonstrations for learning collaborative behaviors. Second, we develop techniques to help learned policies adapt to new scenarios without retraining — introducing frameworks that maintain coordination under different observation conditions and

enable effective information sharing across varying initial state distributions. Building on these completed works, we propose to tackle zero-shot coordination with new teammates and generalization to diverse opponent strategies, aiming to create multi-robot systems that can quickly adapt to new partners and adversaries without extensive retraining.

LEARNING WITH LESS EFFORT:
EFFICIENT TRAINING AND GENERALIZATION IN (MULTI-)ROBOT
SYSTEMS

by

Peihong Yu

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2025

Advisory Committee:

Professor Pratap Tokekar, Chair/Advisor
Professor Nikhil Chopra, Dean's Representative
Professor Abhinav Shrivastava
Professor Kaiqing Zhang
Professor Ruohan Gao

© Copyright by
Peihong Yu
2025

Acknowledgments

Completing this dissertation has been a long journey, and I'm deeply grateful to everyone who supported me along the way.

First and foremost, I would like to thank my advisor, Professor Pratap Tokekar. I still remember receiving his email inviting me to join a research project while I was on a train to the airport, flying back to the U.S. just before the pandemic. At the time, I was struggling with some remaining graduation requirements from my master's, and that message — sent after I had just finished a course with him — gave me a much-needed boost of confidence.

Over the years, Pratap has been consistently supportive. He always encouraged me to follow my curiosity and pursue what I wanted to do. And he has been remarkably patient, giving me the time and space to grow. I submitted my first research paper only after two years of working with him, and he never rushed me. In fact, he always tells us that it is our growth that matters more than publications. His support and the freedom he gave me allowed me to explore, make mistakes, and grow at my own pace. Over time, I became a more confident and independent researcher, capable of taking full ownership of my work.

I would also like to thank my committee members: Professor Nikhil Chopra, Professor Abhinav Shrivastava, Professor Kaiqing Zhang, and Professor Ruohan Gao. I am also grateful to Professor Furong Huang and Professor Tianyi Zhou, who served on my prelim committee. Thank you all for your time, thoughtful feedback, and steady support throughout my Ph.D. Your

insights helped shape the direction of this dissertation and pushed me to think more deeply about my work. I truly appreciate your generosity and commitment, especially during the final stages of this process.

Special thanks to Professor Furong Huang, one of the few female faculty members in our department. I always felt a natural sense of connection and warmth around her, which led me to seek her guidance when I was eager for change. Her words reminded me of my agency and worth when I had started to doubt both, and her encouragement helped me take important steps forward. I will always be thankful for that.

I would also like to thank Dr. Amrit Singh Bedi, who mentored me early on and gave me a lot of help when I was just getting started. I'm also grateful to my industry mentors, Dr. True Price at Meta and Dr. Aswin Raghavan at SRI International, for their guidance and support during my internships. I learned a great deal from working with each of them.

I am grateful to my colleagues and friends in the RAAS Lab, including Dr. Guangyao Shi, Dr. Vishnu Dutt Sharma, Dr. Harnaik Dhami, Amisha Bhaskar, Anukriti Singh, Rui Liu, Chak Lam Shek, Joshua McClellan, Kasra Torshizi, Souradip Chakraborty, Zahir Mahammad, Ruthwik Dasyam, Sahire Ellahy, Dr. Troi Williams, and Dr. Nare Karapetyan, for making this journey more enjoyable and for fostering such a collaborative and welcoming environment. I especially want to thank my closest collaborators: Amisha Bhaskar, Anukriti Singh, and Manav Mishra, who visited our lab as a research student. They have been a constant source of support, insight, and encouragement. I truly learned a lot from everyone.

I'm also deeply grateful to the friends I got to know during my Ph.D., including Shuaiyi Huang, Ruijie Zheng, Siyao Li, Gengjia Zhang, Yuqing Quan, Weini Wang, Yingqing Guo, Lanxuan Zhao, Boao Zhao, Jingxi Chen, Yue Feng, Yiran Xu, Hanyu Wang, Yixuan Ren, Bo He,

Mingyang Xie, Yue Jiang, Duotun Wang, Yuexi Chen, Jingling Li, Yanchao Sun, and Chen Zhu. Some of you I met at school, others through shared housing or unexpected connections in life — but all of you have been an important part of this journey. Thank you for the laughter, support, and all the joyful moments that made these years more than just about research. I want to especially thank Shuaiyi Huang for being my close friend and the best roommate. From late-night research chats to pushing each other toward better routines, from prelim prep to job hunts to defense week stress, you were always there. I couldn't have asked for a better partner through it all, and I'm so grateful that we'll be starting the next chapter together in the Bay Area.

I'm endlessly thankful to my family for their love and steady support throughout the years. To my parents, Zhimin Wang and Haichao Yu, thank you for always trusting me and allowing me to make my own choices. To my younger brother, Peixin Yu, thank you for growing alongside me in your own way. I also miss my grandma, Qingxiang Zhao, deeply. Her kindness and strength continue to stay with me, and I often think of her.

Finally, to Chen Chen, my husband and my partner in everything, thank you for being by my side through it all. I feel incredibly lucky that we were able to go through our Ph.D. journeys together. You have helped me stay grounded through the difficult moments and supported me in more ways than I can name. Thank you for your steady presence, for always taking care of me, for being my most dependable support, and for being the best cook and my best friend. Even though our research focuses differ, you have always listened with care, asked thoughtful questions, and offered the kind of quiet support that mattered most. I could not have done this without you.

I know there are many others who have supported me in ways big and small along the way. If I've missed mentioning you by name, please know that I'm still truly grateful. Your presence and kindness have meant a lot to me.

Table of Contents

Acknowledgements	ii
Table of Contents	v
List of Tables	viii
List of Figures	ix
Chapter 1: Introduction	1
1.1 Research Contribution	4
1.2 Dissertation Outline	9
Chapter 2: Efficient Policy Training with Human-Drawn Trajectory Sketches	10
2.1 Introduction	10
2.2 Related Work	12
2.3 SKETCH-TO-SKILL: Bootstrapping Robot Learning with Human Drawn Trajectory Sketches	14
2.3.1 Sketch-to-3D Trajectory Generator	14
2.3.2 Policy Learning	17
2.4 Experiments	20
2.4.1 Evaluation of the Sketch-to-3D Trajectory Generator	21
2.4.2 Comparisons with Baselines	24
2.4.3 Ablation Studies	27
2.4.4 Hardware Experiments	29
2.5 Conclusions and Future Work	31
2.6 Appendix	32
2.6.1 De Boor’s Algorithm Details	32
2.6.2 Implementation Details and Hyperparameters of Sketch-to-3D Trajectory Generator	33
2.6.3 Implementation Details and Hyperparameters of SKETCH-TO-SKILL Policy and baselines	35
2.6.4 Additional Details of Real-world Experiments	36
2.6.5 Additional Details of Robomimic Experiments	39
2.6.6 Additional Experiments in Metaworld Assembly	40
2.6.7 Using Color Gradients for Overlapping Trajectories	42

Chapter 3:	Efficient Multi-Agent Reinforcement Learning with Personalized Expert Guidance	44
3.1	Introduction	44
3.2	Related Works	46
3.3	Preliminaries	48
3.4	MARL with Personalized Expert Demonstrations	51
3.4.1	Formulating Personalized Expert-Guided MARL	51
3.4.2	Solving Personalized Expert-Guided MARL	54
3.5	Experiments	57
3.5.1	Main Results with Personalized Demonstrations	58
3.5.2	Results with Joint Demonstrations	61
3.6	Conclusions	63
3.7	Appendix	65
3.7.1	Environment Details	65
3.7.2	Demonstration Details	67
3.7.3	Network architecture and hyperparameters	70
3.7.4	Additional Visualizations	72
Chapter 4:	Efficient Multi-Agent Reinforcement Learning with Demonstrations from Various Team Sizes	74
4.1	Introduction	74
4.2	Method	76
4.2.1	Background: Dec-POMDP with Entities	76
4.2.2	DynPegMARL: Extending Demonstration-Guided Learning Across Team Sizes with GNN-Based Observation Encoding	77
4.3	Experiments	79
4.3.1	Main Results	79
4.3.2	Ablation Study	81
4.4	Conclusion	82
Chapter 5:	Generalization across Sight Range through Task-Agnostic Contrastive Pre-training	84
5.1	Introduction	84
5.2	Related Work	87
5.3	Preliminaries	90
5.4	TACTIC: Towards task-agnostic adaptive communication in MARL	92
5.4.1	Offline Training of Communication Mechanism	92
5.4.2	Online Training of Agent Policy	95
5.5	Experiments	96
5.5.1	Experimental Setup	97
5.5.2	Policy Generalization Across Sight Ranges	98
5.5.3	Online Policy Training Efficiency	101
5.5.4	Ablation Study	102
5.6	Conclusion	103

Chapter 6:	Generalization across Initial States through Common Operating Picture In-	105
	tegration	
6.1	Introduction	105
6.2	Related Works	107
6.3	Preliminaries	108
6.4	Enhancing Multi-Agent Coordination through COP Integration	109
6.5	Experiments	111
6.6	Appendix	113
6.6.1	Motivating Example in Details	113
6.6.2	Pipelines of QMIX-based Baselines	115
6.6.3	Visualization of OOD Laydowns	115
6.6.4	Evaluate Message Dropping	117
6.6.5	QMIX with Varying Observation Sight Ranges	117
6.6.6	Hallucination	118
6.6.7	Additional Evaluation Details	119
Chapter 7:	Conclusion and Future Work	122
7.1	Summary of Contribution	122
7.2	Limitations and Areas for Improvement	123
7.3	Future Research Directions	124
Bibliography		126

List of Tables

2.1	Performance Metrics for generator model	23
2.2	Hyperparameters for RL in SKETCH-TO-SKILL.	36
2.3	Performance Metrics for Overlapping and Non-Overlapping Trajectories	43
3.1	The details of suboptimal demonstrations for the gridworld environment.	68
3.2	MAPPO Hyperparamters	71
3.3	DM2’s IPPO Hyperparameters	71
3.4	DM2’s GAIL Hyperparameters	71
6.1	Evaluations results on <i>1o_10b_vs_1r</i> over 20,000 episodes.	119
6.2	Comparison of our algorithm with different embedding sizes on <i>1o_10b_vs_1r</i> over 20,000 episodes.	120
6.3	Evaluation results of various algorithms on <i>1o_10b_vs_1r</i> over 20,000 episodes. .	120
6.4	Evaluation results of various algorithms on <i>1o_2r_vs_4r</i> over 20,000 episodes. . .	121
6.5	Evaluation results of various algorithms on <i>3s_vs_5z</i> over 20,000 episodes.	121

List of Figures

1.1	Overview of dissertation contributions organized around two complementary goals: reducing initial training effort (left) and improving generalization to reduce policy retraining (right).	3
1.2	Overview of the SKETCH-TO-SKILL framework for learning robotic manipulation from human-drawn 2D sketches.	5
1.3	PegMARL enables learning cooperative behaviors from individual agent demonstrations rather than requiring joint multi-robot demonstrations.	6
1.4	TACTIC uses contrastive learning to enable agents to effectively coordinate across different observation ranges through adaptive communication.	8
1.5	The Common Operating Picture (COP) approach enables robust multi-agent coordination through shared situational awareness.	9
2.1	Learning a new skill in the SKETCH-TO-SKILL framework. Step 1: Capture the task scenario from two views and collect human-drawn sketches. Step 2: Convert 2D sketches to 3D trajectories using a pretrained generator. Step 3: Execute generated trajectories to collect experience data. Step 4: Learn manipulation policy using reinforcement learning bootstrapping from behavior cloning and using guidance for experience data.	11
2.2	The Sketch-to-3D Trajectory Generator takes dual-view 2D sketches as inputs and predicts B-spline parameters to generate the final 3D trajectory output. . . .	15
2.3	Overview of SKETCH-TO-SKILL integrating sketch-generated demonstrations with reinforcement learning. Sketch-generated experiences train an IL policy, which bootstraps the RL process. A discriminator guides exploration by rewarding similarity to sketch-generated trajectories. The final action, combining IL and RL policy outputs, further enhances the exploration guidance. The asterisk after "Replay Buffer" indicates that the buffer is initialized with the open-loop serving demonstrations.	17
2.4	Multi-stage trajectory generation and execution. On the left, we show hand-drawn sketches on scenario RGB images and the extracted sketches on a blank background, (a) generated trajectory from the Sketch-to-3D Trajectory Generator, and (b) executed trajectory via open-loop serving. In (c), we visualize a teleoperated demo for the same task for reference.	22
2.5	Latent space interpolation results showing smooth transitions between original samples (leftmost and rightmost) and reconstructed samples (middle) from two viewpoints, demonstrating the model's ability to generate coherent 3D trajectories from 2D sketch pairs.	23

2.6	Evaluation Scores (success rate) for the robomimic PickPlaceCan environment during evaluation.	25
2.7	Evaluation Performance of SKETCH-TO-SKILL in MetaWorld This figure shows the success rate across six MetaWorld tasks under randomized initial gripper and object positions. SKETCH-TO-SKILL achieves comparable performance to IBRL while surpassing pure RL.	26
2.8	Behavioral Cloning (BC) scores using actual teleoperated data and sketch generated demonstrations. The blue bars represent the baseline BC policy trained with 3 high-quality demonstrations, while the red bars show BC policies trained with sketch-generated demonstrations, varying in the number of demonstrations m per input sketch pair (1, 3, 5, and 10). Darker shades of red indicate an increase in the number of sketch-based demonstrations used for training. Despite poor success rate, the actual trajectories and policy learned with sketches are useful for bootstrapping as evidenced by the training performance (Figures 2.7).	27
2.9	This figure illustrates ablation training scores for SKETCH-TO-SKILL with varying m , the number of demonstrations generated per sketch pair (1, 3, 5, and 10).	28
2.10	Reward weighting term ablation	28
2.11	Complete setup for the ButtonPress task in a real-world experiment. The configuration includes a UR3e robot arm equipped with a Robot Hand gripper, and a RealSense D435i camera mounted on the wrist. Two additional RealSense cameras are positioned orthogonally to capture the trajectory from two different viewpoints.	29
2.12	Training curves and hardware setups for real-world robotic tasks: ButtonPress, ToastButtonPress, and ToastPickPlace.	30
2.13	Diversity in generated 3D trajectories. Each subplot shows multiple generated trajectories (colored lines) for a single input, demonstrating variability. Scattered points represent the ground truth trajectory.	34
2.14	Latent space interpolation results showing smooth transitions between original samples (leftmost and rightmost) and reconstructed samples (middle) from two viewpoints, demonstrating the model’s ability to generate coherent 3D trajectories from 2D sketch pairs.	35
2.15	Left: sketches for the MetaWorld Assembly task. Right: training scores (success rate).	40
2.16	Trajectories with time-based color gradient for Assembly Metaworld simulation task.	42
3.1	Joint demonstrations are costly to collect but offer rich information on collaborative behaviors. Personalized demonstrations are easier to collect, but solely focus on individual agent goals, so they lack cooperative elements.	45

3.2	An example of utilizing personalized demonstrations to learn cooperative multi-agent policies. To learn successful cooperation, the agents are required not only to imitate the demonstrations to achieve personal goals but also to learn how to avoid conflicts and collaborate. We visualize the state visitation frequency of the personalized demonstrations and the joint policies learned by our algorithm and MAPPO, where a darker color means a higher value. We observe that the demonstrations guide the agents in exploring the state space more efficiently than in MAPPO.	49
3.3	When joint demonstrations are sampled from co-trained policies, the agents' behaviors exhibit compatibility. In contrast, personalized demonstrations solely focus on how each agent achieves its individual goal and lack cooperative elements, potentially leading to conflicts.	52
3.4	A motivating example illustrating the imitation of personalized demonstrations in a multi-agent environment. The primary technical challenge lies in the discrepancy between the transition dynamics in the personalized MDP and the local transition dynamics for each agent in the multi-agent environment.	53
3.5	The discrete gridworld environments. Circles indicate the starting locations of the agents, while squares of the same color denote their respective goal locations. More details can be found in Appendix 3.7.1.1.	57
3.6	Learning curves of PegMARL versus other baseline methods under the lava scenario . PegMARL converges to higher rewards and generalizes better to larger numbers of agents. The star symbols (*) in the legend indicate that suboptimal personalized demonstrations are adopted.	59
3.7	Learning curves of PegMARL versus other baseline methods under the door scenario . PegMARL shows better robustness in terms of convergence and generalizability.	59
3.8	(a) The modified cooperative navigation scenario. (b) The learning curves demonstrate that PegMARL is effective in continuous environments.	61
3.9	Learning curves of PegMARL versus DM ² in two tasks under the SMAC scenarios . The suffix "diff" in the legend indicates that the joint demonstrations used are sampled from non-co-trained policies. Otherwise, the demonstrations are sampled from co-trained policies.	62
3.10	An example of personalized demonstrations for the lava scenario (we did not visualize all the optimal paths). There is only one agent in the environment. The agent may take both the left and the right path toward the goal.	68
3.11	Agent policies and state occupancy measures estimated from the suboptimal demonstrations for the lava scenario	68
3.12	Agent policies and state occupancy measures estimated from the suboptimal demonstrations for the door scenario . The top row is for the red agent, and the bottom row is for the green agent.	69
3.13	An illustration of personalized demonstrations for the cooperative navigation task. There is only one agent in the environment. The agent's objective is to reach one of the two goal locations.	69
3.14	Ablation study over the weighting term η on the lava environment using optimal personalized demonstrations.	72

3.15	Ablation study over the weighting term η on the 5m_vs_6m SMAC map. Left: results using joint demonstrations sampled from co-trained policies; Right: results using joint demonstrations sampled from non-co-trained policies.	72
3.16	State visitation frequency of the joint policies learned by PegMARL (with sub-optimal demonstrations) and MAPPO for the lava scenario . The darker color means a higher value. MAPPO failed to learn any meaningful policies in all three settings.	73
3.17	State visitation frequency of the joint policies learned by PegMARL (with sub-optimal demonstrations) and MAPPO for the door scenario . The darker color means a higher value. MAPPO converges to a suboptimal policy in the easy case, while failing to learn in the hard case.	73
4.1	The network structure of DynPegMARL. Suppose there are N agents and M enemies in the team.	78
4.2	Training performance of MAPPO, MAPPO-GNN, DynPegMARL, and DynPegMARL-GNN across three Protoss scenarios in SMACv2 with different team sizes: 5v5, 10v10, and 20v20. DynPegMARL-GNN consistently achieves faster convergence and higher final performance by incorporating demonstration data and using GNN-based policy networks. Curves show the mean and standard deviation over 5 runs.	80
4.3	Training performances of DynPegMARL-GNN across different demonstration sources.	81
4.4	Training performances of DynPegMARL-GNN across different demonstration qualities.	82
5.1	TACTIC utilizes contrastive learning to align the integration of local observations o_i and messages $\{m_{ji}\}$ with the full egocentric state $\hat{s}_i$ for each agent i , enabling agents to "see" beyond their limited sight ranges through communication.	84
5.2	QMIX-ATT and TACTIC's performances on Protoss 10v10 from SMACv2 [1] with varying sight ranges (SRs). Different SRs are achieved by applying different sight-range ratios (SRRs) to the agents' original SRs in the implementation. Policies trained at SRR=0.2, 1, and 5 are tested across a broader set of SRRs. QMIX-ATT Struggles to generalize to unseen SRs, while TACTIC generalizes much better.	86
5.3	The offline training pipeline of the adaptive communication mechanism(Section 5.4.1). It includes three key components: an egocentric state encoder, an adaptive message generator, and a message-observation integrator. The training pipeline consists of two contrastive learning processes: Global Information Alignment (GIA) for aligning the features generated from the egocentric state encoder across all agents and timesteps, and Feature Integration Alignment (FIA) for aligning features from the message-observation integrator and the egocentric state encoder on an individual agent. Two auxiliary loss functions are introduced in the total loss function to enhance training: a deconstruction loss for learning to recover the egocentric states and a dynamic loss for learning temporally coherent representations.	93

5.4	Online policy training pipeline of TACTIC, illustrating the integration of QMIX architecture with pre-trained communication components. The pre-trained message generator (Mess Gen) and message-observation integrator (Mess-obs Integrator) remain fixed during the policy training.	96
5.5	Performance of TACTIC and baseline models on policy generalizability across various sight ranges in the Protoss map.	99
5.6	Performance of TACTIC and baseline models on policy generalizability across various sight ranges in the Terran map.	99
5.7	Performance of TACTIC and baseline models on policy generalizability across various sight ranges in the Zerg map.	100
5.8	Training curves of the online policy learning stage in TACTIC and baseline models under different environment setups.	101
5.9	Performance of TACTIC on policy generalizability across various sight ranges under 5 different offline training schemas. The environment setup is the Terran map with a 5v5 agent configuration. (a) Offline training with a <code>Exploratory</code> dataset; (b) Offline training with a <code>Random</code> dataset; (c) Offline training without L_{recon} ; (d) Offline training without L_{dyn} ; (e) Offline training without L_{recon} and L_{dyn}	102
6.1	The purple and blue agents observe each other, the orange agent is beyond visual range. This leads to multiple potential COPs.	106
6.2	Motivating COPs in MARL.	107
6.3	The pipeline of our method. (a) The overall architecture. (b) Information integration and policy learning. (c) COP formation module.	109
6.4	Masking in the formation of egocentric COP. v is the visual range of the agent and m is the map size.	110
6.5	Comparison of methods on OOD initial states	112
6.6	Training curves of various methods	112
6.7	Varying message size (<code>1o_10b_vs_1r</code>).	113
6.8	MSE vs win rate (<code>1o_2r_vs_4r</code>).	113
6.9	An example of reconstructed COP (<code>1o_10b_vs_1r</code>).	113
6.10	Pipelines of QMIX-based baseline algorithms.	115
6.11	Illustration of modified initial distributions for <code>1o_10b_vs_1r</code> , <code>1o_2r_vs_4r</code> and <code>3s_vs_5z</code> . In this figure, "at" means that all indicated agents are generated at the same point P , while "in" denotes that all indicated agents are randomly generated at distinct locations within the region R	116
6.12	Evaluation results of our algorithm with and without communication.	117
6.13	Training and evaluation results of QMIX with different sight ranges on <code>1o_10b_vs_1r</code> . A larger sight range gives us faster convergence speed and a higher win rate, which indicates that the agent can make wiser decisions if it is more aware of the global state under SMAC scenarios.	118
6.14	False positives across timesteps. $< v$ means within the field of view, and $> v$ means outside the field of view.	119

Chapter 1: Introduction

Robots are increasingly deployed in real-world applications such as warehouse automation [2], disaster response [3], and manufacturing [4]. This surge in deployment has been enabled by significant advances in both perception and control systems. Deep learning has dramatically improved object detection, scene understanding, and state estimation capabilities [5, 6, 7], while control and planning have benefited from the integration of learning-based approaches with classical methods [8, 9], allowing robots to adapt to complex and dynamic environments that would be challenging to handle with purely traditional techniques.

The learning component in robotics typically involves two main paradigms: Imitation Learning (IL) [10, 11], where robots learn from expert demonstrations, and Reinforcement Learning (RL) [12, 13], where they learn through trial-and-error interaction with the environment. For perception tasks, supervised learning has proven highly effective by leveraging large datasets of labeled images and sensor data. However, for control and planning, the learning landscape is more complex, requiring a combination of classical optimization methods and modern learning techniques to achieve robust performance [14].

As these learning-based control and planning capabilities have matured, robotic applications have grown more sophisticated, making multi-robot systems increasingly essential for tasks that require enhanced speed, coverage, and capability compared to single robots [15, 16, 17].

This has led to the development of Multi-Agent Reinforcement Learning (MARL) [18], which provides a framework for training teams of robots to coordinate and collaborate effectively. However, coordinating multiple agents introduces fundamental complexities that distinguish MARL from single-agent learning, leading to two major challenges that limit practical deployment.

The first major challenge is the substantial training effort required for effective coordination. As the number of robots increases, the state-action space grows exponentially, making exploration extremely difficult — robots need to learn not only their individual tasks but also how to coordinate with teammates. To address this challenge, many existing approaches rely on collecting demonstrations where multiple human experts control the robots simultaneously to show desired coordination patterns [19, 20]. However, such joint demonstrations are time-consuming to collect and must be recollected whenever the team composition changes.

The second major challenge persists even after successful training — learned behaviors often fail to generalize beyond training conditions. Changes in observation conditions, initial states, or team configurations [21] can severely degrade performance. This brittleness forces robots to either undergo constant retraining or operate only in highly controlled environments, significantly limiting their real-world applicability.

To overcome these limitations and make multi-robot learning practical, we need systems that can learn effectively from easier forms of human input, adapt to new scenarios without starting from scratch, and transfer knowledge across tasks and teams. This dissertation explores how to make robotic learning **more efficient and generalizable**, particularly in multi-robot systems. Specifically, we ask: *How can robots learn with less effort — less human supervision, fewer demonstrations, and reduced retraining, while maintaining performance in novel environments?*

We address these fundamental challenges in multi-robot learning through two complemen-

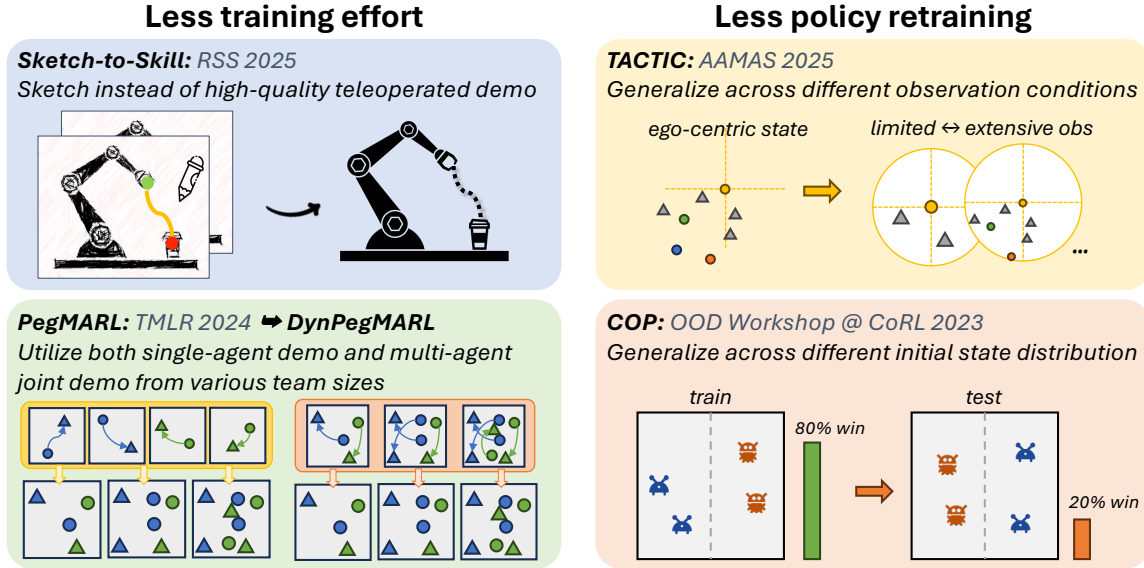


Figure 1.1: Overview of dissertation contributions organized around two complementary goals: reducing initial training effort (left) and improving generalization to reduce policy retraining (right).

tary directions: (1) Reducing initial training effort by developing methods that can learn from simpler and more readily available forms of guidance, making demonstration collection more practical and scalable; (2) Improving generalization to reduce retraining effort by developing frameworks that enable learned policies to maintain coordination across varying conditions.

These advances have the potential to significantly impact real-world robotics applications. Lower training effort means multi-robot systems become accessible to a broader range of users and applications, not just specialized settings with abundant resources. Better generalization capabilities allow robots to adapt to changing conditions without requiring constant retraining, making them more practical for dynamic real-world environments.

1.1 Research Contribution

This dissertation addresses the dual challenges of training effort and generalization in multi-robot learning through four interconnected contributions. Our approach progresses from making individual robot learning more accessible to enabling robust multi-robot coordination, ultimately creating systems that require less human supervision and adapt better to new conditions. Figure 1.1 illustrates how these contributions are organized around two complementary goals: reducing initial training effort (left) and improving generalization to reduce policy retraining (right).

SKETCH-TO-SKILL makes robot learning accessible to non-experts by replacing teleoperated demonstrations with simple 2D sketches. Training robotic manipulation policies traditionally requires numerous demonstrations and/or environmental rollouts. While recent Imitation Learning (IL) and Reinforcement Learning (RL) methods have reduced the number of required demonstrations, they still rely on expert knowledge to collect high-quality data, limiting scalability and accessibility. We propose SKETCH-TO-SKILL, a novel framework that leverages human-drawn 2D sketch trajectories to bootstrap and guide RL for robotic manipulation. Our approach extends beyond previous sketch-based methods, which were primarily focused on imitation learning or policy conditioning, limited to specific trained tasks. SKETCH-TO-SKILL employs a Sketch-to-3D Trajectory Generator that translates 2D sketches into 3D trajectories, which are then used to autonomously collect initial demonstrations. We utilize these sketch-generated demonstrations in two ways: to pre-train an initial policy through behavior cloning and to refine this policy through RL with guided exploration. Experimental results demonstrate that SKETCH-TO-SKILL achieves $\sim 96\%$ of the performance of the baseline model that leverages teleoperated

demonstration data, while exceeding the performance of a pure reinforcement learning policy by $\sim 170\%$, only from sketch inputs. This makes robotic manipulation learning more accessible and potentially broadens its applications across various domains. This work was **accepted at the Robotics: Science and Systems Proceedings (RSS) in 2025 [22]**.

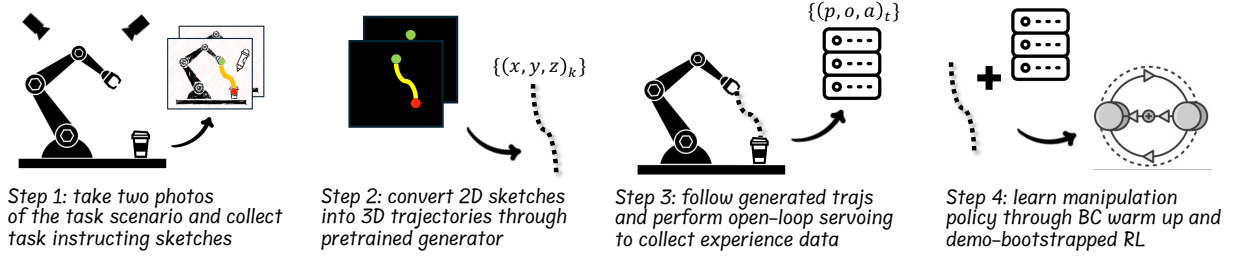


Figure 1.2: Overview of the SKETCH-TO-SKILL framework for learning robotic manipulation from human-drawn 2D sketches.

PegMARL enables multi-robot coordination learning from individual demonstrations rather than requiring joint expert control. Multi-Agent Reinforcement Learning (MARL) algorithms face the challenge of efficient exploration due to the exponential increase in the size of the joint state-action space. While demonstration-guided learning has proven beneficial in single-agent settings, its direct applicability to MARL is hindered by the practical difficulty of obtaining *joint* expert demonstrations. In this work, we introduce a novel concept of *personalized expert demonstrations*, tailored for each individual agent or, more broadly, each individual *type* of agent within a heterogeneous team. These demonstrations solely pertain to single-agent behaviors and how each agent can achieve personal goals without encompassing any cooperative elements, thus naively imitating them will not achieve cooperation due to potential conflicts. To this end, we propose an approach that *selectively* utilizes personalized expert demonstrations as guidance and allows agents to learn to cooperate, namely personalized expert-guided MARL (PegMARL). This algorithm utilizes two discriminators: the first provides incentives based on the alignment of indi-

vidual agent behavior with demonstrations, and the second regulates incentives based on whether the behaviors lead to the desired outcome. We evaluate PegMARL using personalized demonstrations in both discrete and continuous environments. The experimental results demonstrate that PegMARL outperforms state-of-the-art MARL algorithms in solving coordinated tasks, achieving strong performance even when provided with suboptimal personalized demonstrations. We also showcase PegMARL’s capability of leveraging joint demonstrations in the StarCraft scenario and converging effectively even with demonstrations from non-co-trained policies. This work was **accepted at the Transactions on Machine Learning Research (TMLR) in 2024** [23].

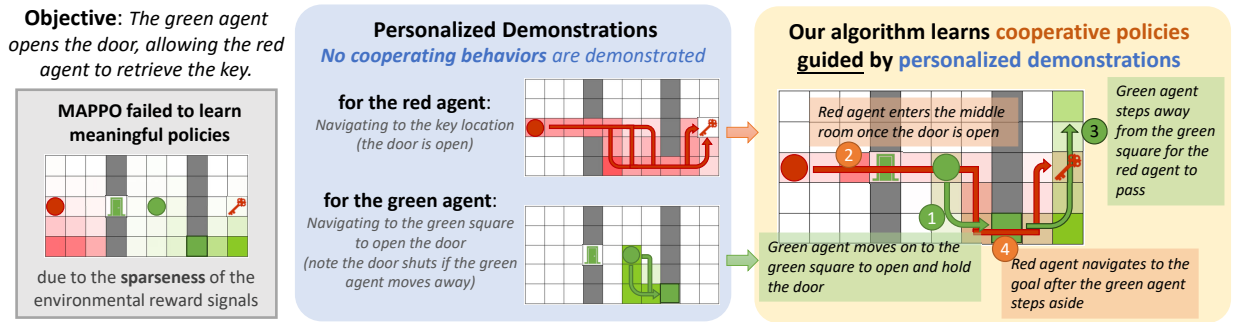


Figure 1.3: PegMARL enables learning cooperative behaviors from individual agent demonstrations rather than requiring joint multi-robot demonstrations.

Building on PegMARL, we develop DynPegMARL, a Graph Neural Network extension that enables learning from demonstrations across different team sizes. This capability is valuable because fundamental coordination patterns — such as collision avoidance, spatial coordination, and formation behaviors — remain consistent and transferable across team scales. While PegMARL can handle personalized and joint demonstrations of the same size effectively, it cannot leverage demonstrations when the team size differs from the target scenario due to mismatched observation dimensions. We address this by developing a GNN-based encoder that pro-

cesses ally and enemy features separately in a permutation-invariant manner, enabling the same architecture to work across different team configurations. This extension significantly improves demonstration reusability by allowing PegMARL to learn from any available demonstration data regardless of the original team configurations.

TACTIC addresses the generalization challenge by enabling coordination across varying observation conditions through adaptive communication. The “sight range dilemma” in cooperative Multi-Agent Reinforcement Learning (MARL) presents a significant challenge: limited observability hinders team coordination, while extensive sight ranges lead to distracted attention and reduced performance. While communication can potentially address this issue, existing methods often struggle to generalize across different sight ranges, limiting their effectiveness. We propose TACTIC, Task-Agnostic Contrastive pre-Training strategy Inter-Agent Communication. TACTIC is an adaptive communication mechanism that enhances agent coordination even when the sight range during execution is vastly different from that during training. The communication mechanism encodes messages and integrates them with local observations, generating representations grounded in the global state using contrastive learning. By learning to generate and interpret messages that capture important information about the whole environment, TACTIC enables agents to effectively “see” more through communication, regardless of their sight ranges. We comprehensively evaluate TACTIC on the SMACv2 benchmark across various scenarios with broad sight ranges. The results demonstrate that TACTIC consistently outperforms traditional state-of-the-art MARL techniques with and without communication, in terms of generalizing to sight ranges different from those seen in training, particularly in cases of extremely limited or extensive observability. This work was **accepted at the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025)** [\[24\]](#).

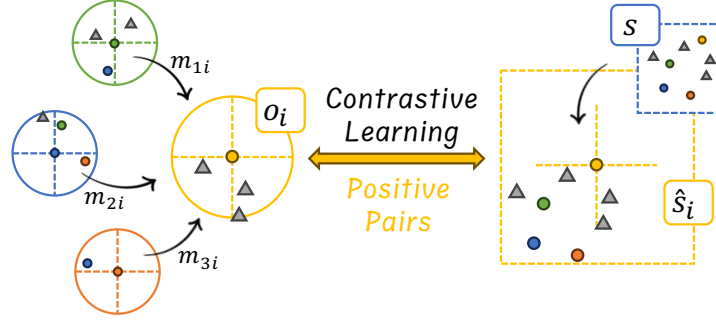


Figure 1.4: TACTIC uses contrastive learning to enable agents to effectively coordinate across different observation ranges through adaptive communication.

COP improves robustness to out-of-distribution conditions by replacing unintelligible communication with interpretable shared representations. In multi-agent systems, agents possess only local observations of the environment. Communication between teammates becomes crucial for enhancing coordination. Past research has primarily focused on encoding local information into embedding messages, which are often unintelligible to humans and can lead to brittle policies when tested on out-of-distribution initial states. We present an approach to multi-agent coordination using a more robust and interpretable representation: the Common Operating Picture (COP). In our method, each agent is equipped with the capability to integrate its (history of) observations, actions, and received messages into a COP and disseminate this representation. This process takes into account the dynamic nature of the environment and the shared mission, creating a more comprehensive and adaptable representation for decision-making. We conducted experiments in the StarCraft2 environment to validate our approach. Our results demonstrate the efficacy of COP integration as a representation method, showing that COP-based training leads to more robust policies compared to state-of-the-art Multi-Agent Reinforcement Learning (MARL) methods when faced with out-of-distribution initial states. This work was accepted at **the First Workshop on Out-of-Distribution Generalization in Robotics at CoRL 2023** [25].

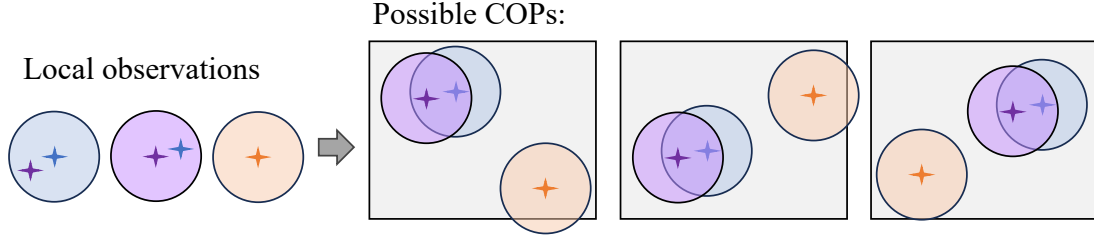


Figure 1.5: The Common Operating Picture (COP) approach enables robust multi-agent coordination through shared situational awareness.

1.2 Dissertation Outline

This dissertation is organized as follows:

- Chapter 2 introduces SKETCH-TO-SKILL, which uses sketch-based inputs to reduce the cost of demonstration collection in manipulation tasks.
- Chapter 3 presents PegMARL, which leverages personalized demonstrations for efficient coordination learning.
- Chapter 4 extends PegMARL with a Graph Neural Network architecture to enable learning from demonstrations across different team sizes.
- Chapter 5 describes TACTIC, a pretraining framework that enables coordination across sight range shifts.
- Chapter 6 presents COP, a method to generalize across initial states through a structured shared state representation.
- Chapter 7 concludes with a synthesis of findings and a discussion of future directions.

This dissertation aims to contribute to the vision of robot systems that are easier to train, more flexible to deploy, and more resilient to change.

Chapter 2: Efficient Policy Training with Human-Drawn Trajectory Sketches

2.1 Introduction

Robots are increasingly being deployed in dynamic environments, where they must perform a wide range of tasks with precision and adaptability. One of the key challenges in enabling robots to learn new skills lies in specifying complex, task-specific behaviors. Learning from Demonstration (LfD) ([26]) has become a widely used approach, allowing robots to acquire novel motions by imitating expert-provided trajectories. However, collecting demonstration data for LfD is challenging, particularly for high degree-of-freedom (DOF) robots performing manipulation.

While methods like kinesthetic teaching and teleoperation [27, 28, 29] are well-established and effective approaches for collecting demonstrations, the robotics community continues to explore complementary methods to expand the toolkit available for robot learning. Several new approaches have emerged, such as using manually-operated grippers instrumented with smartphone apps [30] and Virtual Reality (VR) based teleoperation systems [31], offering more intuitive hardware interfaces for collecting demonstrations. There has also been growing interest in leveraging an innate human ability to communicate spatial ideas and motions through simple sketches. For example, a quick sketch of a path can easily communicate the intended movement for navigating toward a goal location.

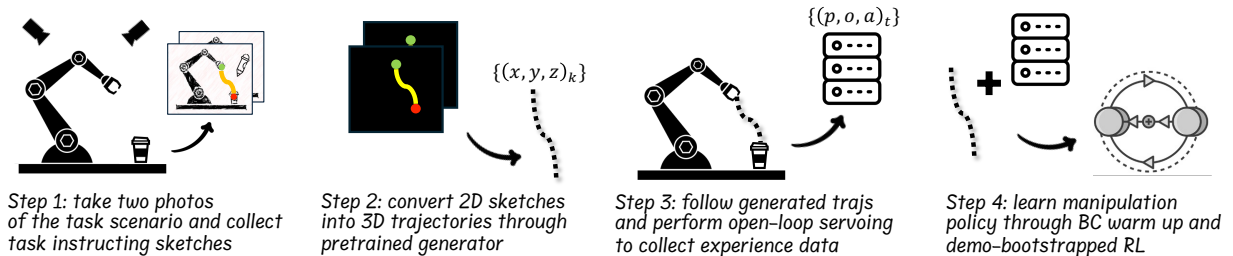


Figure 2.1: Learning a new skill in the SKETCH-TO-SKILL framework. Step 1: Capture the task scenario from two views and collect human-drawn sketches. Step 2: Convert 2D sketches to 3D trajectories using a pretrained generator. Step 3: Execute generated trajectories to collect experience data. Step 4: Learn manipulation policy using reinforcement learning bootstrapping from behavior cloning and using guidance for experience data.

Researchers have begun to explore this promising direction. RT-Trajectory [32] introduced the notion of sketches and showed how to use coarse trajectory sketches for policy conditioning in Imitation Learning (IL). RT-Sketch [33] extended this concept to leverage hand-drawn sketches of the entire environment for goal-conditioned IL. These methods demonstrated the potential of utilizing sketches in robotics, but they were primarily focused on IL and biased towards tasks they were specifically trained on. Zhi et al. [34] expanded this idea with *diagrammatic teaching*, where users instruct robots by sketching motion trajectories directly on 2D images of the scene. Their approach uses density estimation and ray tracing to reconstruct 3D trajectories from the sketches, thus limiting its ability to replicate only the provided sketches and restricting its generalization to new or unseen task setups.

Unlike prior work that used sketches only as conditioning in IL, we present a more generalizable approach that learns to predict 3D trajectories from sketches in Reinforcement Learning (RL). Specifically, we propose SKETCH-TO-SKILL (Figure 2.1), a framework that bootstraps and guides RL using sketches. Our approach first learns to map 2D sketches to 3D trajectories, which are then used to collect demonstrations. We utilize these sketch-generated demonstrations in two ways: first, by pre-training an initial policy through Behavior Cloning (BC), and second,

by refining this policy through reinforcement learning with guided exploration. Although sketch-generated demonstrations are not as precise or high-quality as teleoperated ones, they still contain enough useful information to aid RL and reduce learning time.

This approach capitalizes on the accessibility of sketching, allowing contributions from non-experts without requiring specialized hardware. By treating these sketch-based trajectories as approximate guiding signals, we demonstrate that even simple 2D sketches can enable policies that achieve performance comparable to those trained with teleoperated demonstrations. We summarize our contributions as follows:

- (1) We identify and address a crucial gap by integrating sketches into RL, extending their application beyond imitation learning and policy conditioning.
- (2) We propose SKETCH-TO-SKILL, a framework that leverages sketches to bootstrap and guide RL, reducing reliance on high-quality, real-world demonstrations.
- (3) Through extensive experiments, we demonstrate that sketches, despite their low fidelity, significantly accelerate learning by improving exploration and task comprehension in RL. SKETCH-TO-SKILL achieves $\sim 96\%$ of the performance of the baseline model utilizing high-quality teleoperation demonstrations, while exceeding the performance of a pure reinforcement learning policy by $\sim 170\%$ during evaluation.

2.2 Related Work

Learning from Demonstration (LfD). LfD [26] is a key method in robot learning, allowing robots to acquire skills through expert demonstrations, bypassing the complexities of action programming and cost function design [35]. Kinesthetic teaching, where an expert phys-

ically guides the robot while its movements are recorded, is widely used in methods like DMPs [36, 37], Probabilistic Movement Primitives [38], and stable dynamical systems [39, 40, 41]. However, it is labor-intensive and challenging to scale. Teleoperation [42], where users control robots remotely, offers more flexibility but can be complex and requires expertise to operate. VR interfaces [43, 31] provide a more immersive alternative but depend on specialized hardware. To overcome these limitations, recent research has introduced more accessible approaches, like sketch-based demonstrations [44].

Sketches in Robotics. Sketches have become a powerful tool in computer vision, aiding tasks like scene understanding [45] and object detection [46, 47]. RT-Sketch [33] first explored hand-drawn sketches for goal-conditioned imitation learning (IL), using them to define tasks intuitively. RT-Trajectory [32] extended this by using trajectory sketches as IL policy conditioning, either drawn by users or generated by a Large Language Model from task descriptions. Similarly, the Diagrammatic Teaching framework [34] uses density estimation and ray tracing to reconstruct 3D trajectories from the sketches. These methods, however, only use sketches as conditioning for task completion, and thus do not generalize beyond the tasks where the sketches are provided.

Demonstration-Enhanced Strategies for Efficient RL. Incorporating demonstration data in RL can improve sample efficiency, especially in environments where rewards are sparse. Methods such as Reinforcement Learning from Prior Data (RLPD) [48], Imitation Bootstrapped RL (IBRL) [49] and NAVINACT [50] take advantage of prior demonstrations by embedding them into the agent’s replay buffer. During training, these examples are oversampled, offering the agent more frequent exposure to expert-guided trajectories. Such approaches significantly improve learning speed and performance, particularly in continuous control tasks where learning from scratch can be prohibitively slow and inefficient [23]. Our research expands upon these

techniques by exploring how sketch-based trajectories can be used as an additional source of prior data in RL.

2.3 SKETCH-TO-SKILL: Bootstrapping Robot Learning with Human Drawn Trajectory Sketches

Our approach bootstraps robot learning from trajectory sketches, significantly lowering the barrier to entry for robotic task specification. This section details our three-stage method: (1) training a Sketch-to-3D Trajectory Generator, (2) obtaining 3D trajectories and execution experiences through the Generator and open-loop servoing, (3) pre-training an initial robotic manipulation policy through behavior cloning, and refining the policy through reinforcement learning with guided exploration. By integrating intuitive human input with powerful learning algorithms, our approach aims to create more accessible and adaptable robotic learning systems.

2.3.1 Sketch-to-3D Trajectory Generator

Our method begins with a Sketch-to-3D Trajectory Generator, T , that translates a pair of 2D sketch images (I^1, I^2) obtained from different viewpoints into corresponding 3D robot trajectory ξ_g . To train this generator, we use a dataset consisting of 3D robot end-effector trajectories along with their 2D sketches from two viewpoints. These trajectories can be obtained from various sources, such as play data where the robot executes sequences of actions. Sketches during inference can be provided by a human on RGB images of the scene, as shown in Figure 2.4. However, the sketches fed as input to the generator are 2D projections on blank backgrounds, with green and red dots representing the start and end points respectively, and yellow lines for the

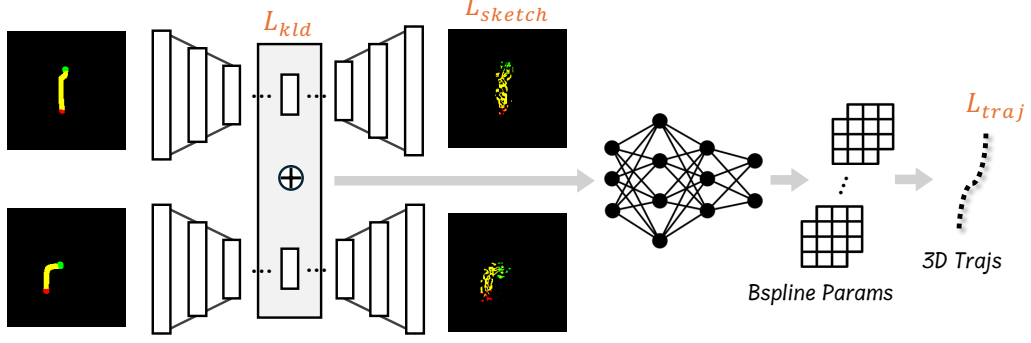


Figure 2.2: The Sketch-to-3D Trajectory Generator takes dual-view 2D sketches as inputs and predicts B-spline parameters to generate the final 3D trajectory output.

trajectory (see Figure 2.2 for an example). By focusing solely on the trajectory information without additional scene complexity, our model can efficiently learn to encode the dual-view sketches and decode them into the corresponding 3D trajectory.

The generator uses a neural network to map dual-view 2D sketches to 3D trajectories, where we adopt a hybrid architecture combining a Variational Autoencoder (VAE) [51] and a Multilayer Perceptron (MLP), as illustrated in Figure 2.2. The VAE encodes sketches from two viewpoints, ideally orthogonal, to resolve depth ambiguity and capture essential trajectory features. The MLP decoder generates B-spline [52] control points $\mathbf{C} \in \mathbb{R}^{n_{cp} \times 3}$ from the latent representation, which we then use to interpolate smooth 3D trajectories. We adopted uniform knots and pre-compute the B-spline parametrization matrix $\mathbf{W} \in \mathbb{R}^{n_{tp} \times n_{cp}}$ to reduce computational complexity and facilitate efficient backpropagation. The calculation of $\mathbf{W}$ only depends on the uniform knot vector $\mathbf{u}$ and the desired number of points n_{tp} in the generated trajectory, and can be pre-calculated using the Cox-de Boor recursion formula (also known as de Boor’s algorithm [53], see Appendix 2.6.1 for details). Then the final trajectory generation is simply a matrix multiplication: $\xi_g \in \mathbb{R}^{n_{tp} \times 3} = \mathbf{W} \cdot \mathbf{C}$. With the generated control point parameter $\mathbf{C}$, we can also easily generate trajectories of varying density from the same parameters, making our

method adaptable to different task requirements.

Our training process uses a multi-component loss function $L = L_{traj} + L_{sketch} + L_{kld}$, where L_{traj} handles trajectory reconstruction, L_{sketch} manages sketch reconstruction (Mean Square Error), and L_{kld} is KL-divergence for latent space regularization (Figure 2.2). This ensures accurate trajectory generation while preserving sketch fidelity and latent space structure. We also applied data augmentation to both the sketch images and the trajectories to enhance the model’s robustness and generalization (more details can be found in Appendix 2.6.2).

We can use the trained Sketch-to-3D Trajectory Generator, T , to generate demonstrations $\{\xi_D\}$ for learning new tasks using sketches drawn by a human. Specifically, the human draws trajectory sketches on two views of RGB images captured from the initial task state. This is similar to how human-drawn sketches are generated in prior works [32, 34]. These paired sketch images, $\{(I^1, I^2)\}$, are input into our trained generator, which produces corresponding 3D trajectories, $\{\xi_g\}$, serving as the basis for guiding the robot’s actions. We can also generate more than one trajectory from the same pair of sketches by adding controlled noise to the latent representation. Then we proceed to collect demonstrations for manipulation policy learning. We execute these trajectories on the robotic arm using open-loop servoing, which enables precise trajectory following based on pre-computed motor commands. During execution, we record a demonstration dataset $\{\xi_D = \{(p_t, o_t, a_t)\}_{t=1}^T\}$ at a fixed frequency, where $p_t = (x, y, z)_t$ denotes the robot’s end-effector 3D position, o_t represents the robot’s observation, a_t is the corresponding action, and T is the total number of timesteps per demonstration. The collected demonstrations, which do not need to be optimal, follow the intended path while capturing the robot’s actual behavior in the target environment. They serve as an effective starting point for bootstrapping the policy learning process, offering initial guidance grounded in the robot’s real-world performance.

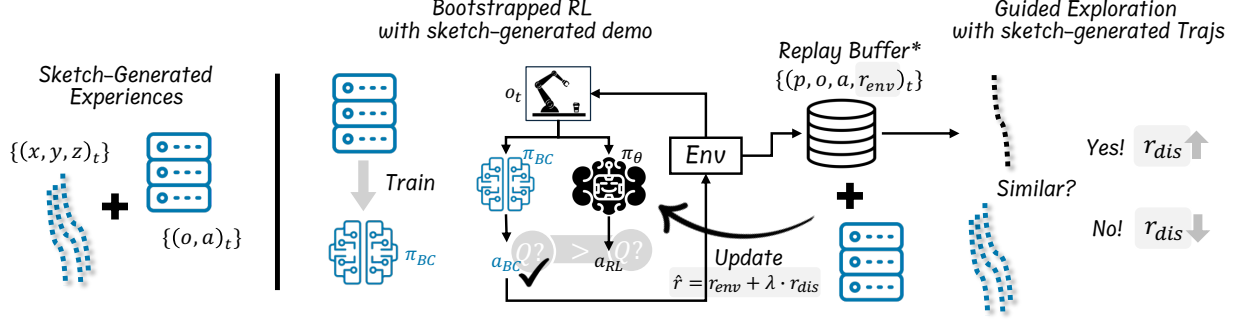


Figure 2.3: Overview of SKETCH-TO-SKILL integrating sketch-generated demonstrations with reinforcement learning. Sketch-generated experiences train an IL policy, which bootstraps the RL process. A discriminator guides exploration by rewarding similarity to sketch-generated trajectories. The final action, combining IL and RL policy outputs, further enhances the exploration guidance. The asterisk after "Replay Buffer" indicates that the buffer is initialized with the open-loop serving demonstrations.

2.3.2 Policy Learning

We now describe the policy learning of the SKETCH-TO-SKILL algorithm (given in Algorithm 1). Taking as input the demonstration data $\{\xi_D\}$ collected from our Sketch-to-3D Trajectory Generator T and through open-loop serving (lines 4–5), our approach combines IL and RL to effectively bootstrap and refine the policy. Specifically, we build upon the Imitation Bootstrapped Reinforcement Learning (IBRL) framework [49], integrating our sketch-based trajectories to guide and constraint policy search space.

In IBRL we replace traditional real-world demonstrations with sketch-generated demonstrations. Initially, these sketch-based demonstrations are used to train an IL policy (line 6), which serves as a coarse approximation of the task. Although these sketches do not capture every fine detail of manipulation (e.g., gripper closing/opening actions or exact force control), our hypothesis posits that they still carry significant, actionable information that can effectively guide the learning process in reinforcement learning (RL). We leverage this information in RL in two

Algorithm 1 SKETCH-TO-SKILL. Major modifications of IBRL highlighted in blue.

- 1: **Hyperparameters:** Number of critics E , number of critic updates G , update frequency U , exploration std σ , noise clip c , number of generated trajectories m per input sketch pair, reward weighting term λ
 - 2: **Inputs:** Pre-trained Sketch-to-3D Trajectory Generator T , sketch dataset $\mathcal{S} = \{(I_i^1, I_i^2)\}_{i=1}^n$,
 - 3: **Outputs:** Policy π_θ , discriminator D_ψ
-
- Stage 1: Demonstration Generation*
-
- 4: $\{\xi_g\}_{1:mn} \leftarrow$ generate m trajectories per sketch from $\mathcal{S}$ using T
 - 5: $\{\xi_D\}_{1:mn} \leftarrow$ generated demonstrations through open-loop servoing
-
- Stage 2: Policy Learning*
-
- 6: Train imitation policy π_{IL} on demonstrations $\{\xi_D\}_{1:mn}$ using the selected IL algorithm.
 - 7: Initialize policy π_θ , target policy $\pi_{\theta'}$, and critics Q_ϕ , target critics $Q_{\phi'}$, discriminator D_ψ for $i = 1, \dots, E$
 - 8: Initialize replay buffer B with demonstrations $\{\xi_D\}_{1:mn}$
 - 9: **for** $t = 1$ **to** N **do**
 - 10: Observe current observation o_t from the environment
 - 11: Compute IL action $a_t^{IL} \sim \pi_{IL}(o_t)$ and RL action $a_t^{RL} = \pi_\theta(o_t) + \epsilon$, where $\epsilon \sim N(0, \sigma^2)$
 - 12: Sample a set K of 2 indices from $\{1, 2, \dots, E\}$
 - 13: Select action a_t with higher Q-value from $\{a_t^{IL}, a_t^{RL}\}$
 - 14: Execute action a_t
 - 15: Store transition $(p_t, o_t, a_t, r_t, p_{t+1}, o_{t+1})$ in replay buffer B
 - 16: **if** $t \% U = 0$ **then**
 - 17: Perform discriminator D_ψ update by optimizing Equation 2.1
 - 18: Perform TD3 update using minibatches from replay buffer B [54]
 - 19: **end if**
 - 20: **end for**
-

ways (as shown in Fig. 2.3):

- (1) Bootstrap RL with Sketch-Generated Demos: Even though sketch-generated trajectories are not as detailed as teleoperated demonstrations, they provide a foundational blueprint of the task. We leverage these initial trajectories to bootstrap our RL algorithm, giving it a preliminary direction and reducing the cold start problem common in RL scenarios. This use of imperfect demonstrations is intended to establish an initial policy that avoids random exploration at the outset, making subsequent training more focused and efficient.
- (2) Guide Exploration During RL: As the agent progresses in its learning, the sketch-generated

trajectories continue to serve as a guide, shaping the exploration strategy. Instead of relying on these trajectories as definitive guides, we treat them as rough outlines that suggest areas of the task space worth exploring. This guided exploration helps concentrate the agent’s learning efforts on potentially fruitful regions of the action space, thus optimizing the learning speed and improving the relevance of the experiences gathered.

In both steps, the use of sketch-generated trajectories acknowledges their limitations—they are not treated as ground truth but as valuable signals to help bootstrap RL and guide exploration throughout the learning process. For the RL algorithm, we employ TD3 [54], an off-policy algorithm known for its sample efficiency. In our approach, the replay buffer is initialized with the sketch-generated demonstration trajectories (line 8), which provide an initial foundation for learning and is later updated with online experiences as the agent interacts with the environment. This combination allows the agent to refine its policy through both sketch-generated demonstration data ξ_D and real-world interaction (line 13).

To further enhance the learning process and maintain consistency with the sketch-generated trajectories, we introduce a discriminator-based guided exploration mechanism [55]. This discriminator, D_ψ , is trained to distinguish between trajectories produced by our Sketch-to-3D Trajectory Generator and those generated by the current policy:

$$\mathcal{L}_{D(\psi)} = \mathbb{E}_{p,g \sim \{\xi_D\}} [\log D_\psi(p, \Delta p, g)] + \mathbb{E}_{p,g \sim \pi_\theta} [\log(1 - D_\psi(p, \Delta p, g))], \quad (2.1)$$

where p represents the end-effector location, Δp is the normalized difference between the current and next end-effector positions, capturing local trajectory characteristics, and g is the task-specific information (e.g., target location). This formulation allows the discriminator to assess trajectory

similarity while accounting for task variability. We then augment the TD3 reward function with an additional term based on the discriminator’s output (line 18):

$$\hat{r}(o_t, a_t) = r(o_t, a_t) + \lambda \log D_\psi(p_t, \Delta p_t, g), \quad (2.2)$$

where λ is a hyperparameter controlling the influence of the discriminator. This augmented reward encourages the policy to explore state-action spaces more likely to produce trajectories similar to those generated from human sketches, potentially leading to faster learning and better performance.

Our overall learning process iterates between TD3 optimization and discriminator training. In each iteration: (1) We update the discriminator using the latest policy-generated trajectories and the original sketch-generated trajectories (line 17). (2) We then update the policy and Q-functions using TD3, with the augmented reward and guidance from the frozen IL policy (line 18). This iterative process allows the policy to refine its behavior while maintaining similarity to the initial demonstrations derived from human sketches. By combining IL, TD3, and discriminator-based guided exploration, we create a cohesive learning framework that effectively leverages sketch-based demonstrations to accelerate and improve the learning of complex manipulation tasks. Please see the Appendix for more implementation details and a complete list of hyper-parameters.

2.4 Experiments

We report our evaluation of SKETCH-TO-SKILL, focusing on its main components: the Sketch-to-3D Trajectory Generator, the Imitation-Bootstrapped RL Policy learning, and the use

of the discriminator. Our experiments address the following key questions:

- Q1 How effectively does the Sketch-to-3D Trajectory Generator convert 2D sketches into usable 3D robot trajectories?
- Q2 Can SKETCH-TO-SKILL utilize sketch-generated demonstrations to achieve comparable performance to traditional methods using high-quality demonstration data?
- Q3 How do various design choices in SKETCH-TO-SKILL, such as the number of generated demonstrations per sketch and the discriminator reward weighting, affect the learning and refinement of robotic policy?
- Q4 How well does our method translate to the real world?

2.4.1 Evaluation of the Sketch-to-3D Trajectory Generator

The Sketch-to-3D Trajectory Generator is a key component of SKETCH-TO-SKILL, translating 2D sketch inputs into 3D robot trajectories. To train this generator, we collect data of the robot arm executing *play* trajectories. We record the 3D trajectories as well as their 2D projections from two viewpoints. We create such a dataset in the Metaworld [56] simulation environment as well as a separate one using actual hardware (Figure 2.11). Once the Sketch-to-3D Trajectory Generator is trained, we can use hand-drawn sketches as input to predict 3D trajectories.

Performance on Hand-drawn Sketches. We provide an example using the `ButtonPress` task to qualitatively assess the generator’s effectiveness with hand-drawn inputs (Figure 2.4). We asked users to provide sketches for the task and also separately collected actual demonstrations as a reference. We see that the Sketch-to-3D Trajectory Generator was able to predict trajectories (Figure 2.4a) similar to the actual demonstrations (Figure 2.4c). We also generate more than one

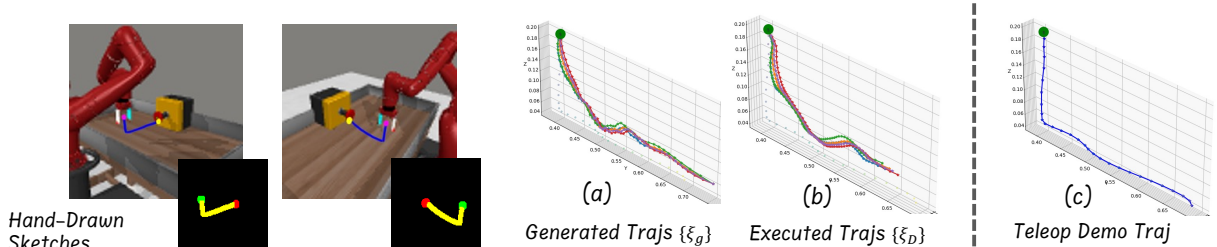


Figure 2.4: Multi-stage trajectory generation and execution. On the left, we show hand-drawn sketches on scenario RGB images and the extracted sketches on a blank background, (a) generated trajectory from the Sketch-to-3D Trajectory Generator, and (b) executed trajectory via open-loop serving. In (c), we visualize a teleoperated demo for the same task for reference.

trajectory from the same pair of sketches by adding controlled noise to the latent representation. This approach allows us to produce a range of plausible trajectories for a given sketch input, enriching the demonstration set and potentially leading to more robust and adaptable robot policies. We then execute the generated trajectories to produce demonstrations for training the policy (Figure 2.4b). Despite the inherent variability in sketch inputs, the executed trajectory further validates the practical applicability of our approach. This demonstrates our model’s robustness to sketch imperfections and its ability to reliably interpret user intent, bridging the gap between simple 2D sketches and actionable 3D robot trajectories.

Latent Space Representation and Interpolation. To further understand the generator’s latent space, we performed linear interpolation in the latent space between different input samples. Specifically, we selected two distinct sketch pairs with different trajectories, extracted their feature vectors, linearly interpolated between them, reconstructed the sketches, and generated new trajectories. Figure 2.14 shows smooth transitions in both 2D sketches and 3D trajectories across the interpolated latent space. This smoothness demonstrates that our model has learned a continuous and semantically meaningful representation, suggesting good generalization capability to unseen inputs that lie between known examples [51]. The coherence between interpolated

sketches and their corresponding 3D trajectories further validates the model’s robust sketch-to-trajectory mapping.

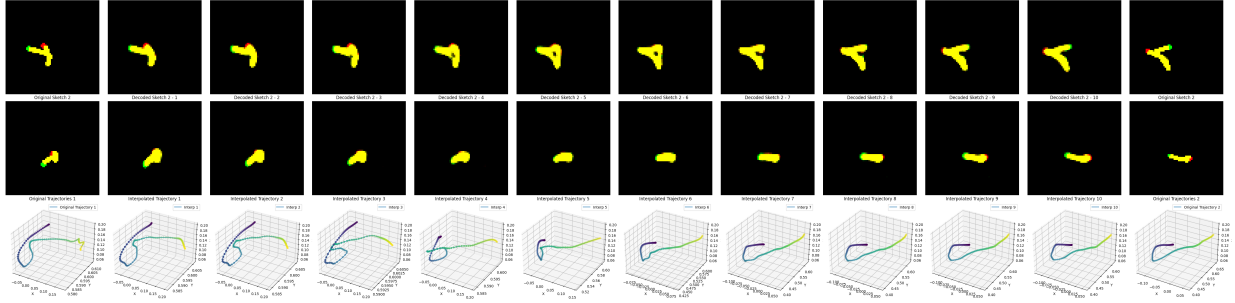


Figure 2.5: Latent space interpolation results showing smooth transitions between original samples (leftmost and rightmost) and reconstructed samples (middle) from two viewpoints, demonstrating the model’s ability to generate coherent 3D trajectories from 2D sketch pairs.

Effect of VAE in Sketch-to-3D Trajectory Generator. We also conducted an ablation study of the Sketch-to-3D Trajectory Generator to evaluate the effect of the VAE on the architecture. Using a dataset of 1000 trajectories split 80:20 for training and validation, we report the training and validation losses in Table 2.1. We compare the performance of the model with the VAE (shown in Figure 2.2) to a variant without the VAE, where only a CNN is used. This CNN has the same architecture as the VAE’s encoder, but without the decoder and loss components. Our results show that incorporating the VAE consistently improves performance across all metrics, including reconstruction loss and trajectory loss.

Table 2.1: Performance Metrics for generator model

Metric	with VAE	without VAE
Training Loss	0.6019	0.6286
Validation Loss	0.9128	1.2804
Reconstruction Loss	0.0004	0.3258
KLD Loss	69.8592	107.167
Parameter MSE Loss	0.0820	0.0928
Trajectory Loss	0.770	0.1180

2.4.2 Comparisons with Baselines

In this section, we conduct extensive experiments in Robomimic [57] and MetaWorld [56] to answer Q2: *can SKETCH-TO-SKILL utilize sketch-generated demonstrations to achieve comparable performance to traditional methods using high-quality demonstration data?* Specifically, we compare SKETCH-TO-SKILL with: (1) IBRL [49], a strong baseline that utilizes traditional high-quality demonstration data (rather than sketches as what our method uses), and (2) TD3 [54], a state-of-the-art pure RL approach without using any demonstrations. We hypothesize that although the sketches have only partial information (namely, 2D projections of 3D trajectories and no gripper information), we can still generate good enough demonstration data to perform comparably with the baseline that uses full demonstrations. We show that to be the case in these experiments.

We perform evaluations on two-stage `PickPlaceCan` task from Robomimic and six tasks from the MetaWorld benchmark, namely `Coffeepush`, `Boxclose`, `Buttonpress`, `Reach`, `Reachwall`, and `ButtonpressTopdownwall`, each using sparse 0/1 task completion rewards at the end of each episode. For each task, we collected 10 high-quality demonstrations using an expert policy in Robomimic and 3 high-quality demonstrations in MetaWorld. These demonstrations served as our baseline for traditional demonstration-based methods. For our approach, we collected a total of three hand-drawn sketches, one on each demonstration’s initial frames (Figure 2.4). These sketches were used to generate and execute trajectories, creating a parallel set of sketch-based demonstrations for comparison.

Figure 2.6 shows the setup where the sketch-based demonstrations provide initial positional cues for the robot in the Robomimic environment. We ask users to draw separate sketches for

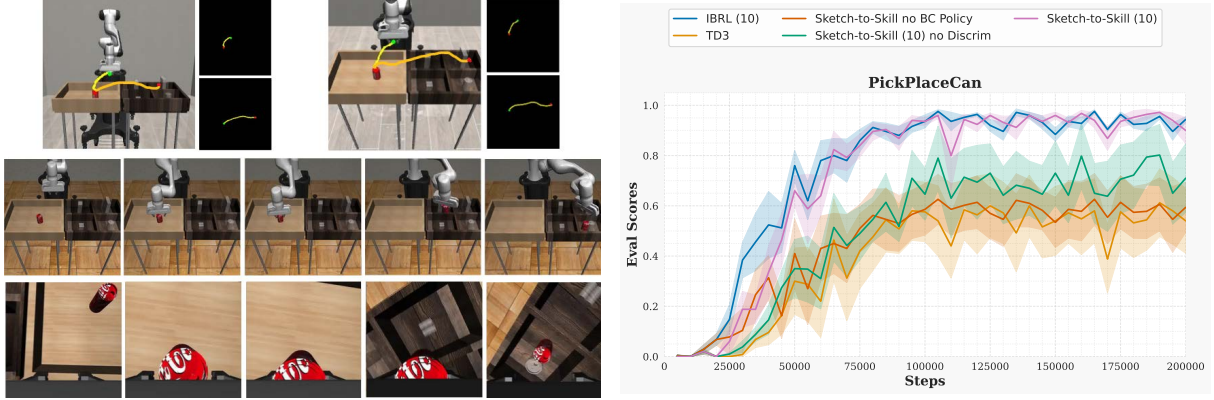


Figure 2.6: Evaluation Scores (success rate) for the robomimic PickPlaceCan environment during evaluation.

each stage, and then combine the trajectories during open-loop servoing. Our framework utilizes RL to refine these initial cues, dynamically adjusting the robot’s approach to manage both the orientation and timing necessary for successful task execution. This method proves particularly effective as it does not rely on fully detailed trajectory information from the start; instead, it uses sketches to guide the initial exploration phase of RL, simplifying the data collection process. Our sketch-to-skill framework, even with limited initial data, performs admirably in this demanding scenario. The results are on par with those from IBRL methods $\sim 98\%$, which benefit from complete and detailed human demonstrations, including explicit orientation details (as shown in Figure 5).

Figures 2.7 show the evaluation performance across all the tasks in the MetaWorld environment. To assess the effectiveness of individual components, we also compare SKETCH-TO-SKILL’s results with variants where the BC policy guidance or the discriminator reward is removed. We observe that SKETCH-TO-SKILL consistently outperforms pure RL across all tasks, often by a notable margin. Both the BC policy guidance and the discriminator reward contribute to policy learning, resulting in the most stable performance across tasks. Remarkably, SKETCH-

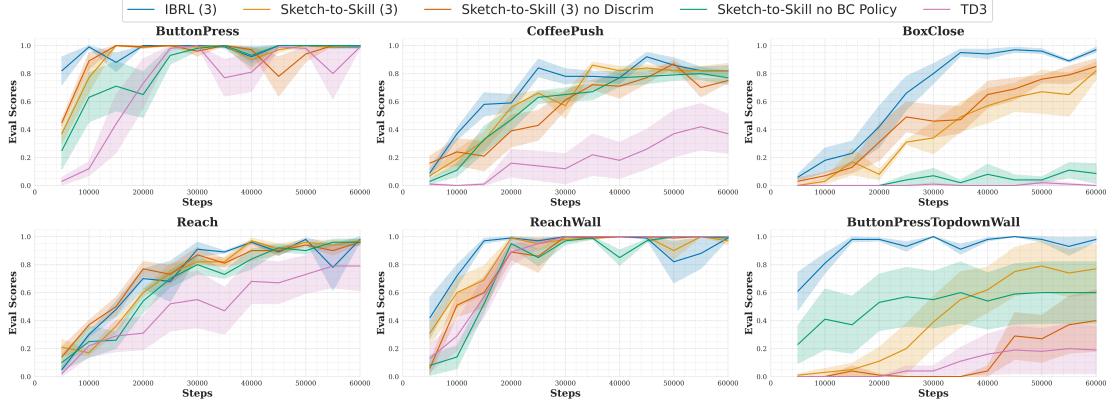


Figure 2.7: **Evaluation Performance of SKETCH-TO-SKILL in MetaWorld** This figure shows the success rate across six MetaWorld tasks under randomized initial gripper and object positions. SKETCH-TO-SKILL achieves comparable performance to IBRL while surpassing pure RL.

TO-SKILL achieves performance comparable to IBRL, which relies on high-quality demonstrations, even when using only sketches as input. This is particularly evident in the `CoffeePush` and `BoxClose` tasks. These tasks require actuating the gripper — information that is not provided in the sketches. Nevertheless, SKETCH-TO-SKILL is able to bootstrap and use guidance from the sketch generated suboptimal demonstrations to learn a policy efficiently. This provides evidence to the claim that the sketch-generated demonstrations do not lead to much degradation in performance while being much easier to obtain.

Behavioral Cloning Performance. SKETCH-TO-SKILL employs behavior cloning (BC) to bootstrap policy learning, similar to IBRL. However, the key difference is that IBRL relies on high-quality teleoperated demonstrations, whereas SKETCH-TO-SKILL uses sketch-generated demonstrations. We compare the performance between them (Figure 2.8) and ablate the number of generated demonstrations m per input sketch pair. Not surprisingly the BC policy with teleoperated data performs better than the sketch generated ones. However, despite the lower performance of the BC policy, SKETCH-TO-SKILL is still able to achieve comparable performance in RL training (as seen in Figures 2.6 and 2.7), showing that it is not as sensitive to the quality of

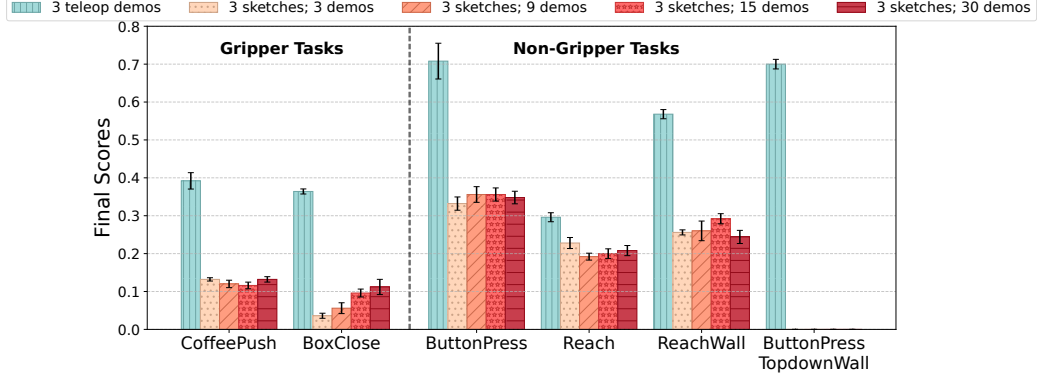


Figure 2.8: **Behavioral Cloning (BC) scores using actual teleoperated data and sketch generated demonstrations.** The blue bars represent the baseline BC policy trained with 3 high-quality demonstrations, while the red bars show BC policies trained with sketch-generated demonstrations, varying in the number of demonstrations m per input sketch pair (1, 3, 5, and 10). Darker shades of red indicate an increase in the number of sketch-based demonstrations used for training. Despite poor success rate, the actual trajectories and policy learned with sketches are useful for bootstrapping as evidenced by the training performance (Figures 2.7).

the bootstrapping policy. Increasing the number of generated demonstrations m per input sketch pair (from 1 to 10) does not significantly improve the BC performance.

2.4.3 Ablation Studies

To understand the impact of the key components in SKETCH-TO-SKILL and answer Q3, we conducted ablation studies focusing on two critical aspects: the number of generated trajectories m per input sketch pair and the reward weighting scheme λ .

Impact of Generated Trajectories per Sketch. We investigated how the number of trajectories generated from each input sketch pair affects the learning performance. Figure 2.9, shows the learning curves for policies trained with varying numbers of generated trajectories per sketch. We observe improved performance when generating $m = 3$ trajectories per sketch instead of just one. The additional demonstrations help compensate for the lack of actual teleoperated demonstrations. However, the benefit of increasing the number of trajectories per sketch becomes less

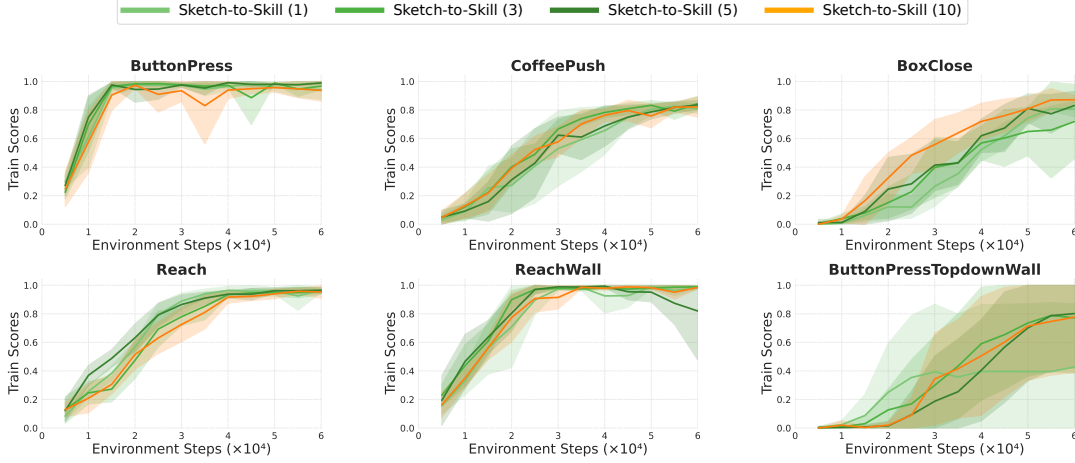


Figure 2.9: This figure illustrates ablation training scores for SKETCH-TO-SKILL with varying m , the number of demonstrations generated per sketch pair (1, 3, 5, and 10).

pronounced beyond a certain point. This approach is particularly useful for challenging tasks, such as `BoxClose` and `CoffeePush`, which involve precise gripper actions, but has limited impact on simpler tasks.

Effect of Reward Weighting: We examined the impact of different reward weighting schemes on policy learning. Our reward function combines the environmental reward with a discriminator-based reward by Equation 2.2, where λ is the weighting parameter.

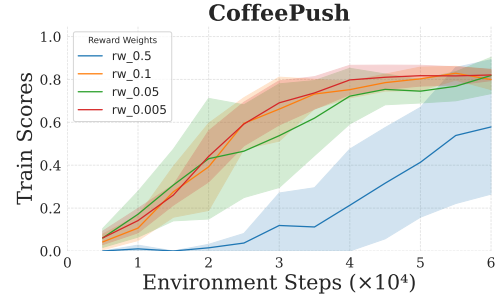


Figure 2.10: Reward weighting term ablation

Figure 2.10 shows the learning performance across various λ values: 0.005, 0.05, 0.1, and 0.5. The model performs comparably with λ values of 0.005, 0.05, and 0.1, indicating relative insensitivity to this hyperparameter. However, it significantly underperforms when λ is set to 0.5.

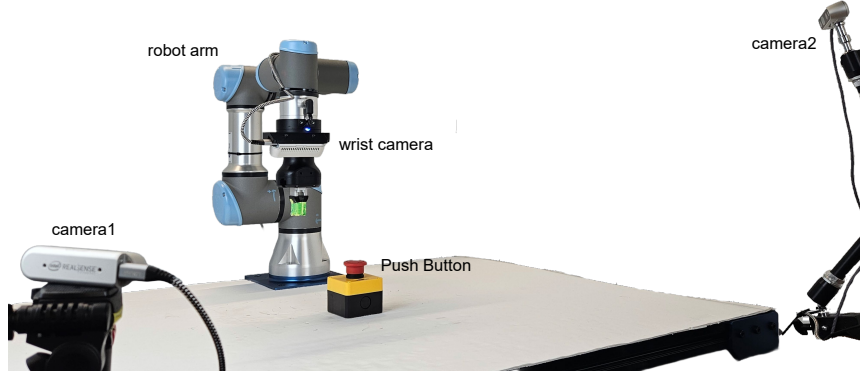


Figure 2.11: Complete setup for the ButtonPress task in a real-world experiment. The configuration includes a UR3e robot arm equipped with a Robot Hand gripper, and a RealSense D435i camera mounted on the wrist. Two additional RealSense cameras are positioned orthogonally to capture the trajectory from two different viewpoints.

2.4.4 Hardware Experiments

We validate SKETCH-TO-SKILL on physical robot hardware to demonstrate its effective transfer from simulation to real-world applications.

Experimental Setup. We set up 3 real-world tasks namely *Buttonpress*, *ToastPress* and *ToastPickPlace* as shown in Figure 2.12. We use a UR3e robot equipped with a Robotiq hand-e gripper and a realsense camera mounted on the wrist. We also use two additional environmental cameras to capture frames for humans to draw sketches on (Figure 2.11). The details of the task, success detection, and reset mechanism are in the Appendix.

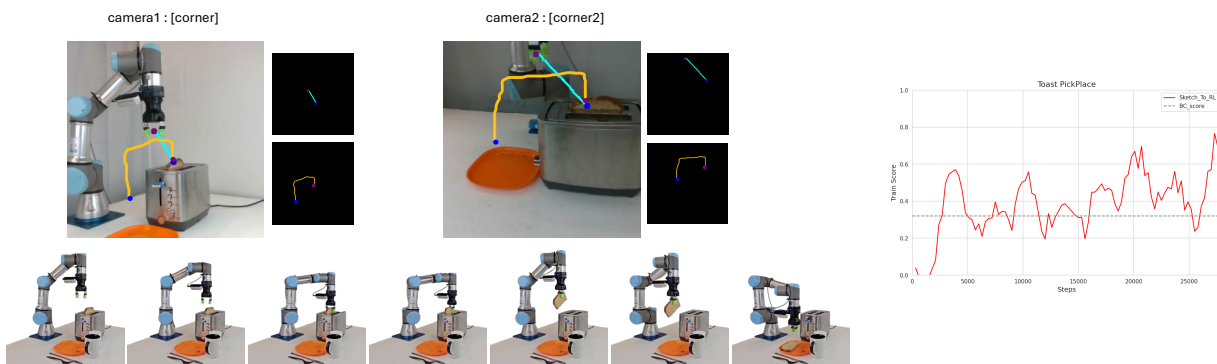
Performance. The evaluation of our approach across multiple tasks highlights the effectiveness of sketch-generated demonstrations in training policies. For the **ButtonPress** task, the BC policy achieved a notable success rate of 80% in a randomized environment, demonstrating the robustness of the approach. Similarly, the Sketch-to-Skill policy, without a discriminator, reached a high training success rate of 80% within just 30,000 samples. In the **ToastPress** task, the BC policy achieved a preliminary success rate of 60% using sketch-generated demonstra-



(a) Training curves for the real-world ButtonPress task.



(b) Training curves for the real-world ToastButtonPress task.



(c) Training curves for the real-world ToastPickPlace task.

Figure 2.12: Training curves and hardware setups for real-world robotic tasks: ButtonPress, ToastButtonPress, and ToastPickPlace.

tions, while the Sketch-to-Skill policy significantly improved performance, achieving approximately 90% success within 10,000 interactions. Lastly, in the **ToastPickPlace** task, the BC policy showed an initial success rate of 36% under dynamic conditions. However, the Sketch-to-Skill policy demonstrated considerable improvement, achieving approximately 80% success within 30k interactions.

2.5 Conclusions and Future Work

We present SKETCH-TO-SKILL that uses 2D sketches to improve the efficiency of learning a manipulation skill. While prior work has shown the power of sketches in IL, we are the first to show how to do so using RL. The key ideas are to train a 2D sketch to 3D trajectory generator whose output is used to bootstrap learning of the RL policy and used as an extra exploration guidance signal, all of which contributes to improved efficiency. There are several avenues for future work. SKETCH-TO-SKILL currently does not include any gripper information or timing information in the sketches. While we have shown that even without this information, we can learn effectively, an immediate line of work would be to include this in the sketches. Fundamentally, this does not change the Sketch-To-3D trajectory generator model which would now also have to predict the gripper state and time parameterization of the trajectory. The second avenue for future work is to generate the sketches from sources other than humans. For example, (author?) [32] showed how to generate sketches using a Vision Language Model given a natural language description of the task. We can directly incorporate such sketches into our framework.

2.6 Appendix

2.6.1 De Boor's Algorithm Details

The uniform knot vector for a B-spline of degree p with $n + 1$ control points is defined as:

$$\mathbf{u} = [\underbrace{0, \dots, 0}_{p+1}, \frac{1}{n-p+1}, \frac{2}{n-p+1}, \dots, \frac{n-p}{n-p+1}, \underbrace{1, \dots, 1}_{p+1}] \quad (2.3)$$

The B-spline basis functions $\mathbf{W}_{i,p}(t)$ are defined recursively using the Cox-de Boor recursion formula:

$$\mathbf{W}_{i,0}(t) = \begin{cases} 1 & \text{if } u_i \leq t < u_{i+1} \\ 0 & \text{otherwise} \end{cases} \quad (2.4)$$

$$\mathbf{W}_{i,p}(t) = \frac{t - u_i}{u_{i+p} - u_i} \mathbf{W}_{i,p-1}(t) + \frac{u_{i+p+1} - t}{u_{i+p+1} - u_{i+1}} \mathbf{W}_{i+1,p-1}(t) \quad (2.5)$$

where u_i are the knot values from the knot vector $\mathbf{u}$.

The B-spline parametrization matrix N for m evaluation points is an $m \times (n + 1)$ matrix:

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_{0,p}(t_1) & \mathbf{W}_{1,p}(t_1) & \cdots & \mathbf{W}_{n,p}(t_1) \\ \mathbf{W}_{0,p}(t_2) & \mathbf{W}_{1,p}(t_2) & \cdots & \mathbf{W}_{n,p}(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{W}_{0,p}(t_m) & \mathbf{W}_{1,p}(t_m) & \cdots & \mathbf{W}_{n,p}(t_m) \end{bmatrix} \quad (2.6)$$

where t_j ($j = 1, \dots, m$) are evenly spaced parameters in the interval $[0, 1]$.

2.6.2 Implementation Details and Hyperparameters of Sketch-to-3D Trajectory Generator

Overview of the Model. The proposed model converts 2D image sketches into 3D motion trajectories using a *Variational Autoencoder (VAE)* combined with a *Multi-Layer Perceptron (MLP)*. The VAE encoder processes 64×64 pixel 2D sketches (3 channels) into a latent vector ($d_v = 32$), while the decoder reconstructs the sketches to retain essential features for trajectory generation. The latent space outputs the mean (μ) and variance (σ^2), sampled using the reparameterization trick. The MLP takes the latent vectors from two sketches, concatenates them, and generates 3D control points for B-spline trajectory interpolation. The MLP takes an input of size ($d_v \times 2$), processes it through hidden layers [1024, 512, 256], and outputs $n_{cp} \times 3$. The generated 3D control points are then used for B-spline interpolation to produce smooth trajectories.

Initialization, Regularization, and Hyperparameters. We initialize all network parameters using *Xavier initialization*. Regularization is done with *Kullback-Leibler Divergence (KLD)*, using a loss function that combines *Sketch Reconstruction Loss*, *KLD Loss*, and *Trajectory Loss*. The *Sketch Reconstruction Loss* is the MSE loss on sketch images, the *Trajectory Loss* computes the MSE between predicted and ground truth trajectories, where the ground truth trajectory is first densely fitted and resampled to ensure uniform point spacing, and the *KLD Loss* applies Kullback-Leibler Divergence for regularization. Key hyperparameters include:

- **Image size:** 64×64 pixels
- **Latent dimension:** 32
- **Number of control points:** 20

- **B-spline degree:** 3
- **MLP hidden layers:** [1024, 512, 256]
- **Learning rate:** 1×10^{-3} (Adam optimizer)
- **Batch size:** 128
- **KLD weight:** 0.0001 (with optional annealing)
- **Training epochs:** 200

To enhance robustness and generalization, our training process employs two concurrent data augmentation strategies. The first applies diverse image augmentations (rotations, scaling, affine transformations, noise) to input sketches, used exclusively for updating the VAE to learn robust sketch representations. The second strategy targets potential mismatches in hand-drawn sketches by subtly modifying both original sketches and their 3D

trajectories. This involves adding noise and minor elastic deformations to sketches, and noise with refitting to trajectories. These augmented pairs update the entire model, preparing it for hand-drawn input variability while maintaining sketch-trajectory consistency. This augmentation approach enhances the model’s ability to handle diverse, imperfect sketches while ensuring accurate 3D trajectory generation in real-world scenarios. To train the sketch generation network we collect 22000 samples for simulation tasks and 85 for the real world. During the training for

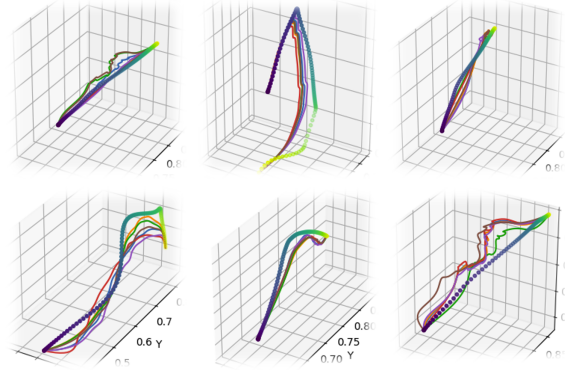


Figure 2.13: Diversity in generated 3D trajectories. Each subplot shows multiple generated trajectories (colored lines) for a single input, demonstrating variability. Scattered points represent the ground truth trajectory.

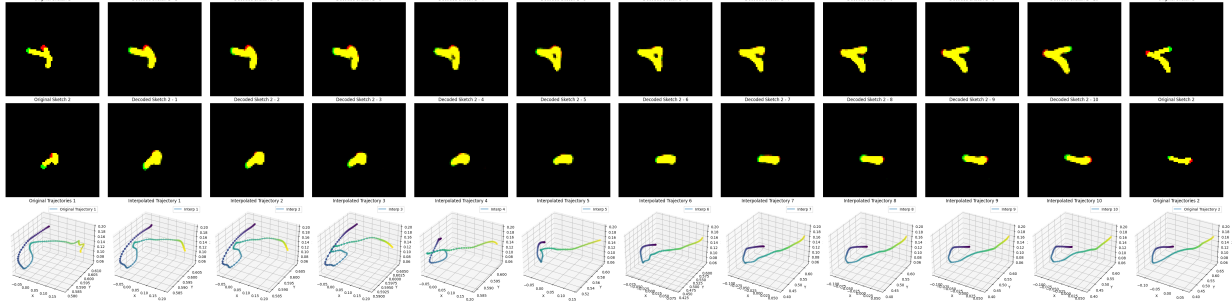


Figure 2.14: Latent space interpolation results showing smooth transitions between original samples (leftmost and rightmost) and reconstructed samples (middle) from two viewpoints, demonstrating the model’s ability to generate coherent 3D trajectories from 2D sketch pairs.

the real-world task, we use the data collected from simulation to train the VAE part as well. We provide additional visualizations to demonstrate the learned model’s generalizability in Figure 2.13 and 2.14.

2.6.3 Implementation Details and Hyperparameters of SKETCH-TO-SKILL Policy and baselines

This section outlines the implementation details of SKETCH-TO-SKILL and the baselines. The behavior cloning (BC) policies utilize a ResNet-18 encoder, where the output is flattened and processed by MLPs to produce final 4D actions. We replace the BatchNorm layers in ResNet with GroupNorm, matching the number of groups to the input channels. To prevent overfitting, we employ random-shift data augmentation. The hyperparameters are shown in Table 2.2.

In the Meta-World environment, we utilize a corner2-image camera setup. We use wrist cameras to enhance generalization and sample efficiency in real-world experiments, specifically using them for the ButtonPress task.

Table 2.2: Hyperparameters for RL in SKETCH-TO-SKILL.

Parameter	Meta-World	Real-World
Optimizer	Adam	Adam
Learning Rate	1e-4	1e-4
Batch Size	256	256
Discount (γ)	0.99	0.99
Exploration Std. (σ)	0.1	0.1
Noise Clip (c)	0.3	0.3
EMA Update Factor (ρ)	0.99	0.99
Update Frequency (U)	2	2
Actor Dropout	0.5	0.5
Q-Ensemble Size (E)	2	N/A
Num Critic Update (G)	1	N/A
Image Size	96×96	96×96
Use Proprio	No	N/A
Proprio Stack	N/A	N/A
State Stack	N/A	N/A
Action Repeat	N/A	N/A

2.6.4 Additional Details of Real-world Experiments

In this section, we present insights into the real-world experiments conducted using SKETCH-TO-SKILL. We utilized a UR3e robot equipped with a Robot Hand gripper, operating in an action space with 4 dimensions: 3 for end-effector position deltas under a Cartesian impedance controller and 1 for the absolute gripper position, with policies functioning at 7.5 Hz.

To train the Sketch-To-3D Trajectory generator, we collected approximately 85 teleoperated trajectories. RGB images were captured using two orthogonally positioned RealSense cameras to enhance trajectory insight. A green marker was placed on the gripper tip to facilitate sketch generation on the frames, enabling the model to learn 2D sketch projections onto 3D trajectories. All methods maintained the same hyperparameters and network architectures as those used in the Meta-World tasks.

2.6.4.1 Common Implementation Details

- **Reset Protocol:** For all tasks, we manually reset the environment between episodes by returning objects to their initial states and randomizing their positions. The robot is initially set to a specific joint configuration known as the home position. Whenever the agent receives a reward or an episode ends, it resets back to the home configuration and the training continues. The object randomization typically begins with placement at the center, remaining unchanged until the agent receives its first reward. After that, the object is gradually moved towards the boundary, circled around the workspace, returned to the center, and the process is repeated until the training ends.
- **Safety Boundaries:** We have restricted the movement of the robot to the x, y, and z directions of the end-effector. Each step that the robot takes is limited to a specific value. If the action taken by the agent exceeds this limit, the robot will not move and will remain in its current position. To avoid collision with the workspace surface, we have set a minimum limit for the z direction of the robot. Even if the agent attempts to move downward beyond this threshold, the robot will remain at the specified z position.
- **Reward Structure:** We used a manual method to reward the agent if it successfully completes the task. The agent receives a reward of 1 for successfully completing the task and a reward of 0 in all other cases. Each episode has a set number of timesteps, and if the agent doesn't succeed within that limit, it resets and starts over. The length of each episode may be shorter than the given limit depending on how quickly the agent completes the task.

2.6.4.2 Task-Specific Details

- **Button Press Task:** After training the Sketch-To-3D Trajectory generator, we created 30 sketches based on RGB frames from the two cameras. We then collected 30 sketch-generated demonstrations, ξ_D using openloop servoing on 3D trajectory ξ_g produced by the generator T . We then train Behavior Cloning (BC) policy using the sketch-generated demonstrations, ξ_D , achieving a score of 0.8 and thus leading to good performance in the SKETCH-TO-SKILL policy. The button position was randomized within a 20cm by 20cm to 25cm trapezoidal area visible from the wrist camera.
- **Toaster Press Task:** The Toaster Press experiment was designed to test the robustness of our approach in cluttered environments. We created 10 sketches based on the camera feeds and collected 10 sketch-generated demonstrations using open-loop servoing. Each episode featured a cluttered environment around the toaster, with objects commonly found in household settings to simulate realistic conditions. The initial position of the gripper was randomized in every episode to test the adaptability of the learned policy. Training a Behavior Cloning (BC) policy using the sketch-generated demonstrations achieved a preliminary success rate of 60%, and SKETCH-TO-SKILL training over 12k interactions achieved $\sim 90\%$ success rate within 10k interactions.
- **Bread Pick and Place Task** This task involved picking a piece of bread from a toaster and placing it on a nearby plate, requiring precise manipulation and handling. Sketches were specifically collected in randomly cluttered environments to reflect typical variability in real-world scenarios. Both the environment clutter and the initial gripper positions were

randomized in each episode, presenting a different challenge each time to test the robustness of the policy. The BC policy achieved a success rate of 36% and SKETCH-TO-SKILL trained over 30K interactions achieved $\sim 80\%$ success within 30K interactions.

These experiments underscore our method’s capability to handle real-world variability and complex task execution, supporting its potential utility beyond controlled experimental setups. The detailed results from these tasks, illustrated in Figures 16 and 17, highlight the practical applications and adaptability of our SKETCH-TO-SKILL framework in dynamic, cluttered settings.

2.6.5 Additional Details of Robomimic Experiments

2.6.5.1 Task Complexity

The PickPlaceCan task in RoboMimic is an demanding two-stage challenge. It requires the robot to accurately locate and reach a can, then pick it up and place it into a designated bin. This task not only tests the robot’s ability to handle objects with precision but also demands correct orientation of the gripper throughout the process. RoboMimic, a well-established benchmark, provides high-quality demonstrations collected via human teleoperation, which are instrumental for training successful policies.

2.6.5.2 Implementation and Strategy

In this setup, the sketch-based demonstrations provide initial positional cues for the robot. Our framework utilizes reinforcement learning (RL) to refine these initial cues, dynamically adjusting the robot’s approach to manage both the orientation and timing necessary for successful task execution. This method proves particularly effective as it does not rely on fully detailed

trajectory information from the start; instead, it uses the sketches to guide the initial exploration phase of RL, significantly simplifying the data collection process.

2.6.5.3 Results and Performance Metrics

- **Performance:** Our SKETCH-TO-SKILL framework, even with limited initial data, performs admirably in this demanding scenario. The results are on par with those from IBRL methods 98%, which benefit from complete and detailed human demonstrations, including explicit orientation details (as shown in Figure 15).
- **Comparison to IBRL:** The comparative success illustrates the robustness and effectiveness of our method in managing the task’s orientation and other complexities without fully specified trajectory inputs. This is particularly notable given that RoboMimic’s Pick and Place Can task offers a significantly higher level of difficulty than similar tasks in MetaWorld.

2.6.6 Additional Experiments in Metaworld Assembly

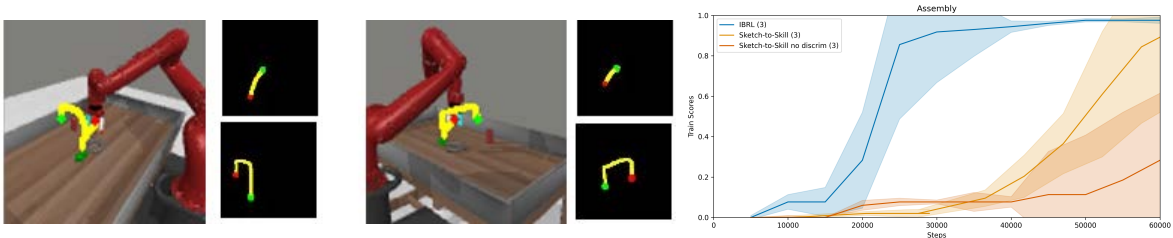


Figure 2.15: Left: sketches for the MetaWorld Assembly task. Right: training scores (success rate).

2.6.6.1 Task Complexity

The Assembly task in MetaWorld is a “hard” task [58] that requires the robot to execute two-stage manipulations. The task involves precise movements to pick up a peg, navigate it to a specific location, and insert it into a hole within a larger assembly fixture. This task tests the robot’s precision, spatial awareness, and ability to handle complex sequences.

2.6.6.2 Results and Performance Metrics

- **SKETCH-TO-SKILL:** Demonstrated a high success rate of 93%, effectively using sketches for coarse guidance to navigate and complete complex task sequences, even without actual teleoperated demonstrations.
- **SKETCH-TO-SKILL without Discriminator:** Initially struggled with a success rate of 20%, but extended interaction up to 100K steps improved performance dramatically, reaching a near-perfect success rate of 98%. This underscores the potential for learning even without discriminator guidance, given sufficient training time.
- **IBRL:** Achieved the highest success rates of approximately 100%, benefiting from high-quality teleoperated demonstrations that include precise details on orientation and positioning.
- **Standard BC:** Achieves a success rate of around 60%, showing limitations in environments where adaptive behaviors and fine-tuning through reinforcement learning are necessary.

These results highlight the effectiveness of the SKETCH-TO-SKILL approach in handling complex, multi-stage tasks through initial coarse guidance, with the potential for significant im-

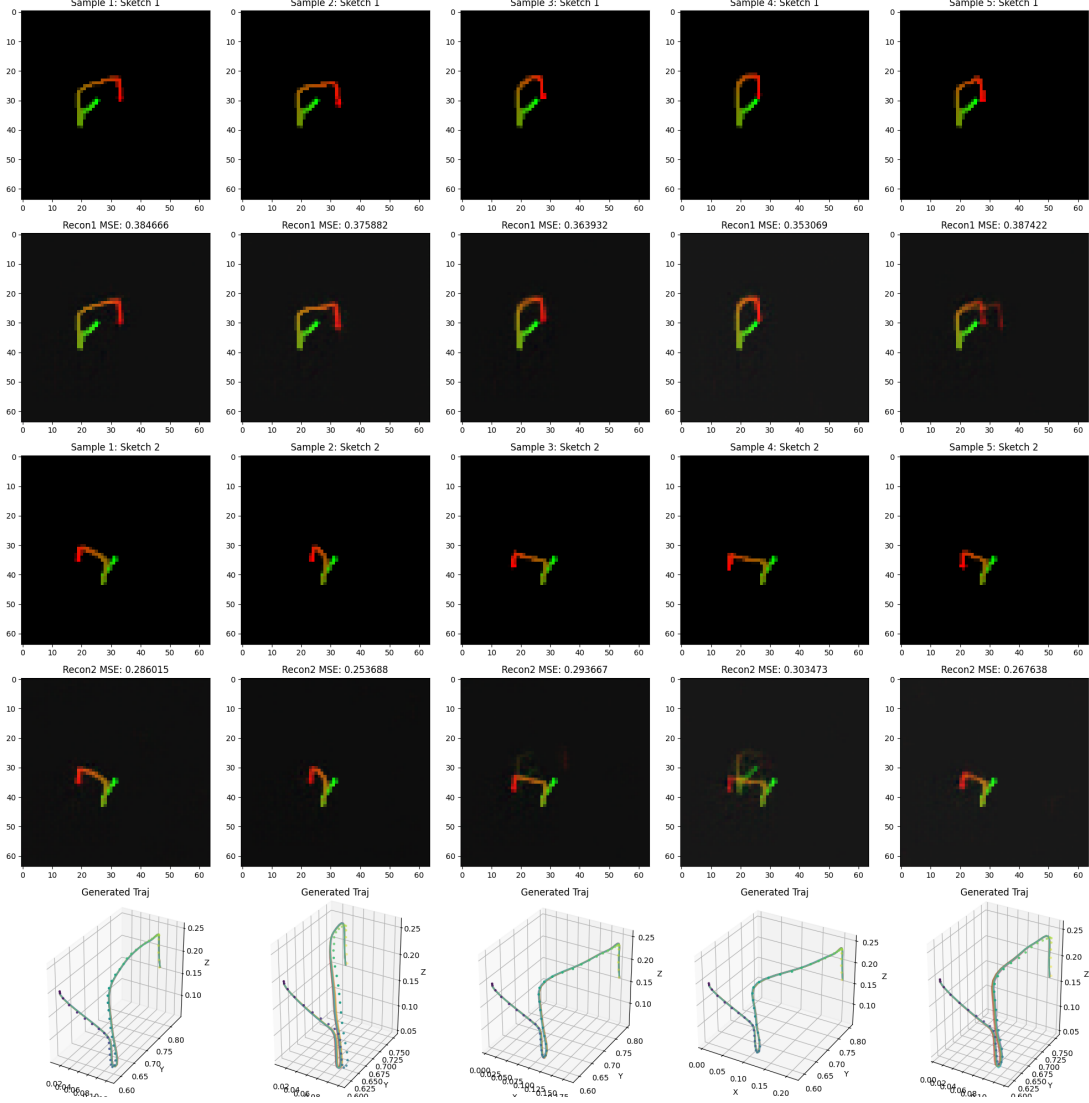


Figure 2.16: Trajectories with time-based color gradient for Assembly Metaworld simulation task.

provement over time. They also illustrate the advantage of incorporating discriminator feedback to accelerate learning and enhance performance.

2.6.7 Using Color Gradients for Overlapping Trajectories

We can also incorporate time-parameterization of the trajectory in the sketches using a color gradient, instead of a binary sketch image. This notion of color gradient in sketches was

introduced by RT-Trajectory [32]. An example is shown in Figure 2.16. Here, the color of the sketch changes from green (start) to red (end). This is particularly helpful when the sketch crosses each other or overlaps. We conduct additional experiments with the generator to evaluate the effect of incorporating gradients in the Sketch-To-3D trajectory generator.

Specifically, we use the MetaWorld Assembly task (Figure 2.15) where the sketch overlaps in the middle as the end-effector picks up the tool and carries it to the goal position. We trained two generators without and with color gradients. The performance of these two generators are reported in Table 2.3. We observe that incorporating the gradient in such a case results in lower reconstruction and trajectory losses. With lower trajectory losses we can handle more complex trajectories that overlap with color gradients. Note that incorporating gradients also does not require any change to the downstream architecture, and only requires minimal changes to the generator architecture.

Table 2.3: Performance Metrics for Overlapping and Non-Overlapping Trajectories

Metric	with color gradient	without color gradient
Training Loss	0.2438	0.3107
Validation Loss	0.3120	0.3585
Reconstruction Loss	0.0001	0.0003
KLD Loss	77.167	76.4683
Parameter MSE Loss	0.1028	0.1135
Trajectory loss	0.1530	0.2379

Chapter 3: Efficient Multi-Agent Reinforcement Learning with Personalized Expert Guidance

3.1 Introduction

The use of expert demonstrations¹ has been proven effective in accelerating learning in single-agent reinforcement learning, as evidenced by studies such as [55, 59, 60]. This approach has since been extended to Multi-Agent Reinforcement Learning (MARL) [61, 62], which typically assumes the availability of high-quality collaborative **joint demonstrations**. However, from a practical standpoint, collecting joint demonstrations can be labor-intensive, demanding one user per agent in cooperative scenarios. Furthermore, these demonstrations are not scalable. If we change the *number* of agents or introduce new *types* of agents, we will need to gather a new set of demonstrations to learn from.

In contrast, it is much easier to obtain demonstrations for individual agents, or even better, for each *type* of agents in a heterogeneous setting. We thus ask the following research question: *could we leverage individual-wise task demonstrations instead?* In this work, we refer to such expert demonstrations that address single-agent behaviors for personal objectives as **personalized demonstrations** (see Figure 3.1). Since the personalized demonstrations will not necessarily

¹In this context, the term “expert guidance” refers to demonstrations that provide meaningful, above-random performance in complex environments, even if the demonstrations are not necessarily optimal.

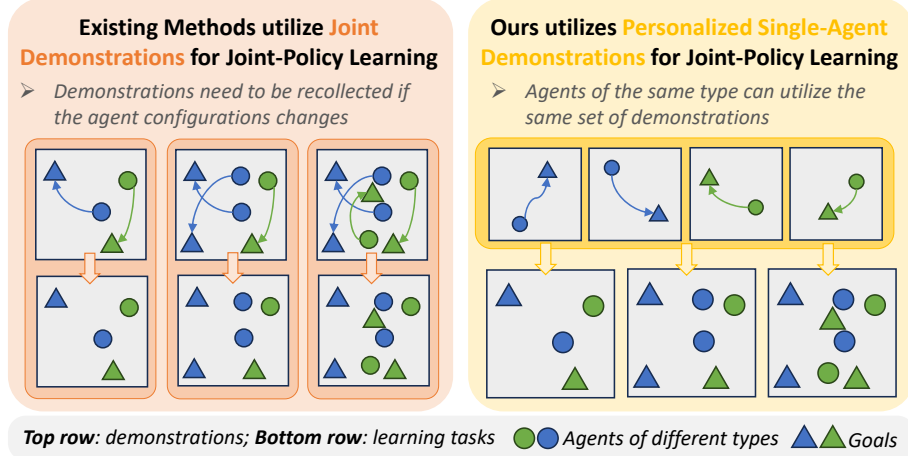


Figure 3.1: Joint demonstrations are costly to collect but offer rich information on collaborative behaviors. Personalized demonstrations are easier to collect, but solely focus on individual agent goals, so they lack cooperative elements.

reflect how the agents can collaborate and may even conflict with each other in the joint setting, naively mimicking the demonstrations will not achieve cooperation. Therefore, purely imitation learning-based approaches would not be effective. We need an approach that *selectively* utilizes suitable personalized expert demonstrations as guidance and allows agents to learn to cooperate via collecting reward signals from the environments.

To this end, we present our algorithm, Personalized Expert-Guided MARL (PegMARL), which carries out personalized occupancy matching as a form of guidance through reward-shaping. We implement this via two discriminators. The first, a personalized behavior discriminator, evaluates local state-action pairs, providing positive incentives for actions that align with the demonstration and negative incentives for divergent ones. The second, a personalized transition discriminator, assesses whether a local state-action pair induces a desired change in dynamics similar to that observed in the demonstration, adjusting the incentive weight accordingly. We demonstrate the effectiveness of PegMARL on both discrete gridworld and continuous multi-agent particle environments [63, 64]. The main contributions of this work are summarized as

follows:

(1) We propose PegMARL, the first approach that enables utilizing **personalized demonstrations** for policy learning in heterogeneous MARL environments, which (i) avoids demonstration recollection regardless of the *number* and *type* of agents involved, and (ii) is compatible with most MARL policy gradient methods.

(2) We demonstrate PegMARL’s effectiveness with **personalized demonstrations** in both **discrete** and **continuous** environments. Our algorithm outperforms state-of-the-art decentralized MARL algorithms, pure multi-agent imitation learning, and reward-shaping techniques in terms of scalability and convergence speed, and achieves robust performance even when provided with suboptimal demonstrations.

(3) We showcase PegMARL’s capability to also leverage **joint demonstrations**, regardless of whether they are sampled from co-trained or non-co-trained policies². Experimental results on the StarCraft environment [65] demonstrate that PegMARL converges effectively even with demonstrations from non-co-trained policies, which include potentially conflicting behaviors.

3.2 Related Works

Imitation Learning (IL). IL methods seek to replicate an expert policy from demonstration data generated by that policy. Behavior Cloning (BC) is a commonly employed IL technique [66, 67], where the expert policy is estimated through supervised learning on demonstration data. However, BC is susceptible to compounding errors resulting from distribution shifts [68]. Another thread of IL research is Inverse Reinforcement Learning (IRL) [69, 70], in

²“Co-trained policies” refers to policies that have been trained together in the same environment with shared experiences. We refer the readers to Figure 3.3 for an illustration and Section 6.2 of [19] for more information.

which the underlying reward function is estimated from the demonstration data and then used for policy learning. To alleviate the computational overhead of IRL, Generative Adversarial Imitation Learning (GAIL) [11] was introduced, allowing direct policy learning from observed data without intermediate IRL steps. Song et al. [20] extended this approach to introduce Multi-Agent GAIL (MAGAIL), which adapts GAIL to high-dimensional environments featuring multiple cooperative or competing agents. In general, IL approaches can rarely perform better than demonstrations. Therefore, they are not directly suitable for scenarios where only personalized expert demonstrations that do not demonstrate how to collaborate are available.

Learning from Demonstration (LfD). In contrast to IL methods, LfD aims to leverage demonstration data to facilitate learning rather than simply mimicking expert behavior. Existing LfD works incorporate demonstration data into a replay buffer with a prioritized replay mechanism to accelerate the learning process. Due to the off-policy nature of the demonstration data, most methods are value-based [71, 72]. There has been some recent work where the demonstrations are used to aid exploration, especially in environments with large state-action spaces [55, 59, 60]. For instance, POfD [55] learns an implicit reward from the demonstration data using a discriminator and incorporates it into the original sparse reward. LOGO [60] uses the demonstration data to directly guide the policy update: during each update iteration, the algorithm seeks a policy that closely resembles the behavior policy within a trust region. However, these approaches are primarily focused on single-agent settings.

In multi-agent settings, Qiu et al. [62] suggest using demonstrations to pretrain agents through imitation learning as a warm start, followed by optimization of the pretrained policies using standard MARL algorithms. Lee et al. [61] augment the experience buffer with demonstration trajectories and gradually decrease the mixing of demonstration samples during training

to prevent the learned policy from being overly influenced by demonstrations. Similar to POfD, DM² [19] enables agents to enhance their task-specific rewards by training discriminators as well. Each agent matches toward a target distribution of concurrently sampled trajectories from a joint expert policy to facilitate coordination. These approaches, however, require joint demonstrations of the same team configurations sampled from cohesively trained policies. While effective, this requirement can be cumbersome and limiting, especially in real-world scenarios where such coordinated demonstrations are difficult to obtain (as mentioned in Section 3.1).

Our approach differs in that we leverage only personalized expert demonstrations to learn a cooperative policy. Notably, PegMARL can also utilize joint demonstrations when they are available, including those from non-co-trained policies. By accommodating these diverse and potentially conflicting demonstrations, PegMARL can leverage a wider range of available data, offering greater flexibility and robustness across various multi-agent applications.

3.3 Preliminaries

We start by considering a Markov Decision Process $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$ for a cooperative multi-agent setting with N agents. Here, $\mathcal{S}$ denotes the global state across all the agents, which can be decomposed as the product of N local spaces $\mathcal{S}_i$ as $\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 \times \cdots \times \mathcal{S}_N$. By noting the local state of agent i as $s_i \in \mathcal{S}_i$, the global state is $s = (s_1, s_2, \cdots, s_N)$. Similarly, we define the global action space $\mathcal{A}$ as $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_N$, meaning that for any $a \in \mathcal{A}$, we may write $a = (a_1, a_2, \cdots, a_N)$ with $a_i \in \mathcal{A}_i$. The transition probability from state s to s' after taking a joint action a is denoted by $P(s'|s, a) = \prod_i P_i(s'_i|s, a)$. All agents share a common reward function $r = R(s, a)$, and $\gamma \in [0, 1]$ is the discount factor. In this work, we

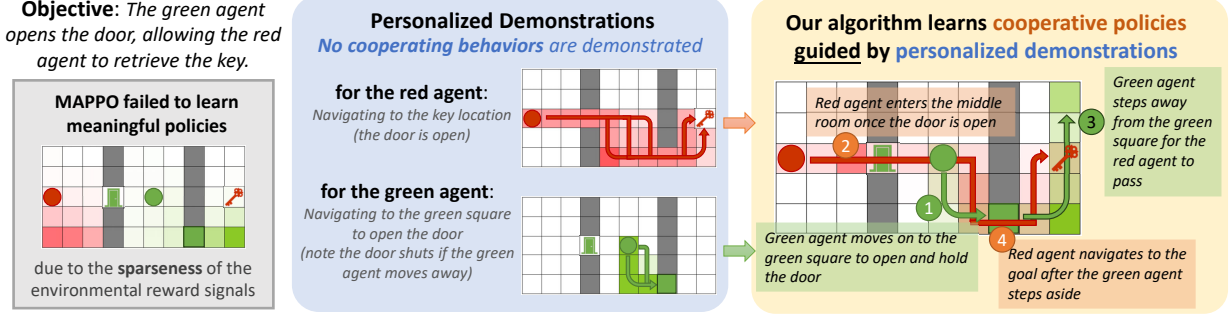


Figure 3.2: An example of utilizing **personalized** demonstrations to learn **cooperative** multi-agent policies. To learn successful cooperation, the agents are required not only to imitate the demonstrations to achieve personal goals but also to learn how to avoid conflicts and collaborate. We visualize the state visitation frequency of the personalized demonstrations and the joint policies learned by our algorithm and MAPPO, where a darker color means a higher value. We observe that the demonstrations guide the agents in exploring the state space more efficiently than in MAPPO.

focus on decentralized learning and define the global policy as $\pi_\theta(a|s)$, where $\theta \in \Theta$ are the policy parameters. Specifically, we have $\theta = (\theta_1, \theta_2, \dots, \theta_N)$ as the factorized global policy parameters, and we can write $\pi_\theta(a|s) = \prod_i \pi_{\theta_i}(a_i|s)$ using policy factorization.

Personalized Tasks and MDPs. To introduce the notion of **personalized demonstration**, we need to extract the individual tasks of each agent from the collective tasks of multiple agents. For example, in Figure 3.2, the green agent’s personalized task is to open the door and the red agent’s personalized task is to reach the key *without the other’s presence*. We then define Personalized Markov Decision Processes (PerMDPs) $(\mathcal{S}_i, \mathcal{A}_i, Q_i, r_i, \gamma)$ for each agent or, more generally, for each *type* of agents that share the same objective within a heterogeneous team. Here, a *type* refers to behaviorally identical agents who share the same state and action spaces, as well as the same reward function, using the formalism from [73], and we provide the PerMDP definition for each agent for simplicity. We assume that the state space $\mathcal{S}_i$ and action space $\mathcal{A}_i$ of the personalized task $\mathcal{T}_i$ are the same as the local state and action spaces of agent i from the joint task. The transition probability from state s_i to s'_i after taking a action a_i is represented as

$Q_i(s'_i|s_i, a_i)$. By following an arbitrary policy $\pi_i(a_i|s_i)$ for this personalized task, we can collect a set of personalized demonstrations $\mathcal{B}_i = \{(s_i^t, a_i^t)\}_{t=0}^H$, where H is the episode horizon. We'd like to emphasize that while we assume the personalized tasks and the joint task are conducted in the same environment map, the underlying transition dynamics are different. Additionally, the joint reward r may not necessarily equal the summation of rewards r_i for each personalized task.

Occupancy Measures. For a joint policy $\pi = (\pi_1, \pi_2, \dots, \pi_N)$, we can write the global state action occupancy measure $\lambda^\pi(s, a)$ as

$$\lambda^\pi(s, a) = \sum_{t=0}^{\infty} \gamma^t \cdot \mathbb{P}(s^t = s, a^t = a \mid \pi) \quad (3.1)$$

and write the corresponding local cumulative state-action occupancy measure as

$$\lambda_i^\pi(s_i, a_i) = \sum_{t=0}^{\infty} \gamma^t \cdot \mathbb{P}(s_i^t = s_i, a_i^t = a_i \mid \pi) \quad (3.2)$$

for $\forall a_i \in \mathcal{A}_i, s_i \in \mathcal{S}_i$. An interesting observation to note here is that we can write the local occupancy measure as the marginalization of the global occupancy measure with respect to all other agents. Mathematically, it holds that

$$\lambda_i^\pi(s_i, a_i) = \sum_{a \in \{a_i\} \times \mathcal{A}_{-i}} \sum_{s \in \{s_i\} \times \mathcal{S}_{-i}} \lambda^\pi(s, a) \quad (3.3)$$

with $\mathcal{A}_{-i} = \prod_{j \neq i} \mathcal{A}_j$ and $\mathcal{S}_{-i} = \prod_{j \neq i} \mathcal{S}_j$.

3.4 MARL with Personalized Expert Demonstrations

Now, we are ready to present the main problem we are interested in solving in this work. Assume each agent i is associated with a personalized task $\mathcal{T}_i$. We collect one set of expert demonstrations for each agent i or, equivalently, each personalized task $\mathcal{T}_i$. By letting an expert user perform each personalized task in the respective personalized MDP, we obtain a collection of expert demonstrations denoted by $\{\mathcal{B}_{E_1}, \mathcal{B}_{E_2}, \dots, \mathcal{B}_{E_N}\}$. We assume that the underlying expert policy associated with $\mathcal{B}_{E_i}$ is $\pi_{E_i}(a_i|s_i)$, and $\lambda^{\pi_{E_i}}$ is the occupancy measure following the expert's policy π_{E_i} for agent i . Note that while we establish the problem formulation and develop our algorithm in a fully observable setting, the experiments are conducted in both fully and partially observable settings, encompassing both discrete and continuous environments (details in Section 3.5).

3.4.1 Formulating Personalized Expert-Guided MARL

In standard Multi-Agent Reinforcement Learning, agents aim to discover optimal joint policies that maximize the long-term return $R(\pi_\theta) := \langle \lambda^{\pi_\theta}, r \rangle = \frac{1}{N} \sum_{i=1}^N \langle \lambda_i^{\pi_\theta}, r \rangle$. Typically, the learning process commences with random exploration, which is often inefficient due to MARL's exponentially growing exploration spaces, especially when rewards are sparse. Beginning with the intuition of leveraging personalized demonstrations as guidance for how each agent should accomplish their personalized tasks to promote more effective exploration, akin to [55] for single-agent cases, we can define the objective for learning from personalized demonstrations in multi-

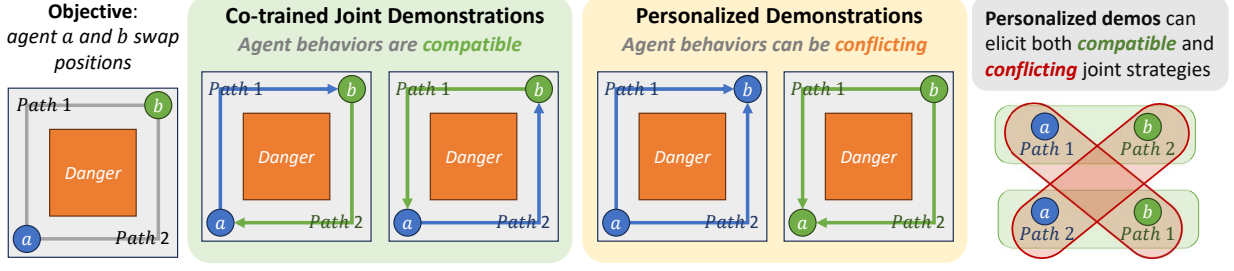


Figure 3.3: When joint demonstrations are sampled from co-trained policies, the agents’ behaviors exhibit compatibility. In contrast, personalized demonstrations solely focus on how each agent achieves its individual goal and lack cooperative elements, potentially leading to conflicts.

agent settings as follows:

$$\max_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^N \left(\langle \lambda_i^{\pi_\theta}, r \rangle - \eta \mathbb{D}_{\text{JS}} \left(\lambda_i^{\pi_\theta} \parallel \lambda^{\pi_{E_i}} \right) \right), \quad (3.4)$$

where η is a weighting term balancing the long-term reward and the personalized policy similarity. The Jensen-Shannon (JS) Divergence terms enable individual agents to align their actions with their respective personalized demonstrations and facilitate the achievement of their specific objectives.

However, only imitating the personalized demonstration may not always yield favorable outcomes and can even impede the learning process. Previous works in MARL have predominantly utilized joint demonstrations as guidance. As demonstrated by DM² [19], trajectories must be sampled from a *co-trained* joint expert policy for their joint action-matching objective to converge. The crucial aspect that jointly coordinated demonstrations offer, which personalized demonstrations lack, is compatibility. To illustrate, consider the example depicted in Figure 3.3, where two agents aim to swap positions without entering the danger region. Two sets of optimal joint policies exist for this task: agent *a* takes path 1 and agent *b* takes path 2, or vice versa.

When sampling joint demonstrations from a set of co-trained policies, the agents’ behaviors will be naturally compatible. In contrast, if personalized demonstrations were provided, agents could indifferently choose between both paths to reach their goals, potentially resulting in conflicting strategies through naive imitation.

Hence, we pose the following question: *How can we leverage the benefits of using personalized demonstrations to enhance exploration efficiency while circumventing the imitation of conflicting behaviors that might hinder joint-policy training?*

Consider the scenario depicted in Figure 3.4, where the blue agent seeks to replicate its personalized demonstration within a multi-agent environment. The blue agent can successfully transition to the neighboring grid on the right if it’s unoccupied; otherwise, the transition fails. Viewing the blue agent as the focal agent and the green agent as part of the environment, successful imitation hinges on the agent’s “local transition”

matching those of the environment where the personalized demonstrations originated. We define $P(s'_i | s_i, a_i)$ as an approximation for the i -th agent’s *local transition probability* in the multi-agent environment, which is governed by the true transition dynamics $P(s' | s, a)$

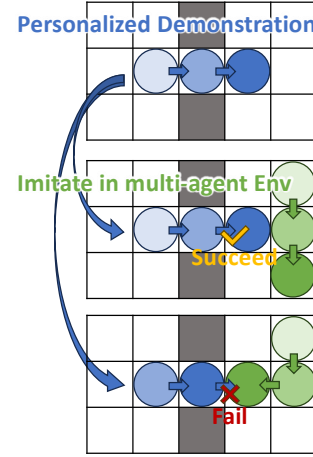


Figure 3.4: A motivating example illustrating the imitation of personalized demonstrations in a multi-agent environment. The primary technical challenge lies in the discrepancy between the transition dynamics in the personalized MDP and the local transition dynamics for each agent in the multi-agent environment.

and the policy π_θ . With this insight, we further refine the objective function as follows:

$$\max_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^N \left(\langle \lambda_i^{\pi_\theta}, r \rangle - \eta \mathbb{D}_{\text{JS}} \left(\hat{\lambda}_i^{\pi_\theta} \parallel \hat{\lambda}^{\pi_{E_i}} \right) \right), \quad (3.5)$$

where $\hat{\lambda}_i^{\pi_\theta}$ and $\hat{\lambda}^{\pi_{E_i}}$ have the same definition as the original $\lambda_i^{\pi_\theta}$ and $\lambda^{\pi_{E_i}}$ but with a restricted domain:

$$\text{dom}_{\hat{\lambda}_i^{\pi_\theta}} = \text{dom}_{\hat{\lambda}^{\pi_{E_i}}} = \{(s_i, a_i) \in \mathcal{S}_i \times \mathcal{A}_i \mid \mathbb{D}_{\text{JS}}(P_i(s'_i | s_i, a_i) \parallel Q_i(s'_i | s_i, a_i)) \leq \epsilon\}. \quad (3.6)$$

where $Q(s'_i | s_i, a_i)$ represents the true transition dynamics of the personalized MDP from which demonstrations are collected, and ϵ is a threshold parameter that theoretically controls transition mismatch tolerance. By adjusting the learning objective in this manner, the occupancy matching only happens on those local state-action pairs that guide us toward the desired next local state under the current policy.

3.4.2 Solving Personalized Expert-Guided MARL

Now, we can begin the development of a practical algorithm. By adopting Theorem 2 from [55]³, we can substitute the JS divergence between occupancy measures with

$$\mathbb{D}_{\text{JS}}(\lambda_i^{\pi_\theta} \parallel \lambda^{\pi_{E_i}}) \approx \sup_{D_i} \left(\mathbb{E}_{\pi_\theta}[\log(1 - D_i(s_i, a_i))] + \mathbb{E}_{\pi_{E_i}}[\log(D_i(s_i, a_i))] \right), \quad (3.7)$$

³We refer the reader to [55] for the full derivation. We use the same calculations with the change of notation $\rho_\pi \leftarrow \lambda_i^{\pi_\theta}$ and $\rho_E \leftarrow \lambda^{\pi_{E_i}}$.

where $D(s_i, a_i) : \mathcal{S}_i \times \mathcal{A}_i \rightarrow (0, 1)$ is a discriminative classifier to discern if the (s_i, a_i) pair is from the demonstration or the current policy. However, this doesn't account for the constraint in equation 3.6. Therefore, by introducing an indicator function $\mathbb{1}(s_i, a_i)$ for whether (s_i, a_i) belongs to the domain of $\hat{\lambda}_i^{\pi_\theta}$ and $\hat{\lambda}^{\pi_{E_i}}$, we can obtain

$$\mathbb{D}_{\text{JS}} \left(\hat{\lambda}_i^{\pi_\theta} \parallel \hat{\lambda}^{\pi_{E_i}} \right) \approx \sup_{D_i} \left(\mathbb{E}_{\pi_\theta} [\mathbb{1}(s_i, a_i) \cdot \log(1 - D_i(s_i, a_i))] + \mathbb{E}_{\pi_{E_i}} [\mathbb{1}(s_i, a_i) \cdot \log(D_i(s_i, a_i))] \right). \quad (3.8)$$

Since direct access to the transition distributions is unavailable, verifying whether each (s_i, a_i) pair is within the domain is not directly feasible. To address this, we further approximate the indicator function using another discriminative classifier $D_i(s_i, a_i, s'_i) : \mathcal{S}_i \times \mathcal{A}_i \times \mathcal{S}_i \rightarrow (0, 1)$, estimating the likelihood of a (s_i, a_i, s'_i) tuple being from the demonstrations. In essence, it quantifies the likelihood that the corresponding (s_i, a_i) induces a desired change in local state transition as observed in the demonstrations, effectively serving as a learned proxy for the ϵ -constrained domain in Equation 3.6.

After approximating the inner JS divergence term in equation 3.5 with two discriminators, we integrate the discriminative rewards derived from these discriminators into the environmental reward for the outer problem. Specifically, we estimate the reshaped reward $\hat{r}_i$ as $\hat{r}_i = r - \eta D_{\bar{\phi}_i}(s_i, a_i, s'_i) \log(1 - D_{\phi_i}(s_i, a_i))$, where ϕ_i and $\bar{\phi}_i$ are parameters for the two discriminators. The personalized *behavior* discriminator D_{ϕ_i} evaluates local state-action pairs, providing positive incentives for actions that align with the demonstration and negative incentives for divergent ones, while the personalized *transition* discriminator $D_{\bar{\phi}_i}$ assesses if a local state-action pair induces a desired transition in local state akin to that observed in the demonstration, adjusting the incentive weight accordingly. Subsequently, policy optimization is conducted by maximizing the long-term

Algorithm 2 Personalized Expert-Guided MARL (PegMARL)

Input: Number of agents N ; environment env ; personalized expert trajectories $\mathcal{B}_{E_1}, \dots, \mathcal{B}_{E_N}$; batch size M ; weight parameters $\{\eta^k\}$.

Initialize: Policies $\{\pi_{\theta_i}\}$, discriminators $\{D_{\phi_i}\}$ and $\{D_{\bar{\phi}_i}\}$, where $i = 1, 2, \dots, N$.

Output: Learned policies $\{\pi_{\theta_i}\}$.

- 1: **for** iteration $k = 0, 1, 2, \dots$ **do**
- 2: Gather trajectories of multi-agent rollouts from env , $\mathcal{B}^k = \text{ROLLOUT}(\pi, env)$.
- 3: **for** agent $i = 0, 1, \dots, N - 1$ **do**
- 4: Sample M tuples of (s_i, a_i, s'_i) from demonstration $\mathcal{B}^{E_i}$ and $\mathcal{B}^k$;
- 5: Update personalized behavior discriminator D_{ϕ_i} :

$$\max_{\phi_i} \left(\mathbb{E}_{\mathcal{B}^k} [\log(1 - D_{\phi_i}(s_i, a_i))] + \mathbb{E}_{\mathcal{B}^{E_i}} [\log D_{\phi_i}(s_i, a_i)] \right). \quad (3.9)$$

- 6: Update personalized transition discriminator $D_{\bar{\phi}_i}$:

$$\max_{\bar{\phi}_i} \left(\mathbb{E}_{\mathcal{B}^k} [\log(1 - D_{\bar{\phi}_i}(s_i, a_i, s'_i))] + \mathbb{E}_{\mathcal{B}^{E_i}} [\log D_{\bar{\phi}_i}(s_i, a_i, s'_i)] \right). \quad (3.10)$$

- 7: Estimate the reshaped reward as $\hat{r}_i^k = r - \eta^k D_{\bar{\phi}_i}(s_i, a_i, s'_i) \log(1 - D_{\phi_i}(s_i, a_i))$.
 - 8: Update agent policy π_{θ_i} .
 - 9: **end for**
 - 10: **end for**
-

return with the reshaped reward.

Our PegMARL algorithm, detailed in Algorithm 2, is compatible with any policy gradient methods, with MAPPO ([74]) adopted in our implementation. While presented in a fully observable setting, PegMARL can be adapted for partially observable scenarios. In such cases, we process observations from joint environment rollouts by removing dimensions not observable in the Personalized MDP for discriminators inputs. Notably, PegMARL can also utilize joint demonstrations; in this case, the observation processing step is unnecessary as both demonstrations and environment rollouts are derived from the same joint MDP. Compared to DM² [19], which also uses a discriminator-based approach for multi-agent learning from demonstrations, PegMARL introduces the personalized transition discriminator $D_{\bar{\phi}_i}$ as a key innovation. This ad-

ditional discriminator helps address the challenge illustrated in Figure 3.3 by evaluating whether local state-action pairs induce desired transitions similar to those in the demonstrations, effectively filtering out less desirable actions while retaining beneficial ones. This component enables PegMARL to handle both personalized demonstrations and joint demonstrations from various sources, including those sampled from non-co-trained policies, while DM² requires demonstrations from co-trained policies to achieve convergence.

3.5 Experiments

In this section, we empirically evaluate the performance of PegMARL, focusing on the following questions: (1) How does PegMARL, which leverages personalized demonstrations, compare to state-of-the-art MARL techniques? (2) How does PegMARL scale with an increasing number of agents and in the case of continuous state-action spaces? (3) How does the suboptimality of personalized expert demonstrations affect the performance of PegMARL? and (4) How does PegMARL perform when trained with joint demonstrations sampled from co-trained or non-co-trained expert policies?

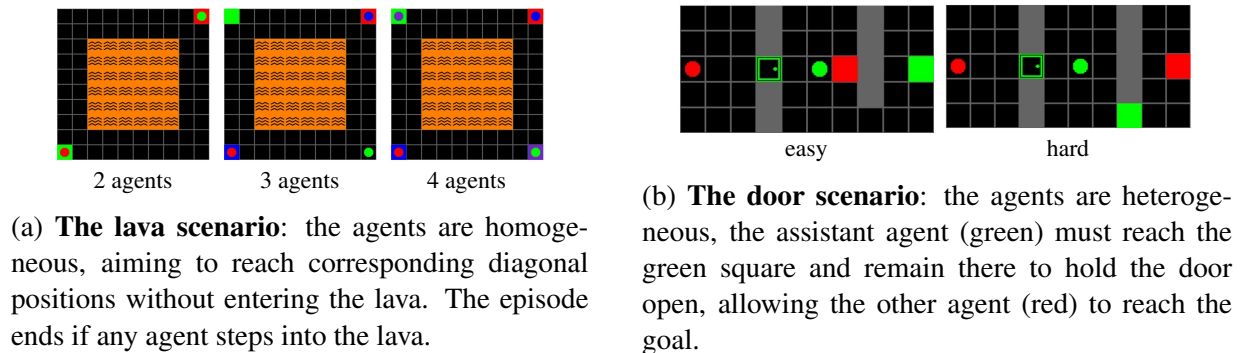


Figure 3.5: The discrete gridworld environments. Circles indicate the starting locations of the agents, while squares of the same color denote their respective goal locations. More details can be found in Appendix 3.7.1.1.

3.5.1 Main Results with Personalized Demonstrations

We evaluate PegMARL with personalized demonstrations on discrete gridworld environments (Figure 3.5) and a continuous multi-agent particle environment (MPE, Figure 3.8a). The gridworld environment is fully observable with discrete state and action spaces, while the MPE is partially observable with a continuous state space and discrete action space. Both environments feature sparse reward signals, which adds to the challenge for standard MARL algorithms to learn effectively.

Our comparison includes four strong baselines: MAPPO [74], a leading decentralized MARL algorithm; MAGAIL [20], a state-of-the-art multi-agent imitation learning algorithm; DM² [19], which combines distribution matching reward with environmental reward; and ATA [75], one of the best multi-agent reward-shaping methods. Notably, MAGAIL and DM² were not originally designed for personalized demonstrations. Hence, we have adapted them for use in personalized demonstration settings through necessary modifications. Specifically, we processed the observations from joint environment rollouts by removing dimensions not observable in the Personalized MDP, similar to our approach in PegMARL. This adjustment ensures that the inputs to their respective discriminators are compatible with the personalized demonstration data. In this context, MAGAIL serves as an ablation of PegMARL lacking the environmental signal and transition discriminator, while DM² represents an ablation lacking only the transition discriminator. Details on algorithm implementation and hyperparameter choices can be found in Appendix 3.7.3.

For each scenario, we collect two sets of personalized demonstrations for each agent: *optimal* and *suboptimal*, by training single-agent RL in simplified environments that focus on each

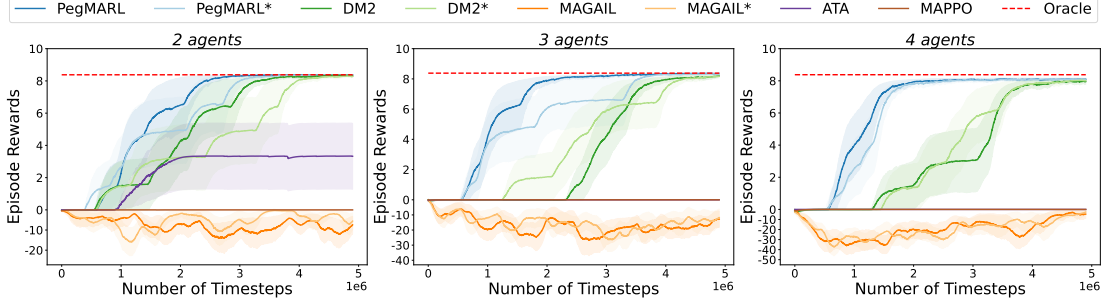


Figure 3.6: Learning curves of PegMARL versus other baseline methods under the **lava scenario**. PegMARL converges to higher rewards and generalizes better to larger numbers of agents. The star symbols (*) in the legend indicate that suboptimal personalized demonstrations are adopted.

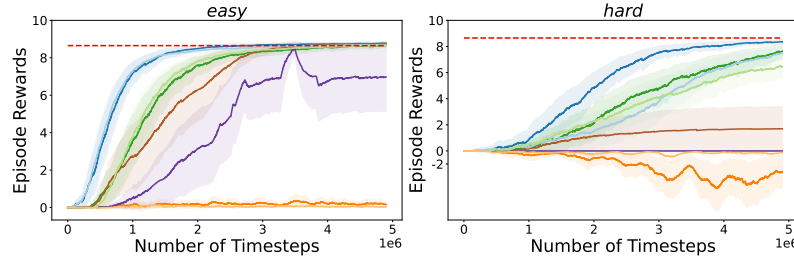


Figure 3.7: Learning curves of PegMARL versus other baseline methods under the **door scenario**. PegMARL shows better robustness in terms of convergence and generalizability.

agent’s specific tasks. The average episodic rewards of suboptimal demonstrations are approximately half of their optimal counterparts (more details can be found in Appendix 3.7.2). We execute each method in every environment with 10 distinct random initializations and plot the mean and variance across all runs. Oracle denotes the best possible return achievable with optimal policies.

How does PegMARL scale with an increasing number of agents? The lava scenario (Figure 3.5a) contains three variations, each involving different quantities of agents and escalating levels of complexity. The agents receive a positive reward only upon reaching the goal, incur penalties for collisions, and instantly die and terminate the episodes if they step into the lava, making learning in this environment challenging. As we observe from Figure 3.6, MAPPO struggles to develop meaningful behavior across all scenarios due to the sparse reward structure.

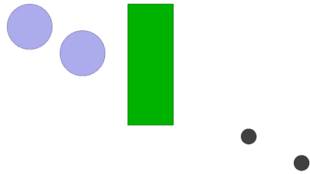
MAGAIL performs worse, primarily due to the absence of environmental reward signals. The naive imitation of personalized agent demonstrations leads to frequent collisions among agents. While ATA can learn suboptimal policies in the 2-agent setting, its learning efficacy declines drastically as the number of agents increases. This underscores the challenges of training as the complexity of the multi-agent environment increases.

In contrast, PegMARL demonstrates superior generalizability across scenarios with more agents, maintaining stable performance despite suboptimal demonstrations. As DM² has been adapted to accommodate personalized demonstrations, it closely resembles PegMARL but lacks the personalized transition discriminator component. As the number of agents increases, we would expect the space of possible interactions among agents to expand, and therefore for the personal demonstrations to be potentially misleading. Consequently, DM² experiences a decrease in convergence speed as the number of agents increases. This highlights the importance of the personalized transition discriminator in PegMARL, enabling effective handling of inter-agent interactions in personalized demonstration scenarios, thereby ensuring its efficacy across varying agent counts.

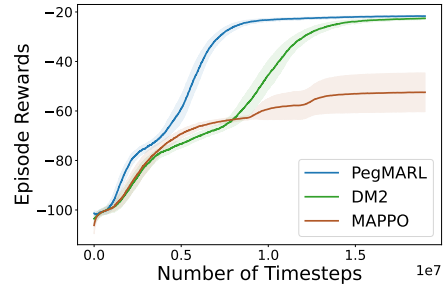
How does PegMARL perform under the heterogeneous setting? The door scenario (Figure 3.5b) contains two variants with varying levels of difficulty. The easy case is fairly straightforward: success should be achievable by adhering to personalized demonstrations, which illustrate how each agent navigates to their respective goal locations. As shown in Figure 3.7, most algorithms, except for MAGAIL, showcase proficient performance. Notably, PegMARL exhibits the swiftest convergence. We attribute MAGAIL’s failure in this case to its inability to direct the green agent to remain positioned at the green square — a behavior not explicitly demonstrated. This underscores the importance of environmental reward signals when integrating

personalized demonstrations into multi-agent learning paradigms. The hard case necessitates a higher degree of agent cooperation: once the red agent gains entry to the middle room, the green agent must move aside from the green square to enable the red agent’s passage into the right room. In this complex setting, only PegMARL and DM² demonstrate commendable convergence. PegMARL, equipped with the personalized transition discriminator, maintains faster convergence compared to DM².

How does PegMARL perform in the continuous setting? We modified the cooperative navigation task from the multi-agent particle environment ([63, 64]) to evaluate the performance of our algorithm in a continuous environment (more details can be found in Appendix 3.7.1.2). Figure 3.8b demonstrates the learning curves of PegMARL versus MAPPO and DM². This validates the ability of PegMARL in continuous environments.



(a) The modified cooperative navigation scenario: the agents need to navigate around the wall to occupy both landmarks.



(b) Learning curves of PegMARL versus MAPPO and DM² under the modified cooperative navigation scenario

Figure 3.8: (a) The modified cooperative navigation scenario. (b) The learning curves demonstrate that PegMARL is effective in continuous environments.

3.5.2 Results with Joint Demonstrations

While PegMARL is primarily designed to leverage personalized demonstrations, it can also be extended to utilize joint demonstrations. Unlike other MARL algorithms like DM² [19] that re-

quire compatible joint demonstrations sampled from co-trained policies to achieve convergence, PegMARL has the flexibility to utilize joint demonstrations even from non-co-trained policies, which potentially contain conflicting behaviors. To demonstrate, we conduct comparative experiments within the StarCraft Multi-Agent Challenge (SMAC) environment [65]. We adopted DM²'s original implementation and parameters for accurate results replication. We employed the same two tasks that were used in DM²: 5mv6m, which involves 5 Marines (allies) against 6 Marines (enemies), and 3sv4z, which features 3 Stalkers (allies) versus 4 Zealots (enemies). The win rates of the demonstrations in both cases are approximately 30%.

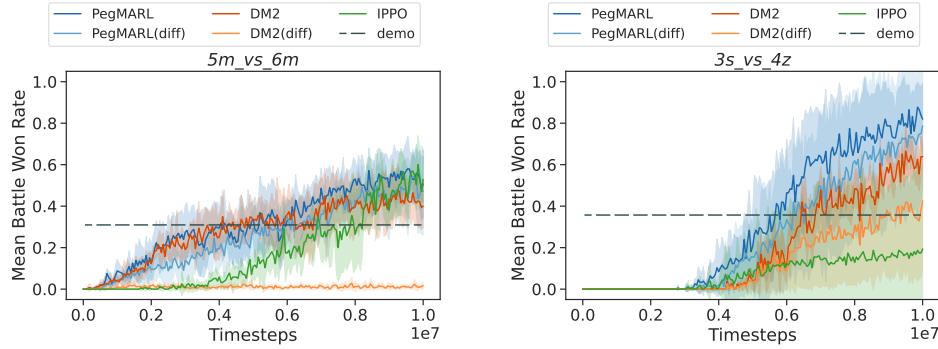


Figure 3.9: Learning curves of PegMARL versus DM² in two tasks under the **SMAC scenarios**. The suffix “diff” in the legend indicates that the joint demonstrations used are sampled from non-co-trained policies. Otherwise, the demonstrations are sampled from co-trained policies.

How does PegMARL scale with joint demonstrations? Figure 3.9 presents the learning curves of PegMARL alongside the DM² baseline in both the 5mv6m and 3sv4z configurations within the SMAC environment. When the joint demonstrations are sampled from **co-trained** policies, PegMARL achieves a comparable, and in some cases even higher, success rate compared to DM² in both tasks.

When the joint demonstrations are sampled from policies that are **non-co-trained**, the success rate of DM² exhibits a significant decline in both tasks. In the 5mv6m task, DM²'s success

rate drops to nearly 0, indicating a failure to learn. In contrast, PegMARL maintains a similar level of performance compared to results from co-trained demonstrations, with only a slight decrease in convergence speed. These findings underscore the versatility and effectiveness of PegMARL in scenarios requiring collaboration among diverse agents, demonstrating its applicability and robustness across various multi-agent scenarios and demonstration types.

3.6 Conclusions

In this work, we introduce PegMARL, a novel approach for Multi-Agent Reinforcement Learning, which adopts personalized expert demonstrations as guidance and allows agents to learn to cooperate. Specifically, the algorithm utilizes two discriminators to dynamically reshape the reward function: one provides incentives to encourage the alignment between the policy behavior with provided demonstrations, and the other regulates incentives based on whether the behavior leads to the desired objective. We demonstrate PegMARL’s effectiveness, scalability, and robustness in both discrete and continuous environments, where it outperforms state-of-the-art decentralized MARL algorithms, pure imitation learning, and reward-shaping techniques. We observe that PegMARL can achieve near-optimal policies even with suboptimal demonstrations. Furthermore, We showcase PegMARL’s capability to leverage joint demonstrations and converge successfully, regardless of whether they are sampled from co-trained or non-co-trained policies.

Applicability and Limitations: PegMARL is most applicable when: a) The task allows for some degree of independent decision-making, allowing individual agent behaviors to be isolated and demonstrated, even if only partially. b) The environment provides feedback on cooperative behaviors, enabling agents to learn coordination beyond individual demonstrations. For example,

in the door scenario, agents gain exploration guidance for door operation and navigation from personalized demonstrations. At the same time, environmental feedback on joint success drives them to learn coordination about timing and positioning (e.g., the green agent learning to make way for the red agent). Similarly, in the lava scenario, personalized demonstrations provide guidance for lava-avoiding paths. Still, environmental penalties for collisions and rewards for collective goal achievement encourage the emergence of cooperative navigation strategies.

However, PegMARL may face challenges in tasks requiring continuous, seamless collaboration among agents, such as cooperative object lifting and relocation by multiple agents. In these scenarios, where constant, integrated teamwork is crucial, the reliance on personalized demonstrations may not provide sufficient guidance for complex, ongoing coordination. Despite this, PegMARL’s capability to utilize joint demonstrations from diverse sources, including non-co-trained policies, suggests the potential for addressing this limitation. It would be interesting future work to explore PegMARL’s capabilities to handle more intricate and continuously collaborative tasks.

Another limitation of our work is the absence of formal convergence guarantees. While DM² establishes convergence for individual agents optimizing distribution matching objectives, our approach introduces additional complexity through the transition discriminator’s dynamic adjustment of demonstration influence. The interplay between the dual discriminators and multi-agent dynamics poses challenges for theoretical analysis, particularly since the joint distribution cannot be directly recovered from individual agent demonstrations in our setting. While our extensive empirical results demonstrate robust convergence across diverse scenarios, establishing theoretical convergence guarantees remains an interesting direction for future work.

3.7 Appendix

3.7.1 Environment Details

3.7.1.1 Discrete Gridworld Environments

We use two gridworld environments with discrete state and action space: the lava scenario and the door scenario (Figure 3.5). The agents in the lava scenario are homogeneous because they have the same objective: navigating to their corresponding goal location. The door scenario has heterogeneous agents: the assistant agent (green) must open the door while the other agent (red) must reach the goal.

- **The lava scenario:** This environment has a 6-by-6 lava pond in the center (Figure 3.5a). We provide three variations of this scenario, each with varying numbers of agents and increasing levels of complexity. The main goal for the agents is to efficiently navigate to their assigned goal locations while avoiding stepping into the lava. An episode terminates when all agents reach their respective goals (succeed), or if any agents step into the lava or the maximum episode length is reached (fail).
- **The door scenario:** This environment is adapted from [76] (see Figure 3.5b). In this scenario, the green agent must navigate to a designated green square and maintain its presence there to sustain the open state of the green door, thereby enabling the entry of a red agent into the right side room. An episode ends when the red agent reaches the red goal location (succeed) or the maximum episode length is reached (fail).

Each agent’s local state space is its $\{x, y\}$ coordinates in the map. We concatenate all the local states together to form the global state and assume all agents have access to the global state,

which has a dimension of $\mathbb{R}^{n \times 2}$ (n is the agent number). The local action space includes five actions: left, right, up, down, and stay. A sparse reward is granted when an episode succeeds, while a small penalty will be subtracted according to the steps taken ($10 - \text{step_count}/\text{max_step}$). Agents will receive a penalty of -1 if they collide with each other.

3.7.1.2 Cooperative Navigation

We modified the cooperative navigation task of multi-agent particle environment [63, 64] to evaluate the performance of our algorithm in a continuous environment. The modified environment consists of 2 agents, 2 goal landmarks and 1 wall between the agents and the goals. The agents need to navigate around the wall to occupy both landmarks. Each agent has a partial observation of the environment, including its position and velocity as well as the relative positions of other agents and landmarks. The action space is discrete, consisting of moving left, right, up, down, and staying in place. We sparsify their original reward as follows:

$$r = \sum_{a \in A} \min(0.3, \min_{g \in G} d(a, g)) + \sum_{g \in G} \min(0.3, \min_{a \in A} d(a, g)). \quad (3.11)$$

3.7.1.3 The SMAC Environment

The StarCraft Multi-Agent Challenge (SMAC) [65] is an environment specifically designed for evaluating multi-agent reinforcement learning (MARL) algorithms. Built on the StarCraft II game, SMAC presents a variety of challenging scenarios where multiple agents must collaborate to control individual units in combat. Each agent has partial observation of its surroundings within a certain sight range. The action space is discrete, including move[direction], at-

tack[enemy_id], stop, and no-op. A global reward is given at each timestep based on the total damage dealt to the enemy units, with an additional bonus of 10 or 200 awarded for killing each enemy unit or for winning a combat respectively. We consider the same two tasks as used in DM2 [19]:

- **5m_vs_6m**: The allied team consists of 5 Marines and the enemy team controls 6 Marines.
- **3s_vs_4z**: The allied team controls 3 stalker units and faces 4 enemy Zealot units.

3.7.2 Demonstration Details

3.7.2.1 Personalized Demonstrations

We let each individual agent perform their designated task in the same environment map without the other agents’ presence to collect personalized demonstrations. This can be implemented through two main methods: a) recording expert human demonstrations for each agent type, and b) training single-agent RL in simplified environments that focus on their specific tasks. In our experiments, we adopted the latter option. It takes about 2 hours to train a single-agent PPO for each personalized task.

Discrete Gridworld Environments: Figure 3.10 shows an example of personalized demonstrations for the lava scenario, and Figure 3.2 visualizes the personalized demonstrations for the door scenario. We summarize the details of the suboptimal demonstrations for the gridworld environment in Table 3.1, where the average episodic rewards are approximately 4.5, about half of their optimal counterparts.

We provide visual representations of the policy and state occupancy measure corresponding to suboptimal demonstrations in Figure 3.11 for the lava scenarios and in Figure 3.12 for the door

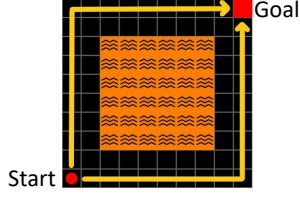


Figure 3.10: An example of personalized demonstrations for the lava scenario (we did not visualize all the optimal paths). There is only one agent in the environment. The agent may take both the left and the right path toward the goal.

Agent Id	$\mathcal{S}$	$\mathcal{A}$	Samples	Average Episodic Reward
1	$\mathbb{R}^2$	5	300	4.42
2	$\mathbb{R}^2$	5	440	4.79
3	$\mathbb{R}^2$	5	344	4.3
4	$\mathbb{R}^2$	5	360	4.41

(a) The lava scenario

Case	Agent	$\mathcal{S}$	$\mathcal{A}$	Samples	Average Episodic Reward
easy	red	$\mathbb{R}^2$	5	643	4.01
	green	$\mathbb{R}^2$	5	612	4.39
hard	red	$\mathbb{R}^2$	5	607	4.34
	green	$\mathbb{R}^2$	5	593	4.36

(b) The door scenario

Table 3.1: The details of suboptimal demonstrations for the gridworld environment.

scenario. The red square symbolizes the agent’s initial position in these visualizations, while the green square designates its respective goal location. Arrows within the figures denote available actions at each state, with arrow length indicating the probability associated with each action.

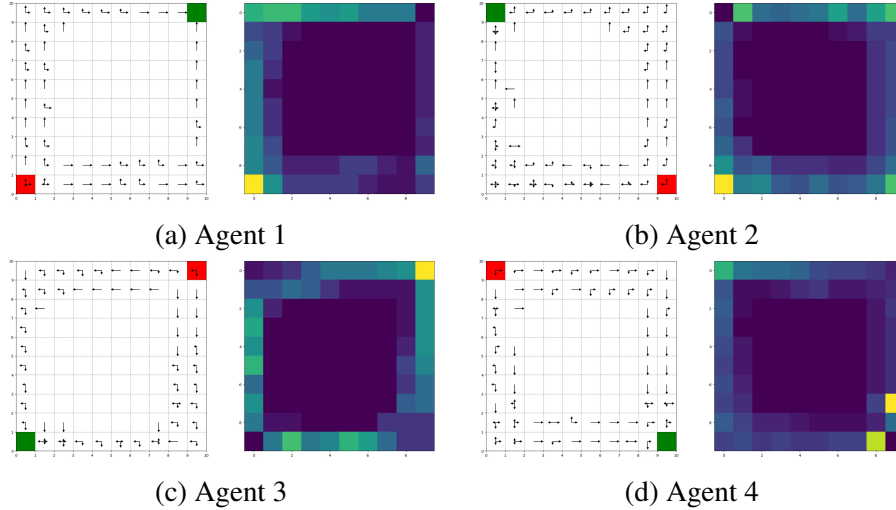


Figure 3.11: Agent policies and state occupancy measures estimated from the suboptimal demonstrations for the **lava scenario**.

Cooperative Navigation: For the cooperative navigation task from the multi-agent particle environment ([63, 64]), the collaborative goal is for the two agents to cover both goals,

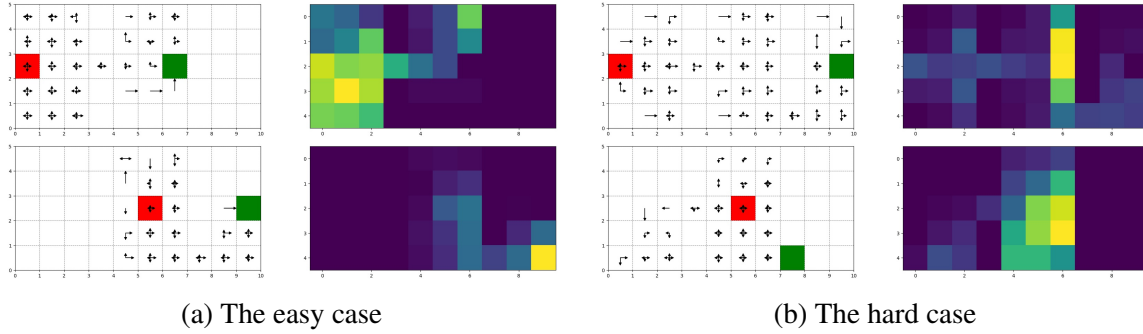


Figure 3.12: Agent policies and state occupancy measures estimated from the suboptimal demonstrations for the **door scenario**. The top row is for the red agent, and the bottom row is for the green agent.

regardless of which agent covers which goal. Therefore, we design the personalized task such that a single agent covers either of the two goals without the other agent’s presence. An illustration of personalized demonstration for this task is shown in Figure 3.13.

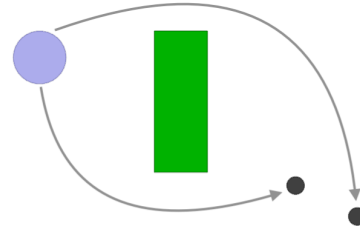


Figure 3.13: An illustration of personalized demonstrations for the cooperative navigation task. There is only one agent in the environment. The agent’s objective is to reach one of the two goal locations.

3.7.2.2 Joint Demonstrations

The SMAC Environment: Regarding the joint demonstrations utilized in the SMAC environment [65], we adopted the demonstrations provided by the authors of DM² [19]. For joint demonstrations sampled from co-trained policies, they are derived from jointly trained expert policies that achieve approximately a 30% win rate. For joint demonstrations sampled from non-co-trained policies, they are obtained from expert policies that were trained independently in separate teams but executed together in the same environment.

3.7.3 Network architecture and hyperparameters

3.7.3.1 Training with Personalized Demonstrations

The algorithms are implemented based on MAPPO [74], with each agent having separate policy and critic networks, discriminators, and optimizers. Both the policy and critic networks are two-layer MLPs with a hidden dimension of 64 and Tanh activation. The discriminators, including the personalized behavior discriminator and the personalized transition discriminator, are three-layer MLPs with a hidden dimension of 64 and Tanh activation. The personalized behavior discriminator takes (s_i, a_i) as input, while the personalized transition discriminator takes (s_i, a_i, s'_i) as input.

In partially observable settings, let's assume that the agent's observation in the multi-agent MDP is denoted as $o_i \in \mathcal{O}_i$, while in the PerMDP, it is denoted as $o'_i \in \mathcal{O}'_i$. Since the PerMDP involves a single agent and the multi-agent MDP involves multiple agents, the observation space for each agent cannot be the same in these two settings. For experiments on cooperative navigation, we use a heuristic function to convert o_i by removing the dimensions that is not within the observation space $\mathcal{O}'_i$ before passing them to the discriminators.

All baseline algorithms, except ATA, use the same codebase we implemented. MAGAIL is trained with the personalized behavior discriminator but does not use the environmental reward. DM2 uses both the environmental reward and the personalized behavior discriminator. PegMARL utilizes the environmental reward, the personalized behavior discriminator, and the personalized transition discriminator. ATA's original implementation is adopted for comparison. We summarized the common hyperparameters used for PegMARL, DM2 and MAGAIL in Table

3.2. The *gail rew coef* in DM2 is set to be 0.02 for all the gridworld scenarios and 0.1 for the cooperative navigation task. The η coefficient in PegMARL is 0.05 for all the gridworld scenarios and 0.2 for the cooperative navigation task.

epochs	4
buffer size	4096
clip	0.2
lr	0.0001

Table 3.2: MAPPO Hyperparameters

3.7.3.2 Training with Joint Demonstrations

We implement PegMARL based on the original DM2’s codebase to conduct experiments in the SMAC environment, augmenting it with our personalized transition discriminator. The hyperparameters for IPPO and GAIL were set to be identical to those used in DM2, as shown in Table 3.3 and 3.4. PegMARL adopted the same hyperparameters as DM2.

	5v6	3sv4z
epochs	10	15
buffer size	1024	1024
gain	0.01	0.01
clip	0.05	0.2

Table 3.3: DM2’s IPPO Hyperparameters

	5v6	3sv4z
gail rew coef	0.3	0.05
discr epochs	120	120
buffer size	1024	1024
batch size	64	64
n exp eps	1000	1000

Table 3.4: DM2’s GAIL Hyperparameters

3.7.3.3 The influence of the weighting term η

In our experiments, we chose η through a grid search to optimize performance. Interestingly, we found that PegMARL is relatively insensitive to η , which we attribute to the dynamic weighting provided by the personalized transition discriminator. This discriminator adaptively

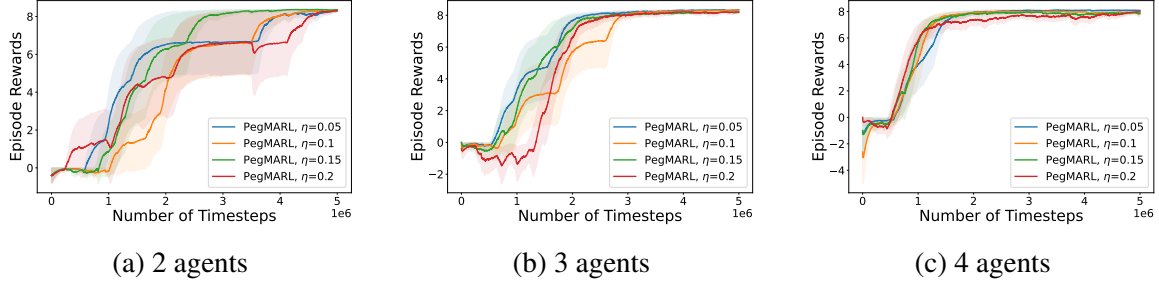


Figure 3.14: Ablation study over the weighting term η on the lava environment using optimal personalized demonstrations.

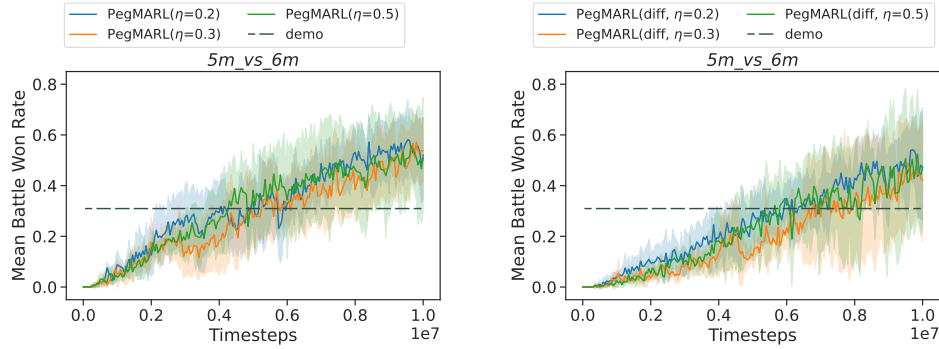


Figure 3.15: Ablation study over the weighting term η on the 5m_vs_6m SMAC map. Left: results using joint demonstrations sampled from co-trained policies; Right: results using joint demonstrations sampled from non-co-trained policies.

adjusts the influence of demonstration alignment based on the likelihood of local state-action pairs leading to desired outcomes, thereby mitigating the impact of suboptimal η values.

3.7.4 Additional Visualizations

We additionally depict the state visitation frequencies of the joint policies learned by PegMARL with suboptimal demonstrations and MAPPO for both the lava (Figure 3.16) and the door scenario (Figure 3.17).

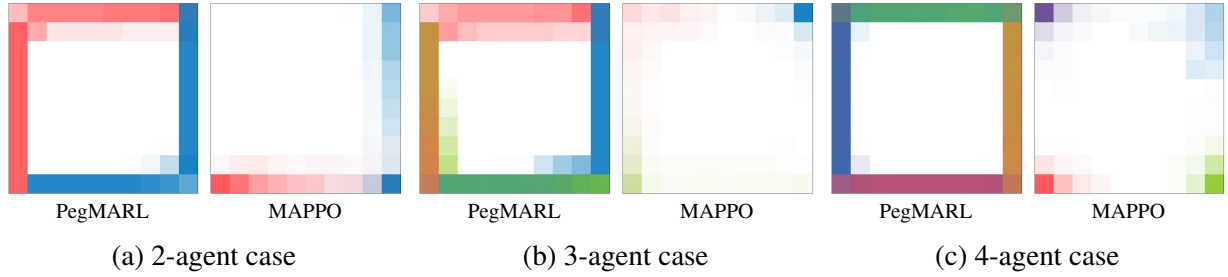


Figure 3.16: State visitation frequency of the joint policies learned by PegMARL (with suboptimal demonstrations) and MAPPO for the **lava scenario**. The darker color means a higher value. MAPPO failed to learn any meaningful policies in all three settings.

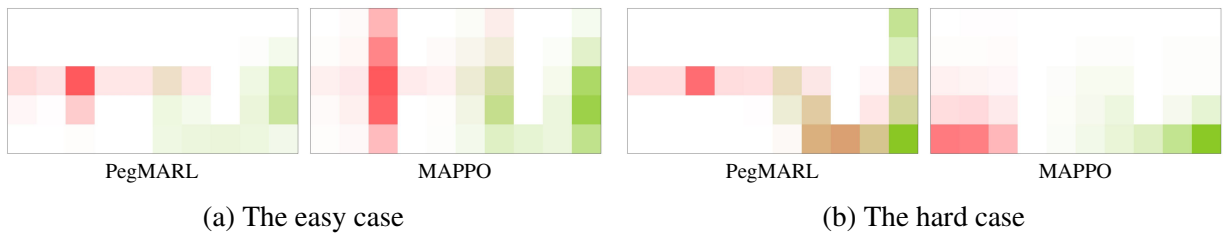


Figure 3.17: State visitation frequency of the joint policies learned by PegMARL (with suboptimal demonstrations) and MAPPO for the **door scenario**. The darker color means a higher value. MAPPO converges to a suboptimal policy in the easy case, while failing to learn in the hard case.

Chapter 4: Efficient Multi-Agent Reinforcement Learning with Demonstrations from Various Team Sizes

4.1 Introduction

Training multi-agent systems to collaborate effectively remains challenging due to the exponential growth of the joint state-action space. Demonstration-guided learning [55, 59, 60] has emerged as a promising solution by offering expert behavior as guidance, but most existing methods rely on joint demonstrations [61, 62, 19], which are difficult to collect, scale, and reuse across scenarios.

To address these challenges, our earlier work proposed PegMARL [23], a novel framework that leverages *personalized expert demonstrations*, where each agent or agent type performs its own task in isolation. When using personalized demonstrations, PegMARL processes observations from joint environment rollouts by removing dimensions not observable in the single-agent Personalized MDP, enabling the framework to learn from individual behaviors within multi-agent environments. To avoid blindly imitating potentially conflicting individual behaviors, PegMARL uses two discriminators: one encouraging behavioral similarity with demonstrations, and another verifying whether these behaviors lead to desirable outcomes. This setup allows agents to extract useful exploration signals while still learning to coordinate through environmental feed-

back. It also enables PegMARL to utilize joint demonstrations from the same team size, even when sourced from non-co-trained policies that exhibit conflicting behaviors.

Yet there are compelling reasons to reuse demonstrations from various team sizes. While the global strategy may vary with the number of agents, local coordination patterns — such as collision avoidance, spatial coordination, and formation behavior — tend to be consistent and transferable. For example, an agent’s demonstrated ability to yield at narrow corridors or avoid interference with nearby teammates can benefit larger or smaller teams alike. These reusable micro-patterns offer an opportunity to generalize demonstration-guided learning across diverse team settings. Unfortunately, the original PegMARL cannot directly take advantage of such demonstrations. When the demonstration and target scenarios differ in team size, it becomes unclear how to align the input dimensions. Removing arbitrary ally or enemy features is not a reliable solution, as it disrupts the structure of observations and breaks consistency between training and execution.

To tackle this limitation, we present **DynPegMARL**, an *architectural extension of PegMARL* designed to handle demonstrations from various team sizes when they are available. Specifically, we develop a graph neural network (GNN)-based observation encoder that separately processes ally and enemy features in a permutation-invariant fashion, which allows us to extract these underlying patterns from demonstrations and apply them flexibly across different configurations. We validate our approach through comprehensive experiments on SMACv2. Our experimental results show that this extended architecture enables agents to learn effective cooperative behaviors even when trained with demonstrations from different team configurations. This extension preserves the benefits of PegMARL and significantly improves its adaptability in realistic multi-agent settings.

4.2 Method

4.2.1 Background: Dec-POMDP with Entities

We model the multi-agent environment as a *Decentralized Partially Observable Markov Decision Process* (Dec-POMDP) with *entities*, defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, P, r, \mathcal{E}, \Phi)$, where: $\mathcal{S}$ is the set of global states, $\mathcal{A}$ is the set of joint actions, $\mathcal{O}$ is the observation space, $P : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$ is the transition function, $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the shared reward function, $\mathcal{E}$ is the set of entities in the environment, Φ is the set of entity types. Each entity $e \in \mathcal{E}$ has a full state representation $s^e \in \mathbb{R}^{d_e}$, and an observable feature vector $f^e \in \mathbb{R}^{d_{fe}}$ that contains only the information visible to other agents. Each entity is also associated with a type label $\phi^e \in \Phi$, indicating its category (e.g., agent, enemy, obstacle). The global state is defined as the set of all entity states: $s = \{s^e \mid e \in \mathcal{E}\}$.

At each timestep, agent i receives a partial observation composed of its own state and features from a subset of observable entities. We define each agent’s observation as a set: $o_i = \{s^i\} \cup \{f^j \mid j \in \mathcal{E}_i\}$, where $\mathcal{E}_i \subseteq \mathcal{E} \setminus \{i\}$ denote the set of observable entities excluding the agent itself. This observation structure is inherently unordered and of variable length. In many existing MARL frameworks, such inputs are processed by imposing a fixed entity ordering and applying zero-padding or masking to maintain a consistent input shape, making them compatible with standard MLP-based architectures.

4.2.2 DynPegMARL: Extending Demonstration-Guided Learning Across Team Sizes with GNN-Based Observation Encoding

Our approach builds upon PegMARL by extending its architecture to handle demonstrations from variable team sizes. PegMARL employs two discriminators to guide multi-agent learning: a *behavior* discriminator $D_{\phi_i}(s_i, a_i)$ that evaluates state-action alignment with demonstrations, and a *transition* discriminator $D_{\bar{\phi}_i}(s_i, a_i, s'_i)$ that assesses whether transitions match demonstration patterns. The framework reshapes environmental rewards as $\hat{r}_i = r - \eta D_{\bar{\phi}_i}(s_i, a_i, s'_i) \log(1 - D_{\phi_i}(s_i, a_i))$, allowing agents to learn coordination through environmental feedback while being guided by demonstration data. In practice, since agents operate under partial observability, we replace states s_i with local observations o_i throughout the implementation.

PegMARL’s original architecture processes each agent’s observation $o_i \in \mathbb{R}^d$ as a fixed-length flat vector that concatenates the agent’s own full state representation s^i , and the features of all allies $\{f^j\}_{j \in \mathcal{N}^{\text{ally}}}$ and enemies $\{f^k\}_{k \in \mathcal{N}^{\text{enemy}}}$, where the entities outside the visibility range are represented as zero vectors. The total dimensionality of o_i depends on the number of allies and enemies in the environment, which varies across team configurations. Consequently, the input size of the network is tied to the specific team configuration. When demonstrations o_i^{demo} are collected under different team sizes than the current environment o_i^{env} , their observation structures may differ in both length and semantic layout. Such mismatches make it infeasible to directly apply discriminators across varying team sizes without architectural changes.

To address this limitation, we adopt a Graph Neural Network (GNN)-based encoder that processes variable-length observations in a structured and permutation-invariant manner. As

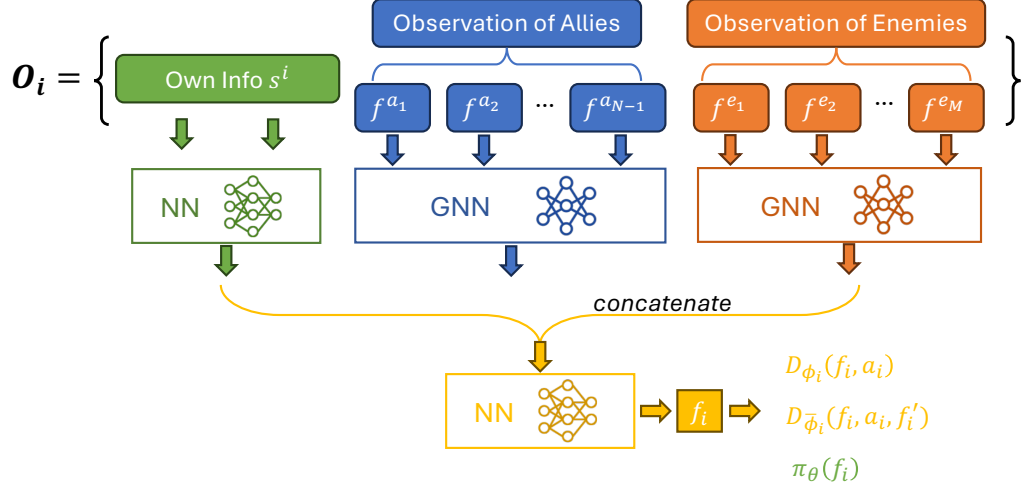


Figure 4.1: The network structure of DynPegMARL. Suppose there are N agents and M enemies in the team.

shown in Figure 4.1, the observations of each agent i is encoded as:

$$h_i^{\text{self}} = \text{NN}_{\text{self}}(s^i), \quad h_i^{\text{ally}} = \text{GNN}_{\text{ally}}(\{f^j\}_{j \in \mathcal{E}_i^{\text{ally}}}), \quad h_i^{\text{enemy}} = \text{GNN}_{\text{enemy}}(\{f^k\}_{k \in \mathcal{E}_i^{\text{enemy}}}), \quad (4.1)$$

and the final encoded observation is formed by:

$$\hat{o}_i = \text{Concat}(h_i^{\text{self}}, h_i^{\text{ally}}, h_i^{\text{enemy}}). \quad (4.2)$$

This design allows the same network to process inputs from environments with different numbers of agents. The use of separate GNNs for ally and enemy sets ensures that the encoder can reason differently about cooperative and competitive interactions, while still maintaining a fixed-size output for downstream components.

We apply the GNN encoder to both discriminators, which operate on GNN-encoded observations, enabling consistent training and inference across scenarios with different team configurations. Optionally, the policy network $\pi_\theta(a_i|o_i)$ can also be replaced with a GNN-based architec-

ture to further enhance generalization across different team configurations, although the method remains effective even with a standard MLP policy. During training, we can sample demonstrations from multiple team sizes in the same batch. Each observation is processed through the GNN-based encoder to obtain consistent embeddings, enabling unified policy and discriminator updates. The overall learning procedure follows PegMARL’s original reward-shaping and policy optimization framework, now augmented with the flexibility to handle demonstration data from diverse team configurations.

4.3 Experiments

We evaluate our approach in the SMACv2 [1] environment using three Protoss combat scenarios with different team sizes: 5v5, 10v10, and 20v20. For each setting, we collect demonstrations by training agents independently under that specific configuration. These trajectories reflect coordinated multi-agent behaviors tailored to each team size.

During training, we use the combined set of demonstrations from all three team sizes, and train separate policies under each target configuration (5v5, 10v10, or 20v20). This setup allows us to assess whether incorporating demonstrations from different team sizes can accelerate learning and improve performance, even when the demonstration and deployment configurations do not match exactly.

4.3.1 Main Results

We compare the following baselines: (1) MAPPO: A standard multi-agent policy gradient method with MLP-based observation encoders; (2) MAPPO-GNN: A variant of MAPPO where

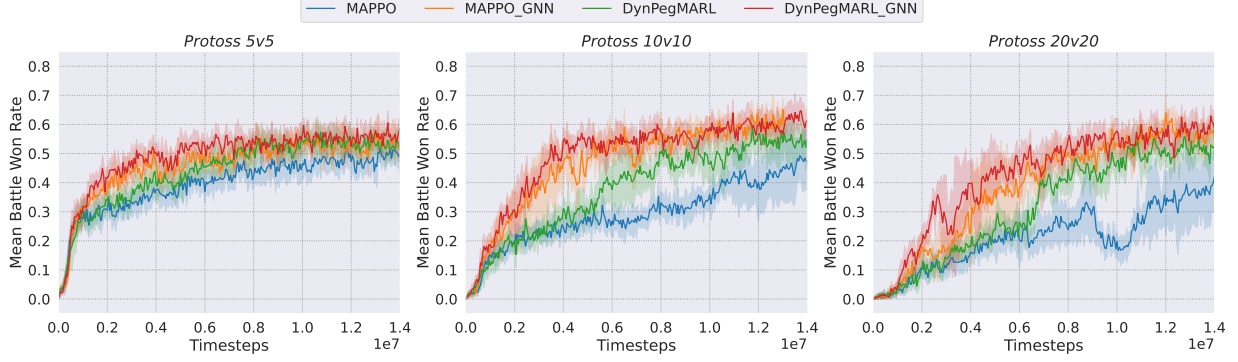


Figure 4.2: Training performance of MAPPO, MAPPO-GNN, DynPegMARL, and DynPegMARL-GNN across three Protoss scenarios in SMACv2 with different team sizes: 5v5, 10v10, and 20v20. DynPegMARL-GNN consistently achieves faster convergence and higher final performance by incorporating demonstration data and using GNN-based policy networks. Curves show the mean and standard deviation over 5 runs.

the MLP-based policy network is replaced with the GNN encoder introduced in our method, but without using any demonstration data; (3) DynPegMARL: Our method, which uses GNN-based observation encoding for both discriminators and leverages demonstrations through reward shaping; (4) DynPegMARL-GNN, which uses GNN-based observation encoding for the policy network as well.

We report training curves under each team size setting and compare convergence speed and final performance across methods in Figure 4.2. In all cases, DynPegMARL-GNN demonstrates clear improvements over the baselines. Compared to MAPPO, the addition of GNN encoding alone (MAPPO-GNN) brings moderate performance gains, particularly in larger team settings. This suggests that GNN-based representations help capture local agent relationships more effectively than flat MLP encodings. The full PegMARL-GNN method further improves both learning speed and final win rate. The benefits are most pronounced in the 20v20 setting, where the complexity of coordination increases substantially. Here, demonstration-guided reward shaping significantly accelerates early learning and leads to better final performance. These results high-

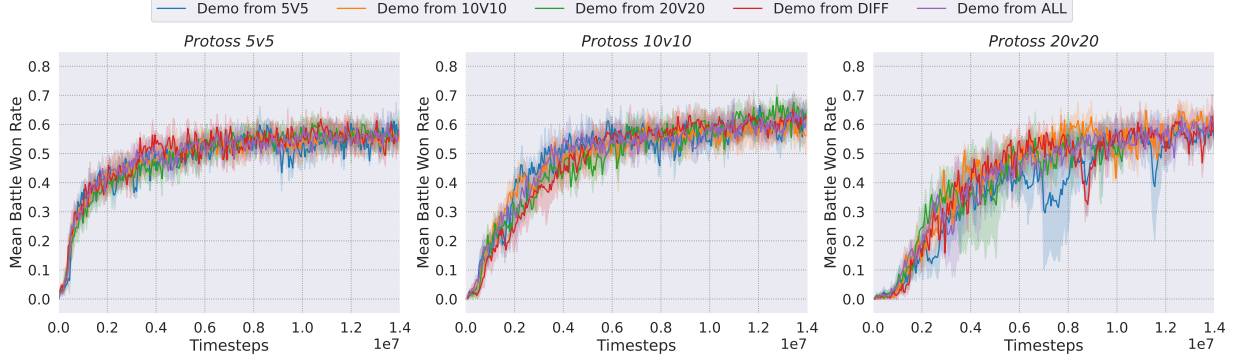


Figure 4.3: Training performances of DynPegMARL-GNN across different demonstration sources.

light the effectiveness of our method in leveraging demonstrations from multiple team sizes to improve multi-agent training efficiency across different coordination scales.

4.3.2 Ablation Study

We further conduct ablation studies to evaluate the robustness of DynPegMARL to variations in demonstration sources and quality.

Demonstration Sources. To assess whether DynPegMARL benefits equally from demonstrations collected under different team sizes, we train the model under each Protoss environment configuration (5v5, 10v10, 20v20) with three types of demonstration sources: (1) **5v5 / 10v10 / 20v20**: Demonstrations collected under the specified team size. (2) **DIFF**: Demonstrations collected from team sizes different from the training one. For instance, in the 5v5 setting, we use demonstrations from 10v10 and 20v20; in 10v10, we use 5v5 and 20v20; and in 20v20, we use 5v5 and 10v10. (3) **ALL**: Demonstrations from all three team sizes combined. As shown in Figure 4.3, DynPegMARL achieves comparable performance across all variants, indicating that it effectively learns from demonstrations regardless of whether they originate from the same team size as the target environment or not. This supports our architectural design choice of Dyn-

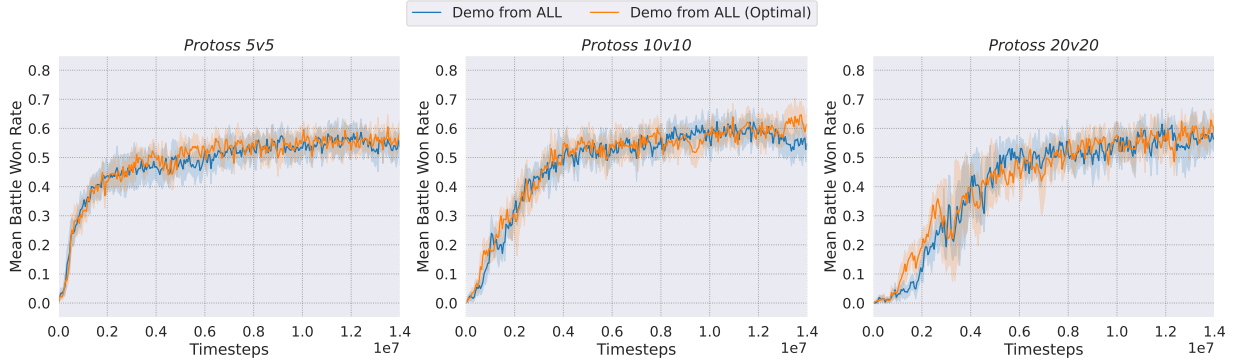


Figure 4.4: Training performances of DynPegMARL-GNN across different demonstration qualities.

PegMARL, which enables general-purpose reuse of demonstrations through size-invariant GNN encoding.

Demonstration Quality. We also examine the effect of demonstration quality on the performance of DynPegMARL. In Figure 4.4, we compare learning curves obtained with: (1) **optimal demonstrations** that contain only successful runs, and (2) **suboptimal demonstrations** that were collected from suboptimal policies with significantly lower win rates. It can be seen that DynPegMARL performs similarly under both settings, indicating that the framework does not rely on high-quality demonstrations for effective learning. This is consistent with PegMARL’s original insight that demonstrations primarily serve to guide exploration, and that agents can still learn to coordinate via environment feedback even when the demonstration behavior is imperfect.

4.4 Conclusion

In this chapter, we presented DynPegMARL, an extension of PegMARL designed to leverage demonstrations from various team sizes in multi-agent reinforcement learning. By introducing a GNN-based observation encoder, our approach handles variable-length, unordered entity

features in a size-invariant manner, enabling consistent training across different team configurations. Experimental results in SMACv2 demonstrate that DynPegMARL improves sample efficiency over strong baselines and can effectively utilize demonstrations collected from both matching and mismatched team sizes. Ablation studies further confirm the robustness of the method to demonstration source and quality, highlighting its flexibility and practicality in real-world multi-agent scenarios.

Chapter 5: Generalization across Sight Range through Task-Agnostic Contrastive Pretraining

5.1 Introduction

Multi-agent Reinforcement Learning (MARL) provides a framework for addressing complex coordination tasks across various domains such as robotics [18, 15, 77], autonomous vehicles [78, 16], and network optimization [79, 17]. In MARL, agents often operate under partial observability, where each agent’s perception is limited to a certain “sight range” around itself, which results in a fundamental challenge known as the *sight range dilemma* [80]. The dilemma lies in balancing tension between an agent’s need for local information to make decisions and the broader context required for effective team coordination. Agents with narrow sight ranges often struggle to coordinate effectively due to limited environmental information, while those with ex-

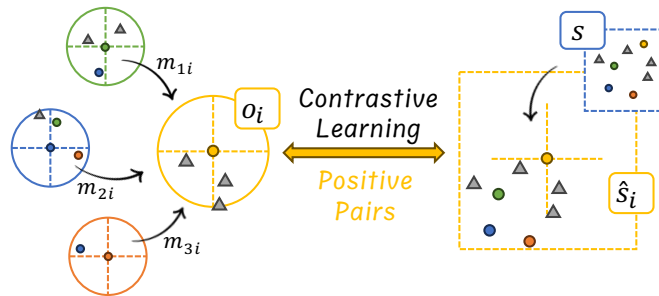


Figure 5.1: TACTIC utilizes contrastive learning to align the integration of local observations o_i and messages $\{m_{ji}\}$ with the full egocentric state $\hat{s}_i$ for each agent i , enabling agents to “see” beyond their limited sight ranges through communication.

tensive sight ranges can become overwhelmed by excessive data, leading to inefficient learning and reduced performance.

Researchers have proposed various approaches to addressing this challenge, primarily focusing on communication mechanisms that allow agents to share information. Those methods include targeted communication strategies [81, 82, 83], attention mechanisms [84, 85, 75], or graph-based methods [86, 87, 88]. For example, QMIX-Att [89] integrates attention into the QMIX framework for selective communication, and TarMAC [81] uses signature-message pairs for context-aware communication. While effective, these methods assume the same sight ranges during training and during execution, limiting their capacity to adapt to varying visibility conditions, as illustrated in Figure 5.2.

Generalizing across varying sight ranges provides considerable benefits for MARL systems. It facilitates more efficient and cost-effective deployments by enabling a single, adaptable model to handle diverse observability conditions, eliminating the need for separate models for each scenario. This flexibility is crucial in real-world applications, where systems must adjust to different visibility conditions. For example, autonomous vehicles need to adapt to visibility fluctuations due to weather or time of day, while search and rescue robots may encounter visual obstructions from debris or smoke. In this work, we are particularly interested in the cases where the sight range during execution is fixed but different from that seen during training.

We hypothesize that *agents can better generalize across different sight ranges if they can communicate in a way that leads to a more comprehensive understanding of the global environment*. Based on this hypothesis, we propose a novel approach that aligns the integrated local observations and messages with each agent’s egocentric (global) state, as illustrated in Figure 5.1. The egocentric state serves as an ideal alignment target, providing a comprehensive yet

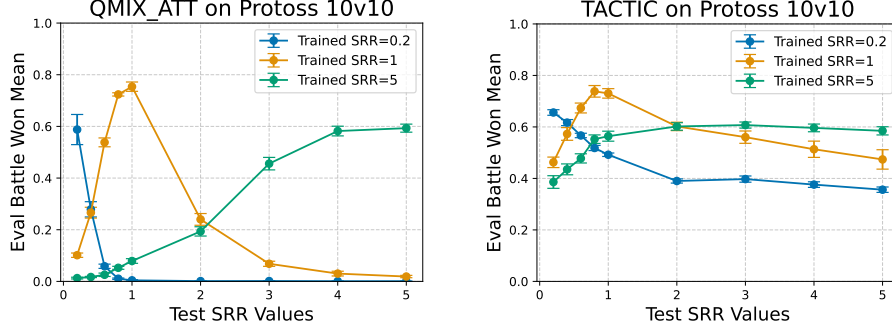


Figure 5.2: QMIX-ATT and TACTIC’s performances on Protoss 10v10 from SMACv2 [1] with varying sight ranges (SRs). Different SRs are achieved by applying different sight-range ratios (SRRs) to the agents’ original SRs in the implementation. Policies trained at SRR=0.2, 1, and 5 are tested across a broader set of SRRs. QMIX-ATT Struggles to generalize to unseen SRs, while TACTIC generalizes much better.

agent-specific view of the environment during training. We use contrastive learning to achieve this alignment, encouraging agents to develop a communication protocol that bridges the gap between limited local observations and the broader environmental context. This process enables agents to effectively “see” more through communication, regardless of their actual sight ranges.

In addition to the contrastive learning objective, we introduce two auxiliary losses: a reconstruction loss and a dynamics loss. The reconstruction loss helps ensure that the learned representations retain essential information from the original observations, while the dynamics loss encourages the model to capture the temporal relationships in the environment. Crucially, our method is task-agnostic in nature, as it does not rely on task-specific reward information when learning to communicate, solely focusing on capturing the underlying environment information, which further enhances the flexibility and adaptability of our method across diverse scenarios.

To this end, we introduce **TACTIC** (**T**ask-**A**gnostic **C**ontrastive pre-**T**raining for **I**nter-Agent **C**ommunication), a novel strategy designed to enhance generalization across varying sight ranges in cooperative MARL. TACTIC operates through two key stages: (1) **Offline contrastive pretraining**, where we use contrastive learning on an offline dataset to pretrain two key com-

munication modules: a message generator and a message-observation integrator. (2) **Online policy integration**, where the pre-trained communication modules are frozen and incorporated into agents’ online policy learning, enabling dynamic communication adaptation during task execution while preserving the learned task-agnostic properties.

We summarize the main contributions of this work as follows:

- A task-agnostic communication mechanism that enables adaptive message generation and interpretation;
- A cooperative MARL framework with communication called TACTIC that alleviates the sight range dilemma;
- A comprehensive evaluation of TACTIC in the SMACv2 environment showing TACTIC’s superior performance regarding generalizability across sight ranges and training efficiency.

Our experimental results on the SMACv2 (StarCraft Multi-Agent Challenge) benchmark show that TACTIC outperforms existing state-of-the-art MARL with communication techniques. Our method demonstrates robust generalization capabilities, enabling effective coordination across various sight ranges (Figure 5.2).

5.2 Related Work

Communication in MARL. In cooperative MARL, communication is a critical component for addressing partial observability and improving agent coordination [90, 91]. Learning effective communication in this context involves several challenges, including determining *who* communicates, *how* messages are conveyed, and *what* information is transmitted under band-

width or sight-range constraints.

Several approaches have been proposed to address these challenges. Targeted communication methods focus on identifying specific agents for message exchange, while graph-based and attention-based models structure communication based on relational dynamics between agents. Graph-based methods, such as the graph-attention network proposed by Niu et al. [84], enable agents to dynamically adjust communication based on relevance, improving scalability in complex environments. Techniques that manage bandwidth limitations address the need to optimize when and what information should be shared. For example, attention-based methods such as TarMAC [81] leverage signature-message pairs and attention mechanisms to enable dynamic, context-aware communication between agents. Information-theoretic approaches, such as NDQ [92], introduce regularization to minimize communication overhead while maximizing the informativeness of messages.

Recent advancements also aim to make multi-agent communication more interpretable and flexible. Lin et al. [93] proposed grounding communication by autoencoding raw observations into messages, allowing agents to develop a shared understanding of communication symbols. Similarly, MASIA [94] aggregates raw observations into latent representations that can be used to reconstruct the global state, providing agents with a more holistic view of the environment. Du et al. [95] introduced methods to learn correlated communication topologies, which reduce redundancy and optimize coordination among agents by refining communication pathways. In parallel, Zhang et al. [96] introduced Temporal Message Control (TMC), a technique that applies temporal smoothing to reduce the number of inter-agent messages, achieving robust and efficient communication in resource-constrained environments without sacrificing performance.

Our work builds on these advancements by addressing adaptive communication under

sight-range limitations, proposing a novel approach that optimizes communication frequency and content based on changing agent observations. This differs from previous works by integrating bandwidth and perceptual constraints into a unified framework, enabling efficient communication without reliance on pre-defined structures.

Contrastive Learning in MARL. Contrastive learning is a representation learning technique that aims to bring similar (positive) samples closer together in the learned feature space while pushing dissimilar (negative) samples farther apart. This is typically achieved using a contrastive loss function. Methods such as Contrastive Predictive Coding (CPC) by Oord et al. [97] laid the groundwork for learning predictive representations by contrasting positive and negative samples. Building on this, TACO [98] adapts contrastive learning to RL by learning useful representations through temporal abstraction. More recently, the use of supervised contrastive loss [99], which incorporates multiple positive and negative samples per anchor point, has been explored. This extension enables richer representation learning by capturing a broader set of relevant relationships, which is particularly valuable in RL tasks with multiple favorable outcomes.

In the multi-agent domain, contrastive learning has seen adaptation for both non-communicative and communicative settings. For MARL without communication, methods like COLA [100] have demonstrated the ability to improve coordination among agents by using contrastive objectives to refine agent policies based on shared goals. This approach emphasizes the utility of contrastive learning in situations where direct agent-to-agent communication is absent or limited. In contrast, for settings where agent communication is possible, contrastive learning has been utilized to optimize communication strategies between agents. Lo et al. [91] apply contrastive learning to improve multi-agent communication, enabling agents to develop more efficient communication protocols that reduce unnecessary message exchanges while maintaining performance.

Additionally, methods such as the one proposed by Singh et al. [101] focus on learning when to communicate, which is crucial in reducing communication overhead in resource-constrained environments. Zhang et al. [102] further refine this by incorporating variance-based control mechanisms, allowing agents to communicate only when necessary, improving overall system efficiency.

Our work extends these ideas by integrating contrastive learning into multi-agent communication under sight range limitations. This approach differs from previous work in that we train the communication module offline specifically with the goal of generalizing across sight ranges.

5.3 Preliminaries

Dec-POMDP with Communication. We consider the fully cooperative MARL problem with communication, which can be modeled as Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [103] and formulated as a tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, P, \Omega, O, R, \gamma, \mathcal{C} \rangle$. The sets $\mathcal{N} = \{1, \dots, n\}$ denotes the indexing of the agents, $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, Ω is observation space, and $\mathcal{C}$ denotes all possible communication messages. Each agent $i \in \mathcal{N}$ acquires an observation $o_i = O(s, i) \in \Omega$, where O is the observation function and $s \in \mathcal{S}$. A joint action $\mathbf{a} = \langle a_1, \dots, a_n \rangle$ leads to the next state $s' \sim P(s'|s, \mathbf{a})$ and a shared global reward $r = R(s, \mathbf{a})$ where R is the reward function.

Each agent selects actions based on the observation-action history $\tau_i \in \mathcal{T} \equiv (\Omega \times \mathcal{A})^*$ ¹ using a policy $\pi(a_i|\tau_i, m_i)$ where $m_i = [m_{ji} \in \mathcal{C}, j \in \mathcal{N}]$ denotes the incoming messages for agent i and m_{ji} is the message sent from agent j to agent i . The policy is shared across agents during training.

¹* denotes the product over time

The overall objective is to find a joint policy $\pi(\boldsymbol{\tau}, \mathbf{a})$ to maximize the global value function

$$Q^\pi(\boldsymbol{\tau}, \mathbf{a}) = \mathbb{E}_{s, \mathbf{a}} \left[\sum_{t=0}^{\infty} \gamma^t R(s, \mathbf{a}) \mid s_0 = s, \mathbf{a}_0 = \mathbf{a}, \pi \right], \quad (5.1)$$

where $\boldsymbol{\tau}$ is the joint observation-action history of all agents and $\gamma \in [0, 1)$ is the discount factor.

We follow the *Centralized Training and Decentralized Execution* (CTDE) paradigm and adopt the architecture of QMIX [104] to form our algorithm ².

Contrastive Learning. Contrastive learning is a powerful paradigm in representation learning, particularly in the context of deep learning, where it aims to learn embeddings by contrasting positive and negative samples. The fundamental idea is to pull together representations of similar instances (positives) while pushing apart those of dissimilar instances (negatives). In the supervised setting, the SupCon (Supervised Contrastive) loss [99] extends this framework by allowing for multiple positive samples per anchor, thereby leveraging label information more effectively. The loss objective can be mathematically expressed as:

$$L_{supcon} = \sum_i \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \left(\frac{\exp(z_i \cdot z_p / v)}{\sum_{a \in A(i)} \exp(z_i \cdot z_a / v)} \right) \quad (5.2)$$

where z_i denotes the normalized embedding of the anchor sample, $P(i)$ is the set of positive samples corresponding to the anchor, $A(i)$ is the set of all samples in the batch excluding the anchor, and v is a temperature parameter that controls the sharpness of the distribution. This formulation enhances the clustering of similar instances in the embedding space and improves robustness against natural corruptions.

²For clarity, we drop the time superscripts for states and actions

5.4 TACTIC: Towards task-agnostic adaptive communication in MARL

In this section, we introduce our algorithm TACTIC. TACTIC consists of two stages: (1) an offline training of a communication mechanism that works well with varied sight ranges, and (2) an online training of the agent coordination policy. The task-agnostic offline training stage solely utilizes the environment states and local observations of agents and doesn't rely on environmental reward signals or any policy. The online policy training stage learns a task-specific policy with the communication module from the first stage frozen. We explain the offline stage in Section 5.4.1 and the online stage in Section 5.4.2 in detail.

5.4.1 Offline Training of Communication Mechanism

One main challenge in Multi-Agent Reinforcement Learning (MARL) is to develop effective joint policies when each agent has only a partial observation of the environment. By learning to communicate more effectively, agents are expected to overcome their limited observability and achieve better coordination. Furthermore, the mechanism used to generate and use messages between agents should be flexible enough to handle different sight ranges.

To this end, we present an approach where, in an offline training stage, we use contrastive learning on a pre-collected dataset to develop a communication mechanism that can adapt to different observation ranges. The rationale is that, by learning to create and integrate messages that capture important information about the whole environment, agents can effectively **“see” more through communication**, regardless of their current sight range.

The offline dataset $\mathcal{D}$ consists of a set of trajectories, where each trajectory $\tau = \{(s^t, o_{1:n}^t, a_{1:n}^t)\}_{t=1}^T$ represents a sequence of T timesteps for n agents. The dataset can be collected through random

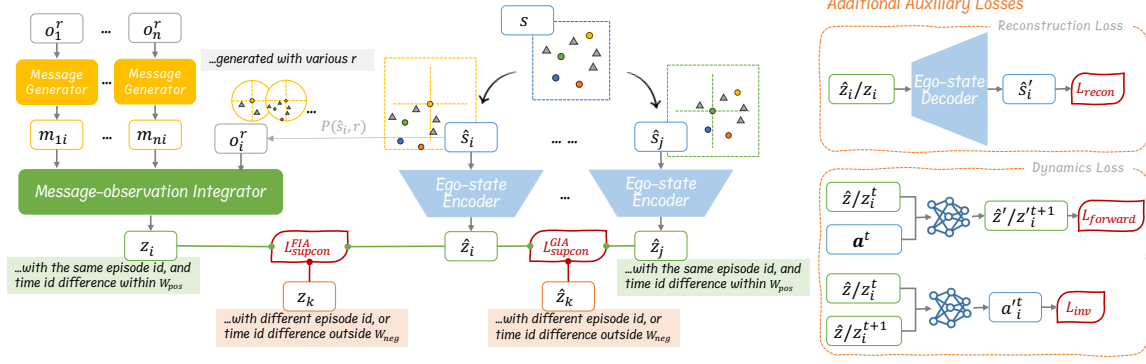


Figure 5.3: The offline training pipeline of the adaptive communication mechanism(Section 5.4.1). It includes three key components: an egocentric state encoder, an adaptive message generator, and a message-observation integrator. The training pipeline consists of two contrastive learning processes: Global Information Alignment (GIA) for aligning the features generated from the egocentric state encoder across all agents and timesteps, and Feature Integration Alignment (FIA) for aligning features from the message-observation integrator and the egocentric state encoder on an individual agent. Two auxiliary loss functions are introduced in the total loss function to enhance training: a deconstruction loss for learning to recover the ego-centric states and a dynamic loss for learning temporally coherent representations.

exploration of the environment (details in Section 5.5.3).

The overall offline training pipeline for the adaptive communication mechanism is illustrated in Figure 5.3, where three key components are present: an egocentric state encoder, an adaptive message generator, and a message-observation integrator:

1. The *egocentric state encoder* takes an egocentric state $\hat{s}_i$ and generates its corresponding feature embedding $\hat{z}_i$, where the $\hat{s}_i$ are obtained from the global state s , which preserves all information from s but represents from the perspective of agent i .
2. The *adaptive message generator* takes the partial observation o_i^r of an agent with *varying* sight ranges and outputs the message $\{m_{ij}\}_{j=1}^n$ it communicates to other agents. We obtain o_i^r by randomly sampling a sight range r and applying a masking operation over the egocentric state $\hat{s}_i$, denoted as $P(\hat{s}_i, r)$.

3. The *message-observation integrator* takes an agent’s partial observation o_i^r and all the messages $\{m_{ji}\}_{i=1}^n$ it receives from other agents, integrating them into a feature embedding z_i .

With the three components at hand, the offline training consists of two contrastive learning processes:

1. *Global Information Alignment (GIA)*: a supervised contrastive (SupCon) loss for aligning the feature embeddings $\hat{z}_i$ generated from the egocentric state encoder across all agents, ensuring that the egocentric state encoder captures consistent and relevant information from agents about the whole environment;
2. *Feature Integration Alignment (FIA)*: a supervised contrastive (SupCon) loss for aligning the integrated feature z_i calculated by the message-observation integrator with the generated feature $\hat{z}_i$ from the egocentric state encoder for each specific agent, pushing the adaptive message generator to learn to synthesize the most informative messages to communicate and allowing the message-observation integrator to reflect a more complete picture of the environment (i.e., the agent’s egocentric state) given the agent’s limited observation and the messages they receive.

In FIA, the function $P(\hat{s}_i, r)$ applies augmentation by randomly varying the sight range r (resulting in o_i^r). This process exposes the adaptive message generator to diverse scenarios with changing sight ranges, rather than fixed ones, thereby enhancing its generalizability. For contrastive learning in both GIA and FIA, we define positive and negative pairs based on the offline dataset $\mathcal{D}$. Features from agents within the same episode (i.e., from the same trajectory) and within a timestep window of length W_{pos} form positive pairs. Conversely, features from agents in

different episodes or separated by more than W_{neg} timesteps constitute negative pairs. Note that both GIA and FIA are task-agnostic as they don't interact with any environmental reward signals.

To further improve the learned representations, we incorporate two additional auxiliary learning objectives: (1) a reconstruction loss (L_{recon}): a decoder network learns to recover the egocentric state $\hat{s}_i$ from the feature embeddings produced by either the egocentric state encoder or the message-observation integrator; and (2) a dynamic loss (L_{dyn}): this includes both forward and inverse dynamics predictions using MLP networks. The forward model predicts $\hat{z}_i^{t+1}$ (or z_i^{t+1}) given $\hat{z}_i^t$ (or z_i^t) and $\mathbf{a}^t$, while the inverse model predicts a_i^t given consecutive feature embeddings $(\hat{z}_i^t, \hat{z}_i^{t+1})$ or (z_i^t, z_i^{t+1}) . These auxiliary objectives promote comprehensive, temporally coherent representations that better support adaptive communication in multi-agent scenarios.

The final loss for the overall offline training is a weighted sum of the contrastive losses and the auxiliary losses:

$$L_{total} = L_{supcon}^{GIA} + L_{supcon}^{FIA} + \alpha L_{recon} + \beta L_{dyn} \quad (5.3)$$

where α and β are weighting factors.

5.4.2 Online Training of Agent Policy

After offline pretraining, we integrate the *adaptive message generator* and *message-observation integrator* into the QMIX [104] framework for online policy training (illustrated in Figure 5.4). In this stage, each agent i processes its observation and receives messages (synthesized with the *adaptive message generator* from other agents) to generate an integrated representation z_i using the *message-observation integrator*. This representation z_i , together with the agent's last action, informs the agent's action selection via a GRU-based Q-network. The QMIX architecture then

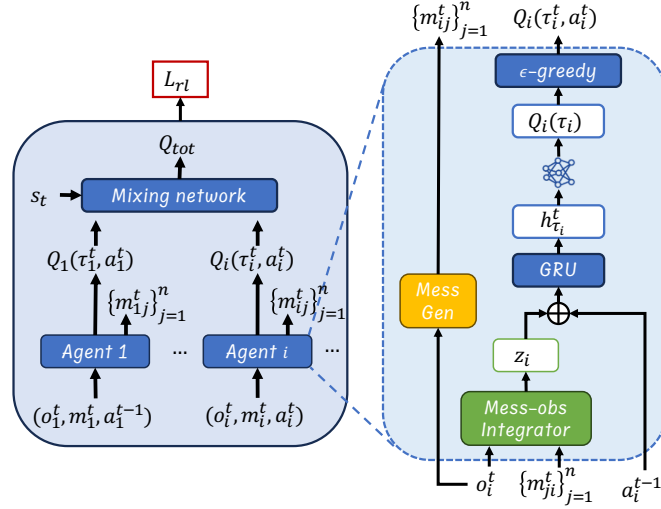


Figure 5.4: Online policy training pipeline of TACTIC, illustrating the integration of QMIX architecture with pre-trained communication components. The pre-trained message generator (Mess Gen) and message-observation integrator (Mess-obs Integrator) remain fixed during the policy training.

combines individual agents' Q-values through a mixing network conditioned on the global state to produce a centralized Q-value, which is used to compute the loss for training agent policies and the mixing network. Importantly, the parameters of the pre-trained *adaptive message generator* and *message-observation integrator* remain fixed during this online training phase and are not updated.

In the remaining sections, we present the experimental setup and results that evaluate the effectiveness of our methodology.

5.5 Experiments

In this section, we report our evaluation of TACTIC. Our experiments aim to address the following key questions:

Q1. Can the policy trained with the adaptive communication mechanism generalize across dif-

ferent sight ranges?

Q2. Does the offline-trained communication mechanism enhance the efficiency of online policy training?

Q3. How do data quality and varying loss terms impact the performance of training effectiveness and generalization?

5.5.1 Experimental Setup

Our experiments are conducted in the SMACv2 environment [1]. Compared to the original SMAC environment [65], SMACv2 incorporates increased stochasticity and meaningful partial observability, necessitating the development of complex closed-loop policies for effective agent coordination. We use three maps from SMACv2 for our experiments: Terran, Protoss, and Zerg. Each map features distinct unit types, with Terran units including Marines, Marauders, and Medivacs; Protoss units comprising Stalkers, Zealots, and Colossi; and Zerg units consisting of Zerglings, Hydralisks, and Banelings. Agents are generated procedurally, with varying numbers ranging from 5 to 20, and are assigned specific sight and attack ranges that enhance the complexity of the scenarios.

For each map, we consider three agent number configurations (5, 10, and 20 agents) and three sight-range ratios (0.2, 1, and 5). The sight-range ratios are applied by multiplying the agents' original sight ranges, which vary among different agents, to adjust their visibility accordingly. This allows us to assess the adaptability of the proposed communication mechanism under varying observability conditions.

For each environment setup specified by a combination of the map and the agent number

configuration, we pre-train an offline communication mechanism (Section 5.4.1). This communication mechanism is then used for online policy training (Section 5.4.2) with a team of agents with fixed sight ranges. The learned policy is further evaluated on a variety of sight ranges (the sight range ratios are between 0.2 and 5) on the same environment setup.

To valid the effectiveness of TACTIC, the performance regarding the generalization across various sight ranges and the online training efficiency is compared to four baselines: QMIX [104], QMIX-Att [89], NDQ [92], and TarMAC [81]. QMIX is one of the more commonly used CTDE MARL algorithms that does not use communication during execution. The others are three variants of QMIX with different communication mechanisms. We adopted the hyperparameters from the original implementations of the baselines and reused the hyperparameters of QMIX for TACTIC. All results in the following sections are from five independent runs with different random seeds.

5.5.2 Policy Generalization Across Sight Ranges

To answer Q1, in this section, we report the generalization capability of our trained policies across various sight ranges. As introduced in Section 5.5.1, our policies are learned based on a pre-trained communication mechanism under the same environment setup with fixed sight ranges; they are then tested on a broader set of sight-range ratios valued from 0.2 to 5. Their performances (i.e., *mean battle won rate*) in all scenarios are recorded and visualized with heatmaps as shown in Figure 5.5, 5.6 and 5.7.

It can be clearly seen that our method TACTIC produces policies that demonstrate more robust performances in all the environment settings and across various sight-range ratios that

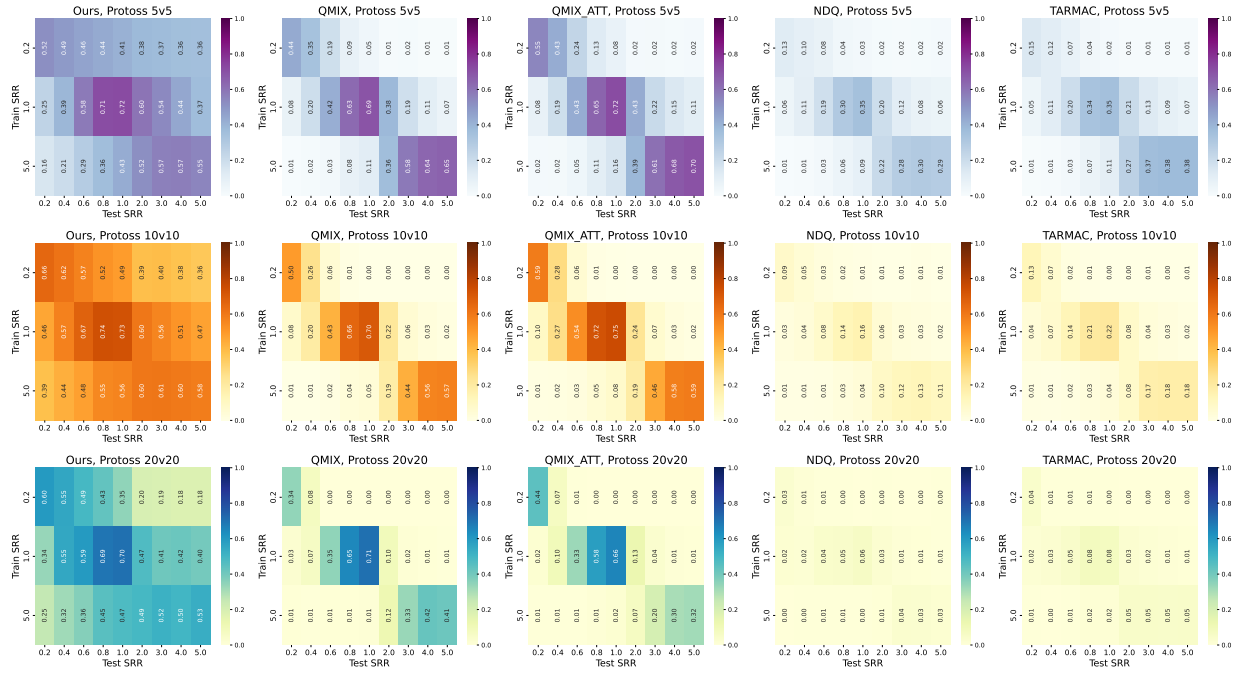


Figure 5.5: Performance of TACTIC and baseline models on policy generalizability across various sight ranges in the Protoss map.

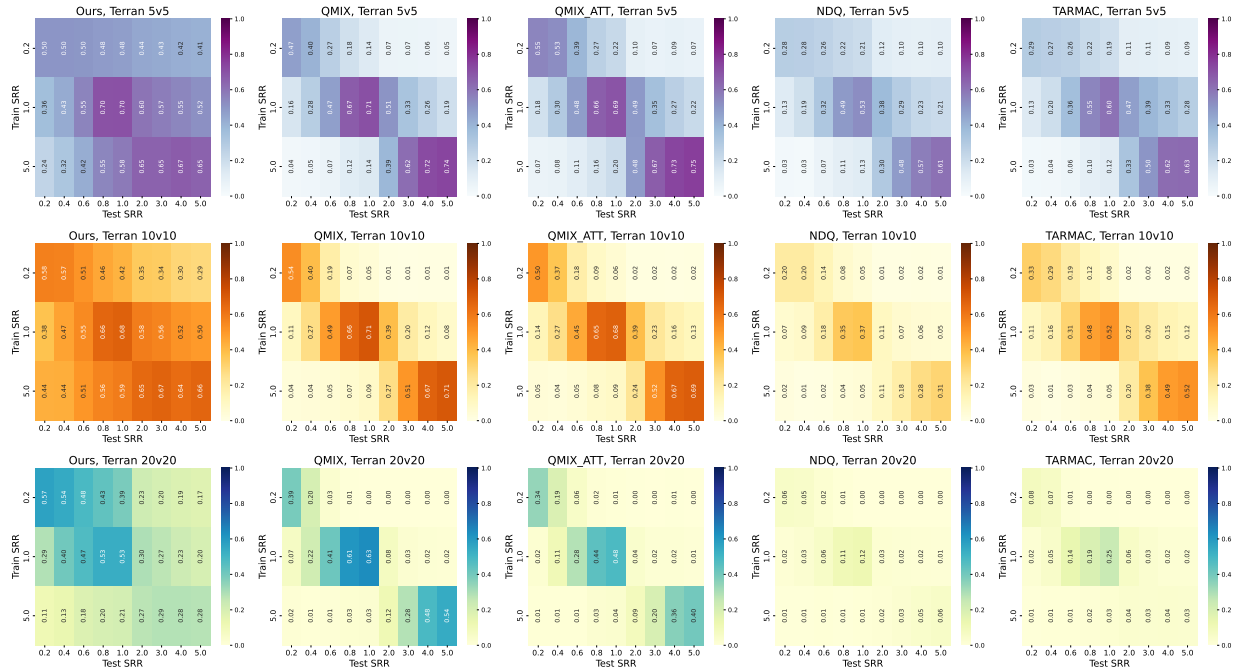


Figure 5.6: Performance of TACTIC and baseline models on policy generalizability across various sight ranges in the Terran map.

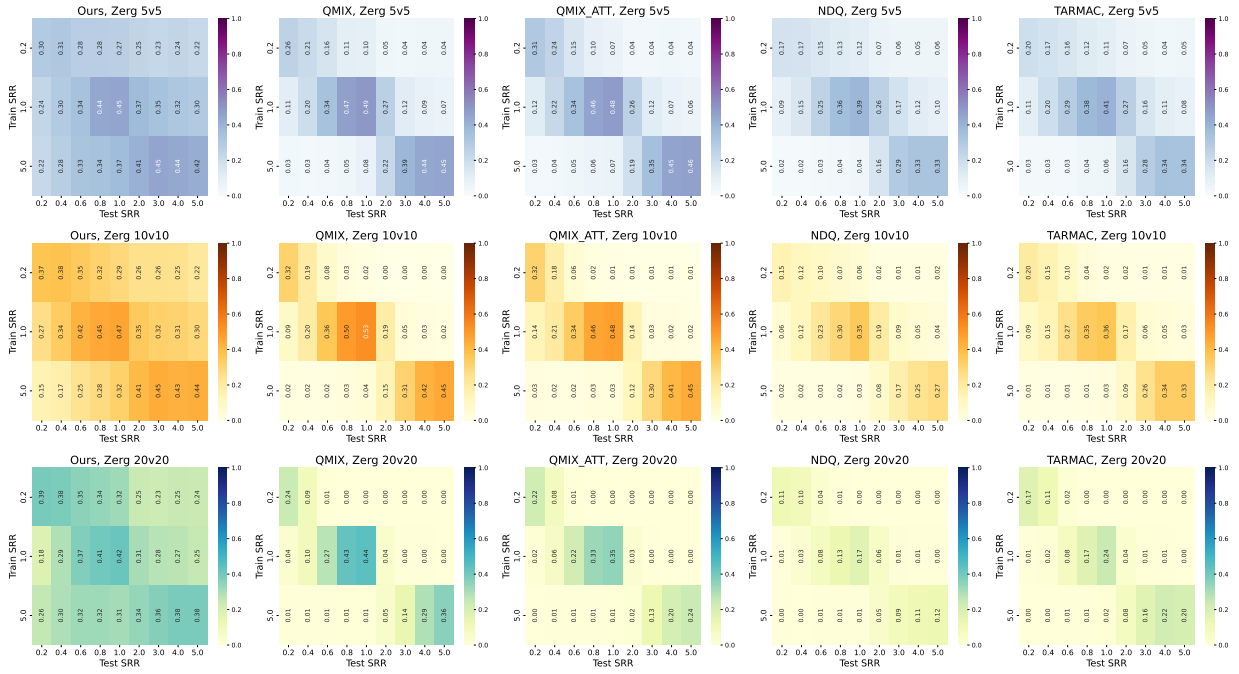


Figure 5.7: Performance of TACTIC and baseline models on policy generalizability across various sight ranges in the Zerg map.

were not seen during policy training. The four baselines can have reasonable performances only when the test and train sight ranges are close (referring to the diagonal cells of the heatmaps for the baselines), while policies learned by TACTIC are capable of maintaining satisfying performances even when the train and test sight ranges are different by large. For example, on the map Terran with 5 agents, policy trained with sight-range ratio 0.2 by TACTIC has a 0.41 *mean battle won rate* when tested with sight-range ratio 5, while the corresponding results for the four baselines are all under 0.10; it is also true if exchanging the train and test sight-range ratios or in other maps. With more agents introduced into the environment, baseline models become more strict on the difference between train and test sight ranges (i.e., their heatmaps become more sparse with more zero values), while our method TACTIC still generalizes well.

The observed phenomenon suggests that our adaptive communication mechanism that is learned offline helps agents better generate and interpret messages and develop flexible strategies

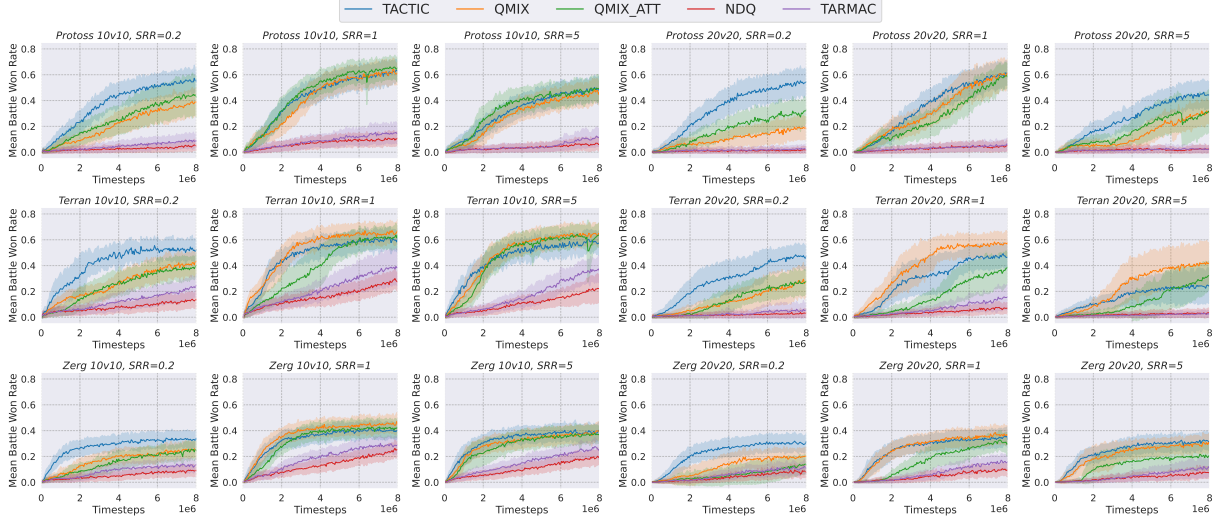


Figure 5.8: Training curves of the online policy learning stage in TACTIC and baseline models under different environment setups.

to take actions to adapt to changing observability conditions, leading to **reusable policies** in environments with unseen sight ranges. However, for the four baseline approaches, one needs to *re-train* the model to get a usable policy if the observation conditions change.

5.5.3 Online Policy Training Efficiency

To answer Q2, we compare the online policy training efficiency of our method TACTIC against the four baselines. Figure 5.8 presents the learning curves (*mean battle won rate* versus *timesteps*) of all five algorithms on every combination of map type, number of agents, and the sight-range ratio.

It can be observed that in every environment setup, the convergence speed of TACTIC is superior to or at the same level as that of QMIX or QMIX-Att, and is always better than that of NDQ and TarMAC. Notably, when the sight-range ratio used during online policy training is small (i.e., 0.2), TACTIC demonstrates significantly better convergence speed compared to the baselines.

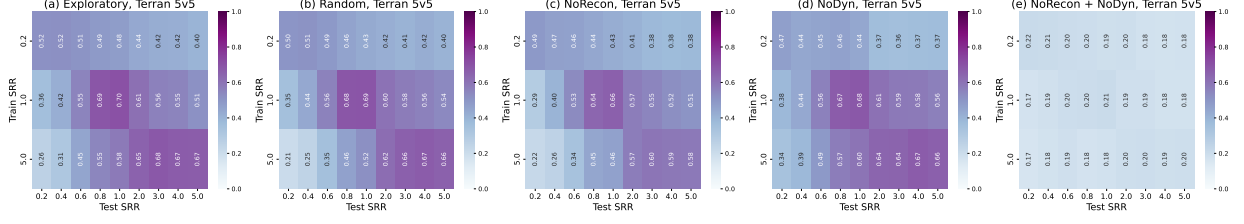


Figure 5.9: Performance of TACTIC on policy generalizability across various sight ranges under 5 different offline training schemas. The environment setup is the Terran map with a 5v5 agent configuration. (a) Offline training with a Exploratory dataset; (b) Offline training with a Random dataset; (c) Offline training without L_{recon} ; (d) Offline training without L_{dyn} ; (e) Offline training without L_{recon} and L_{dyn} .

The results indicate that the offline-trained communication mechanism contributes positively to the training efficiency of online policy learning, especially in scenarios where the number of agents is large or the sight ranges of agents are small.

5.5.4 Ablation Study

In this section, we investigate Q3 by conducting ablation studies on the quality of the offline dataset $\mathcal{D}$ for training the communication mechanism and the impact of different loss terms in our objective function (Eq. 5.3). For this section, we focus on the Terran map and 5-agent setting. **Data quality.** To examine the influence of data quality of $\mathcal{D}$ on the learned online policy, we train the communication mechanism twice on two types of offline datasets.

- **Exploratory:** Trajectories generated in the exploratory stage of QMIX training, with 6000 episodes per task.
- **Random:** Trajectories generated from random environment interactions (where actions are chosen uniformly at random for each agent), with 6000 episodes per task.

We then perform online policy learning twice using the two pre-trained communication

mechanisms, obtaining two policies. They are evaluated in the same way as in Section 5.5.2. The performance is shown in Figure 5.9(a-b). It can be seen that there is a very slight performance decay when you use the RANDOM dataset compared to the EXPLORATORY dataset, indicating that the offline training stage for the communication mechanism can still learn meaningful and informative representations given low-quality data.

Loss terms. We further examine the influence of the two auxiliary loss objectives, the reconstruction loss L_{recon} and the dynamic loss L_{dyn} , by training three communication mechanisms with the weighing factors set to $\alpha = 0, \beta = 1$, $\alpha = 1, \beta = 0$, and $\alpha = 0, \beta = 0$ in Eq. 5.3, respectively. The online policy learning and evaluation remain the same as the data-quality ablation study. The results are shown in Figure 5.9(c-e).

In the first two cases where either the L_{recon} or L_{dyn} is dropped, there are no significant performance decays. Specifically, both terms can help the offline stage learn informative representations individually, while L_{recon} contributes slightly more to the overall performance than L_{dyn} . However, when both terms are dropped, a significant performance decay is observed as shown in Figure 5.9(e). This demonstrates the necessity of adopting at least one of the auxiliary loss objectives, and the combination of the two leads to the most effective representation learning in the offline stage.

5.6 Conclusion

In this work, we introduce TACTIC, a communication mechanism for improving the generalizability of MARL systems. Specifically, we focus on generalizing the policy to scenarios where the sight range of the agents during execution may not be the same as that during training.

The communication mechanism in TACTIC is trained offline and is task-agnostic. Utilizing contrastive loss allows agents to effectively align the integration of local observation and incoming messages with the egocentric state, leading to better situational awareness and improved coordination.

We show that standard benchmark MARL techniques with and without inter-agent communication generalize poorly when the sight range changes. In contrast, we show TACTIC generalizes better across sight ranges. Notably, we find that even random exploration trajectories can be leveraged to learn an effective communication strategy. Our findings suggest that TACTIC provides a robust framework to enhance inter-agent communication, leading towards more adaptive MARL systems.

Chapter 6: Generalization across Initial States through Common Operating Picture Integration

6.1 Introduction

Multi-agent tasks often require collaboration to accomplish shared missions. While Multi-Agent Reinforcement Learning (MARL) algorithms can learn decentralized policies based on local observations, effective information sharing remains crucial for optimal decision-making. This work introduces a novel approach to multi-agent coordination using a unified, grounded representation: the Common Operating Picture (COP) [105].

Our method equips each agent with the ability to integrate its observation history, actions, and received messages into a COP, which serves as a comprehensive representation of the environment. In this work, we consider the state of the underlying DecPOMDP as the COP, encompassing the positions and attributes of all agents. This representation offers two key advantages: it is interpretable by human operators, facilitating potential human involvement, and it improves policy learning by essentially reducing the problem to single-agent RL.

We hypothesize that the COP-based representation enhances situational awareness and leads to improved out-of-distribution (OOD) generalization. However, building a COP in a decentralized fashion presents several challenges. When agents operate in egocentric frames of

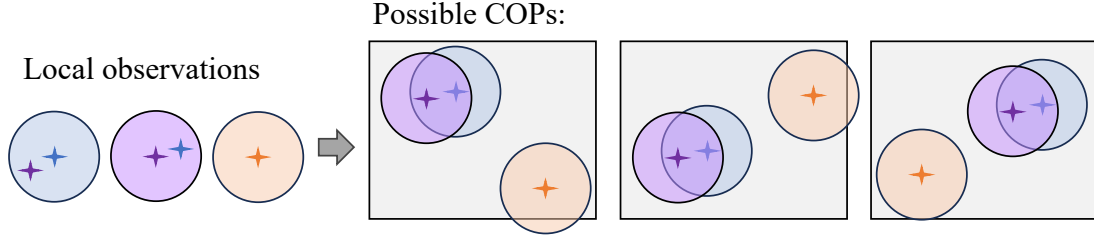


Figure 6.1: The purple and blue agents observe each other, the orange agent is beyond visual range. This leads to multiple potential COPs.

reference, aligning and integrating local observations becomes difficult (Figure 6.1). Limited communication range introduces uncertainty into the COP-building process, potentially hindering the transmission of critical information. Additionally, communicating raw observations is often impractical or unsafe. When agents communicate encoded COPs, the decoding process may introduce errors, leading to potential hallucinations or misinterpretation.

To motivate the benefit of inferring such a COP, we conducted preliminary experiments within the Starcraft2 multi-agent challenge (SMAC) [106]. We tested QMIX [104] and NDQ [92] on initial states different from those used in training (details in Figure 6.11a). The results shown in Figure 6.2 revealed the brittleness of these algorithms in OOD scenarios. However, when QMIX was given access to the global state, its learned policy demonstrated improved robustness. This observation suggests that transmitting messages that help infer the global state could enhance generalization in OOD scenarios. Consequently, we present a MARL method that grounds learned communication in the state, using the COP as a central representation. Our approach aims to address the limitations of existing methods by providing a more robust and interpretable framework for multi-agent coordination.

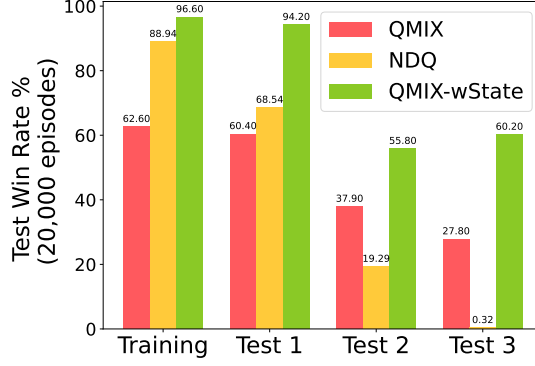


Figure 6.2: Motivating COPs in MARL.

6.2 Related Works

Communication plays a vital role in multi-agent reinforcement learning (MARL) systems [107], enabling agents to coordinate and share information. A central challenge is determining what information should be communicated and how to effectively utilize these messages.

Various approaches have been developed to address this challenge. Attention-based methods, such as TarMAC [81], use signature-message pairs and attention mechanisms for dynamic, context-dependent communication. Information-theoretic approaches like NDQ [92] optimize communication efficiency and effectiveness through regularizers, minimizing overall communication while maximizing message information content.

Recent works focus on developing more interpretable communications. [93] aims to establish a common understanding of communication symbols by autoencoding raw observations as messages, grounding communication in directly observable features. MASIA [94] takes this further by communicating raw observations and aggregating them into a latent representation to reconstruct the global state, providing a more holistic view of the environment.

These approaches represent a progression towards more sophisticated and interpretable rep-

representations in MARL communication. Our work builds on this trend by introducing the Common Operating Picture (COP) as a comprehensive, interpretable representation. The COP aims to enhance both agent coordination and out-of-distribution generalization, addressing limitations in existing methods and potentially improving the robustness of MARL systems in dynamic environments.

6.3 Preliminaries

We consider the fully cooperative MARL problem with communication, which can be modeled as Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [103] and formulated as a tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, P, \Omega, O, R, \gamma, \mathcal{C} \rangle$. The sets $\mathcal{N} = \{1, \dots, n\}$ denotes the agents, $\mathcal{S}$ are the states, $\mathcal{A}$ are the actions, Ω are all observations, and $\mathcal{C}$ are all possible communication messages. Each agent $i \in \mathcal{N}$ acquires an observation $o_i \in \Omega, o_i = O(s, i), s \in \mathcal{S}$. A joint action $\mathbf{a} = \langle a_1, \dots, a_n \rangle$ leads to the next state $s' \sim P(s'|s, \mathbf{a})$ and a shared global reward $r = R(s, \mathbf{a})$.

Each agent selects action based on observation-action history $\tau_i \in \mathcal{T} \equiv (\Omega \times \mathcal{A})^*$ using a policy $\pi(a_i|\tau_i, c_i^{in})$. The policy is shared across agents during training. Incoming communication for agent i , $c_i^{in} = [c_j^{out} \in \mathcal{C} \text{ if } d(i, j) < \rho]$ is determined by the communication range ρ and distance between agents $d(i, j)$. The overall objective is to find a joint policy $\pi(\boldsymbol{\tau}, \mathbf{a})$ to maximize the global value function $Q_{\text{tot}}^{\pi}(\boldsymbol{\tau}, \mathbf{a}) = \mathbb{E}_{s, \mathbf{a}} [\sum_{t=0}^{\infty} \gamma^t R(s, \mathbf{a}) \mid s_0 = s, \mathbf{a}_0 = \mathbf{a}, \boldsymbol{\pi}]$, where $\boldsymbol{\tau}$ is the joint observation-action history and $\gamma \in [0, 1)$ is the discount factor. We follow the *Centralized Training and Decentralized Execution* (CTDE) paradigm and adopt QMIX's [104] architecture to build our algorithm.

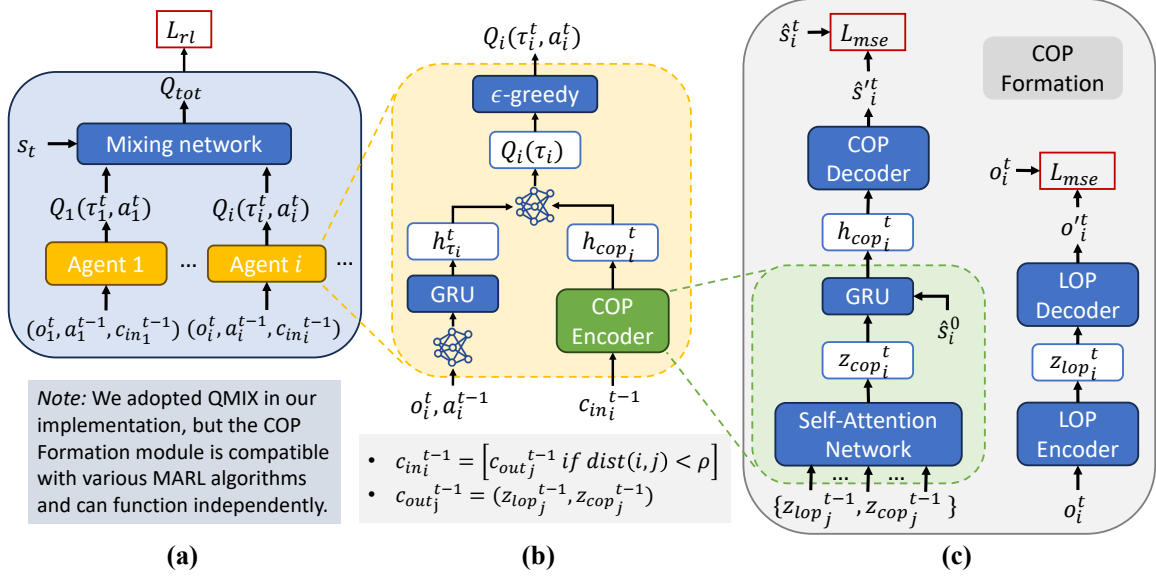


Figure 6.3: The pipeline of our method. (a) The overall architecture. (b) Information integration and policy learning. (c) COP formation module.

6.4 Enhancing Multi-Agent Coordination through COP Integration

Our approach enhances multi-agent coordination by tracking and updating a Common Operating Picture (COP) based on the initial state, observation-action history, and incoming communications. We assume that agents have access to the initial state s_0 . The COP is maintained as a transformation of this initial state, informed by subsequent observations, actions, and communications.

As illustrated in Figure 6.3, the COP Formation module utilizes two parallel threads: the Local Operating Picture (LOP) thread and the Common Operating Picture (COP) thread. In the LOP thread, we encode the local observation o into an embedding z_{lop} , which is then decoded to reconstruct observation o' . The COP thread encodes all received communication messages using a Self-Attention network, producing embedding z_{cop} . This embedding feeds into a GRU, which tracks state evolution with hidden state h_{cop} . The GRU updates h_{cop} using z_{cop} at each step,

integrating new information from other agents. Finally, h_{cop} is decoded to produce the current state estimate $\hat{s}'$, maintaining an up-to-date representation of the environment.

Note that $\hat{s}_i^t$ represents egocentric state centered on agent i at time step t . We use $\hat{s}_i^0$ to initialize the GRU such that it tracks an egocentric representation. The final COP is produced by combining the decoded LOP o' and the estimated state $\hat{s}'$ as follows: $\hat{s}_i^{tt} = \hat{s}_i^{tt} \times \mathbf{1}_{v_i} + o_i^{tt} \times (1 - \mathbf{1}_{v_i})$, where $\mathbf{1}_{v_i}$ is a mask based on the visual range of agent i , as illustrated in Figure 6.4.

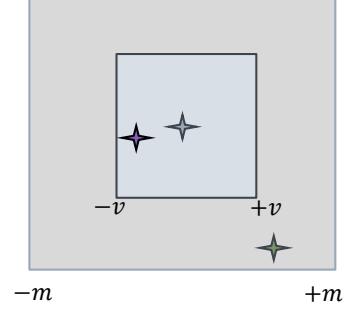


Figure 6.4: Masking in the formation of egocentric COP. v is the visual range of the agent and m is the map size.

We employ a Centralized Training with Decentralized

Execution (CTDE) approach. Our training objective consists of three components:

$$\mathcal{L} = \mathcal{L}_{RL} + \mathcal{L}_{lop} + \mathcal{L}_{cop}.$$

Here, $\mathcal{L}_{RL}$ is the Temporal Difference (TD)-based loss of QMIX, $\mathcal{L}_{lop} = L_{mse}(o', o)$ is the local observation reconstruction loss averaged over agents and time, and $\mathcal{L}_{cop} = L_{mse}(\hat{s}', \hat{s})$ is the egocentric state reconstruction loss averaged over agents and time. It is worth noting that COP formation can also be trained under a fixed policy. By producing an egocentric COP, the agent does not depend on knowledge of its own location in global frame. We produce an egocentric COP for each agent, and by comparing COPs across agents one could quantify uncertainty across agents.

To summarize, the communication strategy in our method involves agents exchanging two types of embeddings: local observation-action embeddings and embeddings describing state evo-

lution. This approach allows for efficient information sharing while maintaining the ability to reconstruct a comprehensive view of the environment.

6.5 Experiments

Methods: We compare OOD generalization of our method against: (1) MASIA [94] predicts the state from other agents’ observations within a QMIX method, (2) NDQ [92] and TarMAC [81] are prior work on MARL where the learned communication is not grounded on state prediction, (3) QMIX [104] and QMIX-based baselines QMIX_wState and QMIX-Att where each agent has access to the state and other agents’ observation, respectively. QMIX_wState uses the state vector in addition to the agents’ observation, a strong baseline for our method that assumes known initial state s_0 . QMIX-Att uses an attention network to integrate others’ observations, baseline for the self-attention architecture. Baseline architectures are shown in the appendix.

Scenarios: We evaluate these methods on three maps from the Starcraft Multi-Agent Challenge [106] namely *1o_10b_vs_1r*, *1o_2r_vs_4r*, and *3s_vs_5z*. The first two maps involve an aerial agent to track and communicate the location of an enemy agent to other friendly agents. The third map is a micromanagement task. For each map, we created OOD initial states that are different than the training initial states. The OOD states are shown in the appendix (see Figure 6.11). The methods are trained on the training map and evaluated in terms of win rate using episodes with the OOD initial states.

Improved Generalization: As shown in Figure 6.5, our method achieves significantly higher win rates on all three maps and in 8 out of 9 OOD states tested, compared to QMIX baselines and the prior MARL work. Compared to QMIX_wState, the prior work MASIA, NDQ

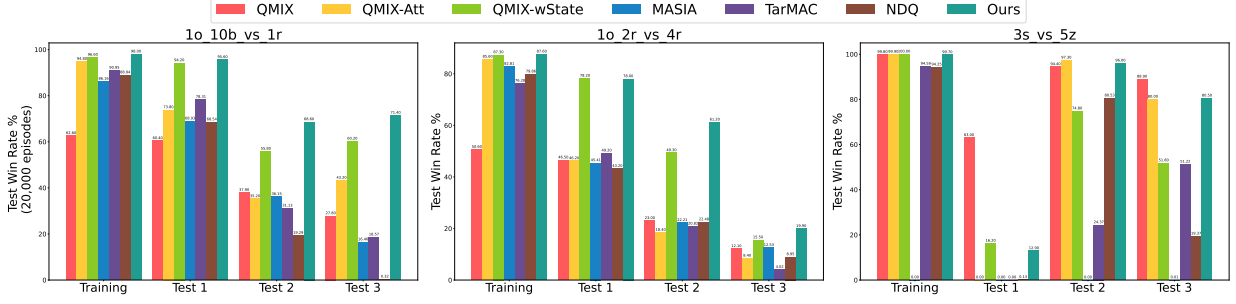


Figure 6.5: Comparison of methods on OOD initial states

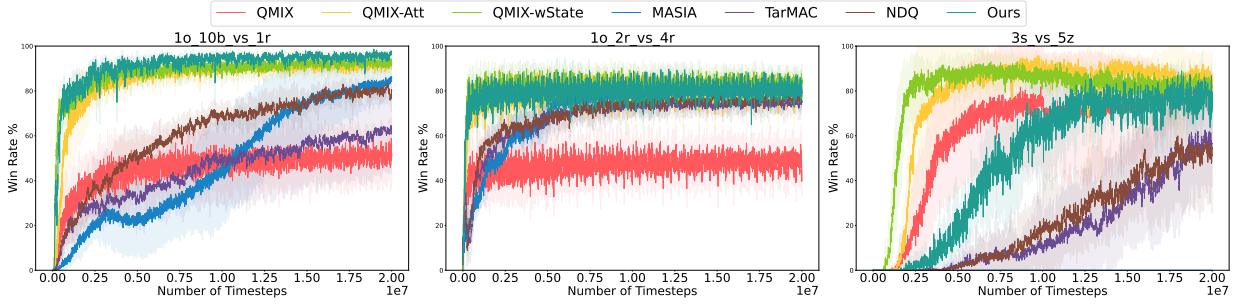


Figure 6.6: Training curves of various methods

and TarMAC shows brittleness. Our method performs even better than QMIX_wState, possibly due to training embeddings that produce both winning policies and accurate state prediction.

Sample Complexity: We observe (Figure 6.6) that our method has significantly lower sample complexity and faster convergence on the maps that require communication beyond visual range (*1o_10b_vs_1r* and *1o_2r_vs_4r*). Obviously, the performance of QMIX plateaus due to lack of communication, and MASIA and NDQ converge at a slower rate. These plots indicate that state prediction as in our COP-based training is an effective bias for quickly converging to robust policies.

Compression: To understand the success of our method further, we reduce the embedding size from 32 to 16, 8, then 4 (Figure 6.7). At lower message length, the win rate on OOD maps is surprisingly only a bit lower. Even at message length of 4, our method produces a significantly higher win rate than all the comparison methods. In the Appendix, we further ablate the impact

of communication by dropping all messages in test episodes.

Evaluation of COPs: Figure 6.8 shows the average MSE of the produced COPs on training and OOD maps. We find that the COPs are highly accurate on training and the first two OOD maps (an MSE of 0.05 is roughly only 6 pixels in a field of view of 9 pixels). We see clearly that the accuracy of the MSE (in our method) is directly correlated and predictive of win rate on OOD states. Beyond OOD generalization, our method produces human-interpretable COPs. Figure 6.9 shows an example COP where the location of the enemy is correctly predicted despite OOD initial state.

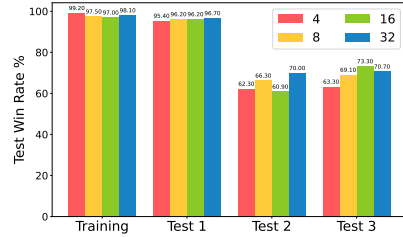


Figure 6.7: Varying message size (*1o_10b_vs_1r*).

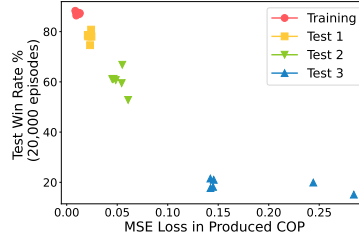


Figure 6.8: MSE vs win rate (*1o_2r_vs_4r*).

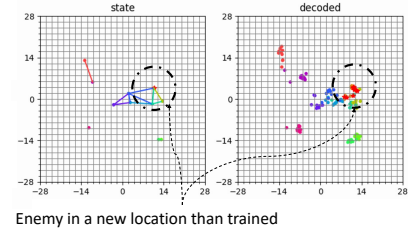


Figure 6.9: An example of reconstructed COP (*1o_10b_vs_1r*).

6.6 Appendix

6.6.1 Motivating Example in Details

Since the COP leads to better situational-awareness, we hypothesize that it also leads to improved out-of-distribution generalizations. We illustrate this with the following example. We use the *1o_10b_vs_1r* map from the Starcraft game [106]. This map comprises a team consisting of 1 flying unit (Overseer) and 10 ground units (Banelings), with the collective objective of eliminating a single enemy unit (Roach). The flying and enemy units are initialized simultaneously at

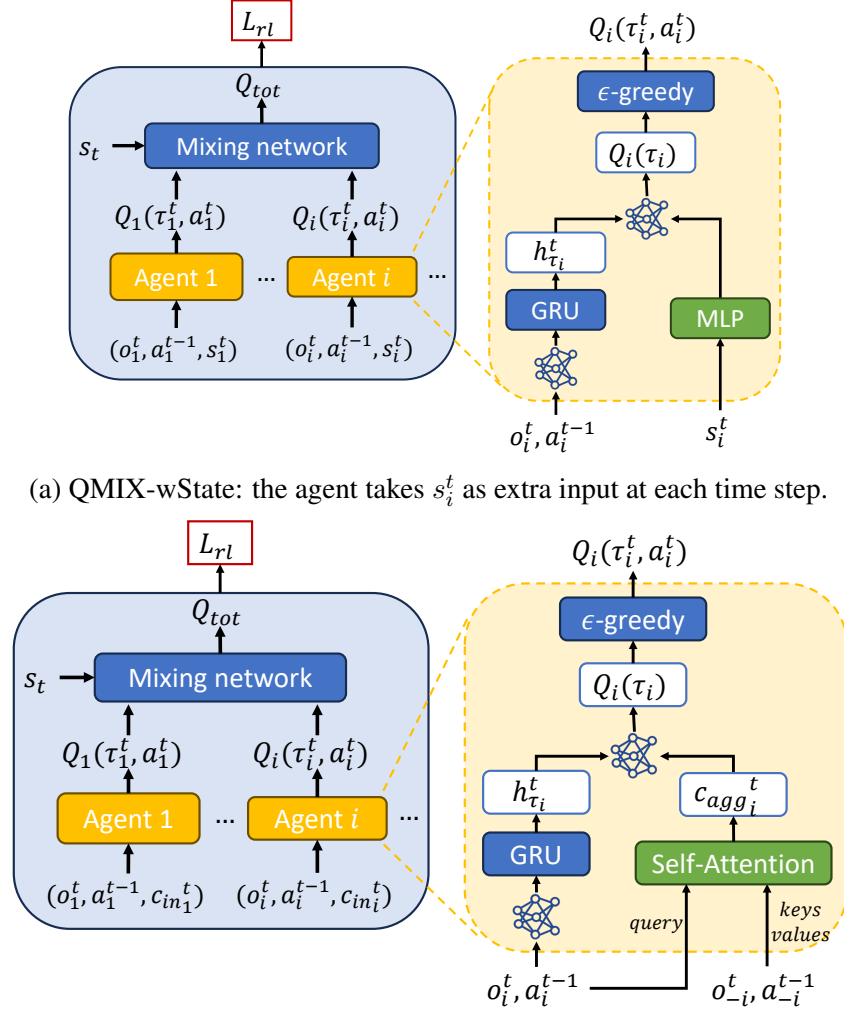
one of the four map corners. The ten ground units are distributed randomly across the map.

To introduce diversity and challenge, we have modified the initial distribution of the flying and enemy units and incorporated three out-of-distribution scenarios for this map, as shown in Figure 6.11a. In Test 1, the setup closely resembles the training scenario. After selecting one corner to generate the enemy unit, we randomly position the flying unit within the nearby circle corresponding to that corner. Test 2 and Test 3 present greater challenges with notably different initial state distributions. In Test 2, flying and enemy units are generated together across the entire map, introducing a more complex spatial arrangement. In Test 3, the flying unit is generated inside the middle circle, considerably distant from the enemy unit, creating a scenario demanding different strategic considerations.

We want to answer the following question: *Will policies exhibit improved generalization on out-of-distribution (OOD) laydowns when agents possess a higher degree of situational awareness?* We employed two strong baseline algorithms to explore this question: QMIX [104] (without communication) and NDQ [92] (with inter-agent communication). Additionally, we implemented QMIX-wstate, which does not utilize communication but assumes that each agent has access to both its own partial observation and the global state. As shown in Figure 6.2, our findings indicate that QMIX-wstate outperforms the other algorithms across all OOD laydown scenarios. This observation strongly suggests that using the global state, i.e., COP, is advantageous for enhancing generalization in OOD scenarios.

6.6.2 Pipelines of QMIX-based Baselines

The pipelines of QMIX-based baselines are visualized in Figure 6.10.

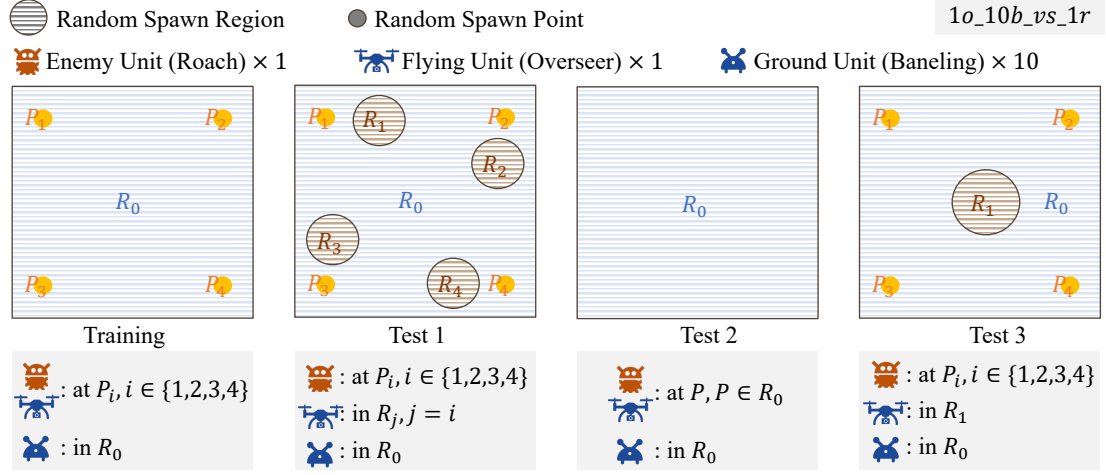


(b) QMIX-Att: the agents receives all other agents' observation and action as communication messages.

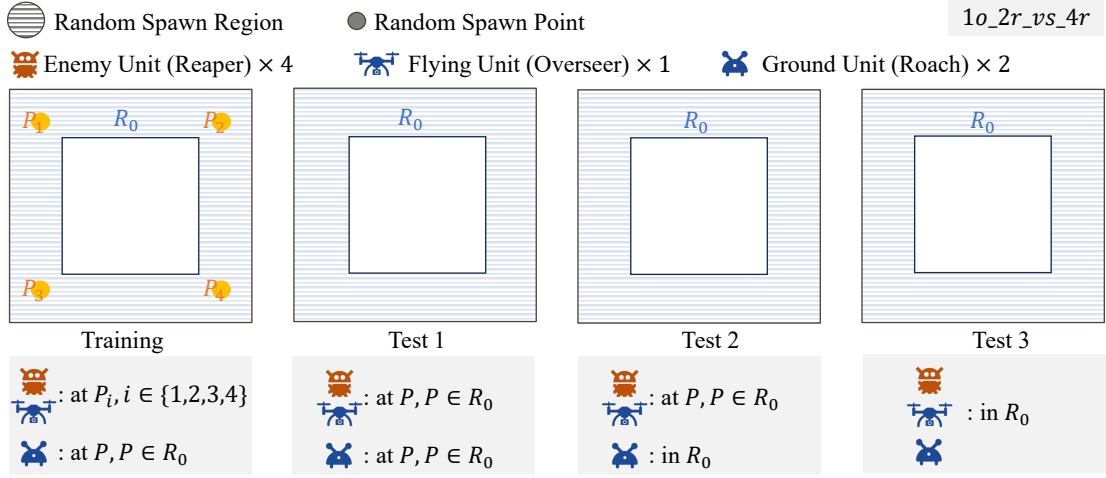
Figure 6.10: Pipelines of QMIX-based baseline algorithms.

6.6.3 Visualization of OOD Laydowns

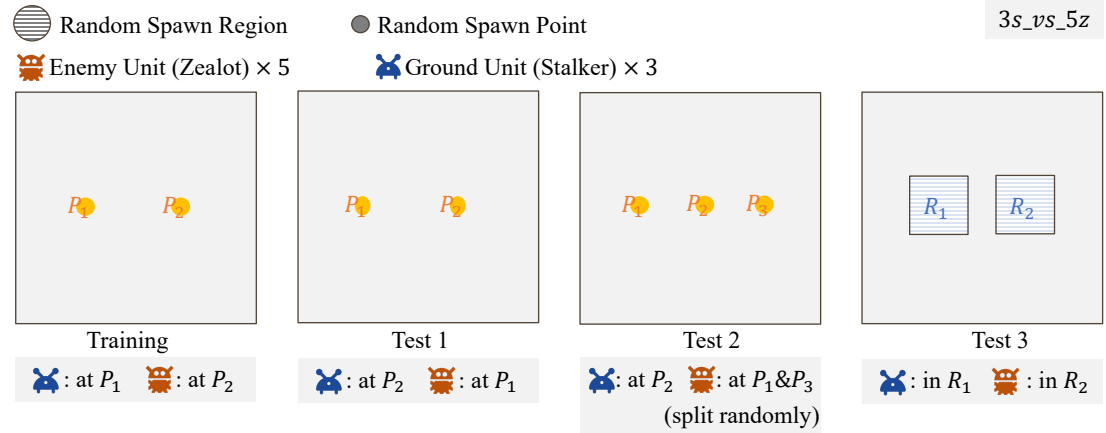
We visualize the OOD laydowns in Figure 6.11.



(a) 1o_10b_vs_1r



(b) 1o_2r_vs_4r



(c) 3s_vs_5z

Figure 6.11: Illustration of modified initial distributions for 1o_10b_vs_1r, 1o_2r_vs_4r and 3s_vs_5z. In this figure, "at" means that all indicated agents are generated at the same point P , while "in" denotes that all indicated agents are randomly generated at distinct locations within the region R .

6.6.4 Evaluate Message Dropping

To assess the significance of communication, we conducted experiments in which all messages were disabled during test episodes. We observe from *3s_vs_5z* test 2 that the performance drops approximately 20% in terms of win rate without communication. In other maps, the enemy is mostly stationary, so the information is primarily conveyed through the initial state and our approach effectively integrates this initial state information. However, even in such scenarios, we observed that the performance with communication was either equal to or superior to the performance without communication. This suggests that communication offers benefits beyond the initial state information, enhancing overall system performance.

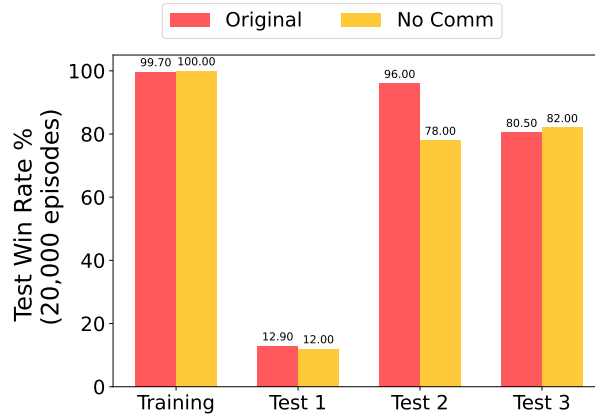


Figure 6.12: Evaluation results of our algorithm with and without communication.

6.6.5 QMIX with Varying Observation Sight Ranges

Previous works [94, 80] discussed the dilemma in sight ranges: agents with small sight ranges can only observe limited information, making it challenging for them to engage in effective cooperation with their teammates. Conversely, agents with extensive sight ranges are more susceptible to distractions, degrading the cooperation quality. However, they only draw conclu-

sions from the traffic junction and collaborative resource collection environments.

We conduct similar experiments on the *1o_10b_vs_1r* map by running QMIX with varying observation sight ranges. Interestingly, we have observed that a larger sight range results in quicker convergence, an increased win rate, and improved generalization to out-of-distribution (OOD) scenarios (see Figure 6.13, and more details in Table 6.1). We posit that this phenomenon is intertwined with the characteristics of the SMAC environment. In this context, effective cooperation among allies is critical for locating and attacking enemies to secure victory, and having a broader perspective on the overall situation is essential for accomplishing these complex tasks.

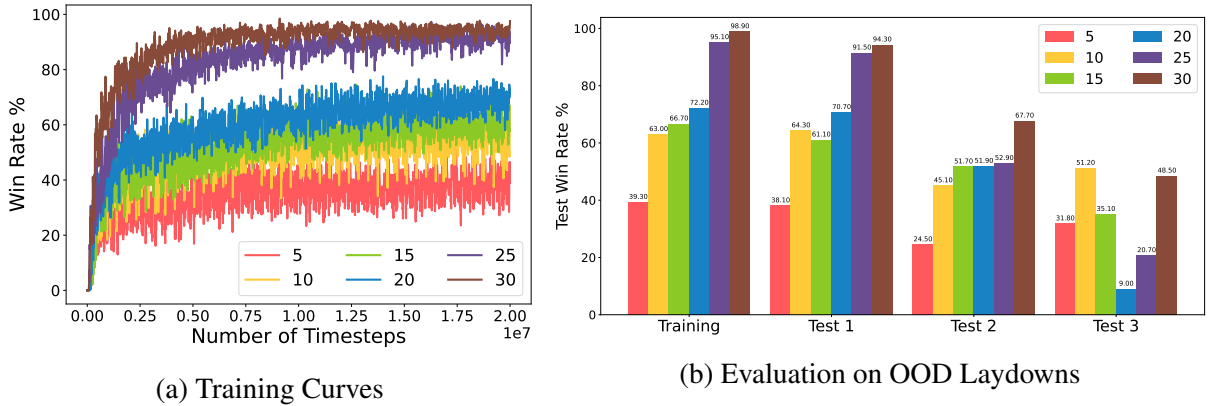


Figure 6.13: Training and evaluation results of QMIX with different sight ranges on *1o_10b_vs_1r*. A larger sight range gives us faster convergence speed and a higher win rate, which indicates that the agent can make wiser decisions if it is more aware of the global state under SMAC scenarios.

6.6.6 Hallucination

We also investigate the occurrence of false positives (hallucinations) at different time steps in the “*1o_10b_vs_1r*” scenario. By employing the disentangle loss to enforce local consistency, we have observed a noteworthy outcome: no hallucinations transpire within the agents’ field of view. The instances of hallucinations that do occur outside the field of view can be primarily

attributed to agents that are no longer active or are considered “dead”. These hallucinations tend to persist for a few time steps before fading away.

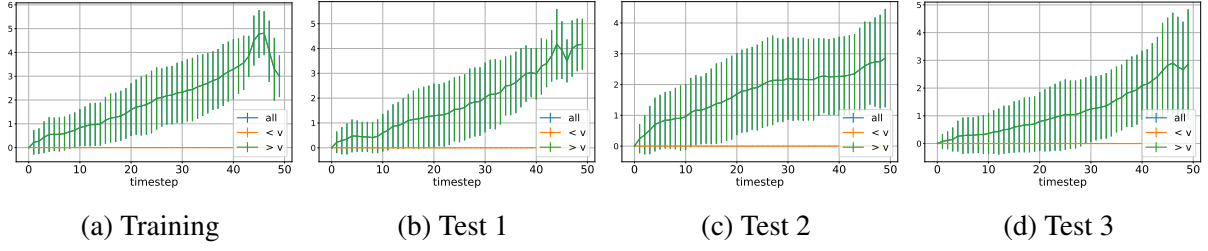


Figure 6.14: False positives across timesteps. $< v$ means within the field of view, and $> v$ means outside the field of view.

Sight Range		Training	Test 1	Test 2	Test 3
5	Win Rate	39.30%	38.10%	24.50%	31.80%
	Return	9.74 ± 8.77	9.52 ± 8.73	7.45 ± 7.61	8.65 ± 8.30
10	Win Rate	63.00%	64.30%	45.10%	51.20%
	Return	14.12 ± 8.42	14.40 ± 8.30	11.13 ± 8.62	12.34 ± 8.51
15	Win Rate	66.70%	61.10%	51.70%	35.10%
	Return	15.50 ± 7.17	14.73 ± 7.30	13.38 ± 7.46	9.86 ± 8.02
20	Win Rate	72.20%	70.70%	51.90%	9.00%
	Return	16.59 ± 6.28	16.37 ± 6.40	13.78 ± 7.00	6.53 ± 4.77
25	Win Rate	95.10%	91.50%	52.90%	20.70%
	Return	19.74 ± 2.84	19.26 ± 3.66	13.86 ± 6.95	8.55 ± 6.32
30	Win Rate	98.90%	94.30%	67.70%	48.50%
	Return	20.21 ± 1.32	19.62 ± 3.01	16.04 ± 6.25	13.28 ± 6.99

Table 6.1: Evaluations results on *1o_10b_vs_1r* over 20,000 episodes.

6.6.7 Additional Evaluation Details

Additional evaluation details can be found in Table 2-5.

Embedding Size		Training	Test 1	Test 2	Test 3
4	Win Rate	99.20%	95.40%	62.30%	63.30%
	Return	20.26 ± 1.14	19.77 ± 2.73	14.85 ± 7.22	15.21 ± 6.92
8	Win Rate	97.50%	96.20%	66.30%	69.10%
	Return	20.04 ± 1.98	19.87 ± 2.43	15.48 ± 6.93	16.25 ± 6.23
16	Win Rate	97.00%	96.20%	60.90%	73.30%
	Return	19.98 ± 2.19	19.89 ± 2.44	14.54 ± 7.40	16.74 ± 6.12
32	Win Rate	98.10%	96.70%	70.00%	70.70%
	Return	20.10 ± 1.75	19.93 ± 2.27	16.11 ± 6.56	16.47 ± 6.14

Table 6.2: Comparison of our algorithm with different embedding sizes on *1o_10b_vs_1r* over 20,000 episodes.

		Training	Test 1	Test 2	Test 3
QMIX	Win Rate	62.60%	60.40%	37.90%	27.80%
	Return	13.88 ± 8.64	13.59 ± 8.63	9.78 ± 8.52	8.73 ± 7.62
QMIX-Attn	Win Rate	94.80%	73.80%	35.20%	43.20%
	Return	19.69 ± 2.93	16.23 ± 7.10	9.63 ± 8.15	11.19 ± 8.28
QMIX-wState	Win Rate	96.60%	94.20%	55.80%	60.20%
	Return	19.92 ± 2.39	19.60 ± 3.12	13.31 ± 8.11	14.61 ± 7.27
MASIA	Win Rate	86.16%	68.93%	36.15%	16.46%
	Return	18.56 ± 4.56	15.59 ± 7.34	9.86 ± 8.16	6.7798 ± 6.50
TarMAC	Win Rate	90.95%	78.31%	31.13%	18.57%
	Return	19.25 ± 3.81	17.14 ± 6.42	8.93 ± 7.97	6.63 ± 7.00
NDQ	Win Rate	88.94%	68.54%	19.29%	0.32%
	Return	18.93 ± 4.18	15.74 ± 7.01	6.96 ± 6.99	1.68 ± 2.22
Our Method	Win Rate	98.00%	95.60%	68.60%	71.40%
	Return	20.11 ± 1.77	19.81 ± 2.63	15.79 ± 6.90	16.42 ± 6.37

Table 6.3: Evaluation results of various algorithms on *1o_10b_vs_1r* over 20,000 episodes.

		Training	Test 1	Test 2	Test 3
QMIX	Win Rate	50.60%	46.50%	23.00%	12.10%
	Return	10.75 $\pm$ 9.58	9.86 $\pm$ 9.65	7.74 $\pm$ 7.46	7.80 $\pm$ 5.15
QMIX-Attn	Win Rate	85.60%	46.20%	18.40%	8.40%
	Return	18.27 $\pm$ 4.28	9.81 $\pm$ 9.64	7.09 $\pm$ 6.91	7.10 $\pm$ 4.68
QMIX-wState	Win Rate	87.30%	78.20%	49.30%	15.50%
	Return	18.51 $\pm$ 3.96	16.82 $\pm$ 6.28	13.18 $\pm$ 7.01	8.78 $\pm$ 5.30
MASIA	Win Rate	82.81%	45.41%	22.21%	12.53%
	Return	17.93 $\pm$ 4.64	9.51 $\pm$ 9.73	7.19 $\pm$ 7.51	7.39 $\pm$ 5.43
TarMAC	Win Rate	76.28%	49.20%	20.82%	4.02%
	Return	16.60 $\pm$ 6.34	10.56 $\pm$ 9.53	7.11 $\pm$ 7.29	5.79 $\pm$ 3.96
NDQ	Win Rate	79.86%	43.20%	22.48%	8.95%
	Return	17.56 $\pm$ 4.97	9.28 $\pm$ 9.57	7.36 $\pm$ 7.46	6.72 $\pm$ 4.95
Our Method	Win Rate	87.60%	78.00%	61.20%	19.90%
	Return	18.61 $\pm$ 3.75	16.71 $\pm$ 6.50	14.92 $\pm$ 6.70	9.61 $\pm$ 5.54

Table 6.4: Evaluation results of various algorithms on *1o_2r_vs_4r* over 20,000 episodes.

		Training	Test 1	Test 2	Test 3
QMIX	Win Rate	99.80%	63.00%	94.40%	88.90%
	Return	21.17 $\pm$ 0.62	20.03 $\pm$ 4.05	21.66 $\pm$ 1.54	20.86 $\pm$ 2.50
QMIX-Attn	Win Rate	99.90%	0.00%	97.30%	80.00%
	Return	22.62 $\pm$ 0.44	0.01 $\pm$ 0.23	22.43 $\pm$ 1.30	20.94 $\pm$ 3.67
QMIX-wState	Win Rate	100.00%	16.20%	74.80%	51.80%
	Return	25.26 $\pm$ 1.13	18.09 $\pm$ 7.63	24.89 $\pm$ 3.57	17.39 $\pm$ 9.64
MASIA	Win Rate	0.00%	0.00%	0.00%	0.01%
	Return	9.81 $\pm$ 0.39	5.91 $\pm$ 2.08	5.28 $\pm$ 1.74	7.73 $\pm$ 1.41
TarMAC	Win Rate	94.59%	0.00%	24.37%	51.22%
	Return	21.67 $\pm$ 1.29	0.00 $\pm$ 0.11	18.84 $\pm$ 4.14	19.30 $\pm$ 4.60
NDQ	Win Rate	94.25%	0.13%	80.53%	19.37%
	Return	22.08 $\pm$ 1.50	6.10 $\pm$ 2.59	21.72 $\pm$ 2.89	14.99 $\pm$ 5.23
Our Method	Win Rate	99.70%	12.90%	96.00%	80.50%
	Return	20.95 $\pm$ 0.60	12.37 $\pm$ 7.28	21.22 $\pm$ 1.35	20.52 $\pm$ 3.26

Table 6.5: Evaluation results of various algorithms on *3s_vs_5z* over 20,000 episodes.

Chapter 7: Conclusion and Future Work

7.1 Summary of Contribution

This dissertation investigated the problem of learning with less effort in (multi-)robot systems, with a particular focus on reducing the burden of training and improving the generalization ability of learned policies. The research presented spans four core contributions, each addressing a key challenge in the broader goal of scalable and robust robot learning.

First, SKETCH-TO-SKILL demonstrated that human-drawn 2D sketches can serve as a lightweight and accessible form of supervision. By converting sketches into 3D trajectories, we enabled robots to bootstrap policy learning with minimal human effort. This approach reduced the need for teleoperated demonstrations while achieving strong task performance in both simulation and real-world settings.

Second, PegMARL introduced a framework for efficient multi-agent reinforcement learning using personalized demonstrations. By shaping learning through both behavior and transition discriminators, PegMARL avoided the need for joint demonstrations, offering a scalable alternative for training large teams with diverse agent roles.

Building on PegMARL, we developed DynPegMARL, a Graph Neural Network extension that enables learning from demonstrations across different team sizes. This architectural enhancement processes ally and enemy features separately in a permutation-invariant manner,

dramatically improving demonstration reusability by allowing the same framework to leverage any available demonstration data regardless of the original team configuration.

Third, TACTIC addressed generalization across observation conditions. Through task-agnostic contrastive pretraining, it enabled agents to communicate effectively and coordinate robustly, even under mismatched training and execution sight ranges. This approach showed consistent gains across a wide range of vision-limited scenarios.

Fourth, COP tackled the problem of generalization under state distribution shifts by learning structured shared representations that support distributed decision-making. By integrating historical and local information into a common operating picture, agents achieved better coordination and adaptability in out-of-distribution settings.

Together, these contributions form a cohesive set of tools and frameworks for reducing human and computational effort in training, while promoting generalization across heterogeneous and dynamic conditions. The overarching insight is that more efficient learning does not necessarily require more data or computation — but rather better structuring of inputs, demonstrations, and information sharing.

7.2 Limitations and Areas for Improvement

While our approaches have shown promise, several limitations suggest directions for continued research. Current methods still require some form of human guidance, whether through sketches, demonstrations, or initial training data. The sketch-based approach, while more accessible than teleoperation, is primarily demonstrated on manipulation tasks and may not readily extend to other domains, such as contact-rich manipulation requiring precise force control or

multi-agent coordination tasks where temporal synchronization between agents is critical. Similarly, our demonstration-guided methods assume the availability of at least some relevant demonstration data, which may not always be feasible in novel or safety-critical scenarios.

The generalization improvements, while promising, are demonstrated primarily in specific domains and may not transfer broadly across all types of environmental changes. For instance, TACTIC addresses sight range variations but may not handle other forms of observation changes, such as sensor modality differences or completely new environmental features.

Furthermore, most of our work operates within the paradigm of centralized training with decentralized execution, which may not scale to very large teams or handle scenarios where state is not available. In fully decentralized or ad hoc settings, agents often lack access to global information during training, and must learn independently and asynchronously. Adapting our methods to such settings remains an open challenge.

7.3 Future Research Directions

Several promising directions emerge from this work that could further advance the field of practical multi-robot learning.

Zero-Shot Coordination with Novel Teammates. One of the most exciting opportunities lies in developing capabilities for zero-shot coordination — enabling robots to immediately collaborate with previously unseen teammates without any additional training. This could build on our demonstration reuse work by developing more sophisticated reasoning about teammate capabilities and intentions. Future research might explore how large language models could help robots understand and adapt to new teammate behaviors through natural language descriptions

or brief observations, potentially eliminating the need for demonstration data entirely in some scenarios.

Cross-Domain Transfer and Meta-Learning. Our current approaches focus primarily on generalization within specific domains. Future work could explore how learned coordination principles transfer across fundamentally different tasks and environments. Meta-learning approaches might enable robots to quickly adapt their coordination strategies when moving from warehouse operations to disaster response, or from ground-based to aerial platforms. This could significantly reduce the training effort for new applications by leveraging coordination patterns learned in previous domains.

Integration with Foundation Models. The rapid advancement of foundation models presents opportunities to enhance multi-robot learning with pre-trained knowledge about the world, common tasks, and coordination principles. Future research might explore how to effectively combine our demonstration-guided approaches with foundation models to achieve even more sample-efficient learning and better zero-shot generalization.

Bibliography

- [1] Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob Foerster, and Shimon Whiteson. Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.
- [2] Peter R Wurman, Raffaello D’Andrea, and Mick Mountz. Coordinating hundreds of co-operative, autonomous vehicles in warehouses. *AI magazine*, 29(1):9–9, 2008.
- [3] Robin R Murphy, Satoshi Tadokoro, and Alexander Kleiner. Disaster robotics. In *Springer handbook of robotics*, pages 1577–1604. Springer, 2016.
- [4] Jörg Krüger, Terje K Lien, and Alexander Verl. Cooperation of human and machines in assembly lines. *CIRP annals*, 58(2):628–646, 2009.
- [5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.
- [6] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.
- [7] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332, 2017.
- [8] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.
- [9] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019.
- [10] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.
- [11] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

- [12] Sergey Levine and Vladlen Koltun. Guided policy search. In *International conference on machine learning*, pages 1–9. PMLR, 2013.
- [13] Abhishek Gupta, Clemens Eppner, Sergey Levine, and Pieter Abbeel. Learning dexterous manipulation for a soft robotic hand from human demonstrations. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3786–3793. IEEE, 2016.
- [14] Ian Lenz, Ross A Knepper, and Ashutosh Saxena. Deepmpc: Learning deep latent features for model predictive control. In *Robotics: Science and Systems*, volume 10, page 25. Rome, Italy, 2015.
- [15] Erfu Yang and Dongbing Gu. Multiagent reinforcement learning for multi-robot systems: A survey. Technical report, tech. rep, 2004.
- [16] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016.
- [17] Tianxu Li, Kun Zhu, Nguyen Cong Luong, Dusit Niyato, Qihui Wu, Yang Zhang, and Bing Chen. Applications of multi-agent reinforcement learning in future internet: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 24(2):1240–1279, 2022.
- [18] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control*, pages 321–384, 2021.
- [19] Caroline Wang, Ishan Durugkar, Elad Liebman, and Peter Stone. Dm2: Decentralized multi-agent reinforcement learning via distribution matching. 2023.
- [20] Jiaming Song, Hongyu Ren, Dorsa Sadigh, and Stefano Ermon. Multi-agent generative adversarial imitation learning. *Advances in neural information processing systems*, 31, 2018.
- [21] Weixun Wang, Tianpei Yang, Yong Liu, Jianye Hao, Xiaotian Hao, Yujing Hu, Yingfeng Chen, Changjie Fan, and Yang Gao. From few to more: Large-scale dynamic multiagent curriculum learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7293–7300, 2020.
- [22] Peihong Yu, Amisha Bhaskar, Anukriti Singh, Zahiruddin Mahammad, and Pratap Tokekar. Sketch-to-skill: Bootstrapping robot learning with human drawn trajectory sketches. *arXiv preprint arXiv:2503.11918*, 2025.
- [23] Peihong Yu, Manav Mishra, Alec Koppel, Carl Busart, Priya Narayan, Dinesh Manocha, Amrit Singh Bedi, and Pratap Tokekar. Beyond joint demonstrations: Personalized expert guidance for efficient multi-agent reinforcement learning. *Transactions on Machine Learning Research*, 2025.

- [24] Peihong Yu, Manav Mishra, Syed Zaidi, and Pratap Tokekar. TACTIC: Task-agnostic contrastive pre-training for inter-agent communication. In *The 24th International Conference on Autonomous Agents and Multi-Agent Systems*, 2024.
- [25] Peihong Yu, Bhoram Lee, Aswin Raghavan, Supun Samarasekera, Pratap Tokekar, and James Zachary Hare. Enhancing multi-agent coordination through common operating picture integration. In *First Workshop on Out-of-Distribution Generalization in Robotics at CoRL 2023*, 2023.
- [26] A. Billard and D. Grollman. Robot learning by demonstration. *Scholarpedia*, 8(12):3824, 2013. revision #138061.
- [27] Linping Chan, Fazel Naghdy, and David Stirling. Application of adaptive controllers in teleoperation systems: A survey. *IEEE Transactions on Human-Machine Systems*, 44(3):337–352, 2014.
- [28] Federica Ferraguti, Nicola Preda, Marcello Bonfe, and Cristian Secchi. Bilateral teleoperation of a dual arms surgical robot with passive virtual fixtures generation. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4223–4228. IEEE, 2015.
- [29] Joao Bimbo, Claudio Pacchierotti, Marco Aggravi, Nikos Tsagarakis, and Domenico Prattichizzo. Teleoperation in cluttered environments using wearable haptic feedback. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3401–3408. IEEE, 2017.
- [30] Nur Muhammad Mahi Shafiullah, Anant Rai, Haritheja Etukuru, Yiqian Liu, Ishan Misra, Soumith Chintala, and Lerrel Pinto. On bringing robots home. *arXiv preprint arXiv:2311.16098*, 2023.
- [31] Tatsuya Kamijo, Cristian C Beltran-Hernandez, and Masashi Hamaya. Learning variable compliance control from a few demonstrations for bimanual robot with haptic feedback teleoperation system. *arXiv preprint arXiv:2406.14990*, 2024.
- [32] Jiayuan Gu, Sean Kirmani, Paul Wohlhart, Yao Lu, Montserrat Gonzalez Arenas, Kanishka Rao, Wenhao Yu, Chuyuan Fu, Keerthana Gopalakrishnan, Zhuo Xu, et al. Rt-trajectory: Robotic task generalization via hindsight trajectory sketches. *arXiv preprint arXiv:2311.01977*, 2023.
- [33] Priya Sundaresan, Quan Vuong, Jiayuan Gu, Peng Xu, Ted Xiao, Sean Kirmani, Tianhe Yu, Michael Stark, Ajinkya Jain, Karol Hausman, et al. Rt-sketch: Goal-conditioned imitation learning from hand-drawn sketches. 2024.
- [34] Weiming Zhi, Tianyi Zhang, and Matthew Johnson-Roberson. Learning from demonstration via probabilistic diagrammatic teaching. *arXiv preprint arXiv:2309.03835*, 2023.
- [35] Harish chaandar Ravichandar, Athanasios S. Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. *Annu. Rev. Control. Robotics Auton. Syst.*, 3:297–330, 2020.

- [36] Jens Kober and Jan Peters. Learning motor primitives for robotics. In *2009 IEEE International Conference on Robotics and Automation*. IEEE, May 2009.
- [37] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25(2):328–373, February 2013.
- [38] Alexandros Paraschos, Elmar Rueckert, Jan Peters, and Gerhard Neumann. Model-free probabilistic movement primitives for physical interaction. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, September 2015.
- [39] S. Mohammad Khansari-Zadeh and Aude Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, October 2011.
- [40] S. Mohammad Khansari-Zadeh and Aude Billard. Learning control lyapunov function to ensure stability of dynamical system-based robot reaching motions. *Robotics and Autonomous Systems*, 62(6):752–765, June 2014.
- [41] Petar Bevanda, Johannes Kirmayr, Stefan Sosnowski, and Sandra Hirche. Learning the koopman eigendecomposition: A diffeomorphic approach. In *2022 American Control Conference (ACC)*, volume 22, page 2736–2741. IEEE, June 2022.
- [42] Weiyong Si, Ning Wang, and Chenguang Yang. A review on manipulation skill acquisition through teleoperation-based learning from demonstration. *Cognitive Computation and Systems*, 3(1):1–16, 2021.
- [43] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 5628–5635. IEEE, 2018.
- [44] Michael Drolet, Simon Stepputtis, Siva Kailas, Ajinkya Jain, Jan Peters, Stefan Schaal, and Heni Ben Amor. A comparison of imitation learning algorithms for bimanual manipulation. *IEEE Robotics and Automation Letters*, 2024.
- [45] Pinaki Nath Chowdhury, Ayan Kumar Bhunia, Aneeshan Sain, Subhadeep Koley, Tao Xiang, and Yi-Zhe Song. Scenetriology: On human scene-sketch and its complementarity with photo and text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10972–10983, 2023.
- [46] Pinaki Nath Chowdhury, Ayan Kumar Bhunia, Aneeshan Sain, Subhadeep Koley, Tao Xiang, and Yi-Zhe Song. What can human sketches do for object detection? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15083–15094, 2023.
- [47] Ayan Kumar Bhunia, Subhadeep Koley, Amandeep Kumar, Aneeshan Sain, Pinaki Nath Chowdhury, Tao Xiang, and Yi-Zhe Song. Sketch2saliency: learning to detect salient

- objects from human drawings. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2733–2743, 2023.
- [48] Laura Smith, Ilya Kostrikov, and Sergey Levine. A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. *arXiv preprint arXiv:2208.07860*, 2022.
 - [49] Hengyuan Hu, Suvir Mirchandani, and Dorsa Sadigh. Imitation bootstrapped reinforcement learning. *arXiv preprint arXiv:2311.02198*, 2023.
 - [50] Amisha Bhaskar, Zahiruddin Mahammad, Sachin R Jadhav, and Pratap Tokekar. Navinact: Combining navigation and imitation learning for bootstrapping reinforcement learning. *arXiv preprint arXiv:2408.04054*, 2024.
 - [51] Diederik P Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
 - [52] Hartmut Prautzsch, Wolfgang Boehm, and Marco Paluszny. *Bézier and B-spline techniques*, volume 6. Springer, 2002.
 - [53] Carl de Boor. Package for calculating with b-splines. *SIAM Journal on Numerical Analysis*, 14(3):441–472, 1977.
 - [54] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.
 - [55] Bingyi Kang, Zequn Jie, and Jiashi Feng. Policy optimization with demonstrations. In *International conference on machine learning*, pages 2469–2478. PMLR, 2018.
 - [56] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019.
 - [57] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021.
 - [58] Younggyo Seo, Danijar Hafner, Hao Liu, Fangchen Liu, Stephen James, Kimin Lee, and Pieter Abbeel. Masked world models for visual control. In *Conference on Robot Learning*, pages 1332–1344. PMLR, 2023.
 - [59] Jie Chen and Wenjun Xu. Policy gradient from demonstration and curiosity. *IEEE Transactions on Cybernetics*, 2022.
 - [60] Desik Rengarajan, Gargi Vaidya, Akshay Sarvesh, Dileep Kalathil, and Srinivas Shakkottai. Reinforcement learning with sparse rewards using guidance from offline demonstration. In *International Conference on Learning Representations*, 2022.

- [61] Hyun-Rok Lee and Taesik Lee. Improved cooperative multi-agent reinforcement learning algorithm augmented by mixing demonstrations from centralized policy. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1089–1098, 2019.
- [62] Yunbo Qiu, Yuzhu Zhan, Yue Jin, Jian Wang, and Xudong Zhang. Sample-efficient multi-agent reinforcement learning with demonstrations for flocking control. *arXiv preprint arXiv:2209.08351*, 2022.
- [63] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Neural Information Processing Systems (NIPS)*, 2017.
- [64] Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908*, 2017.
- [65] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.
- [66] Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.
- [67] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.
- [68] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.
- [69] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.
- [70] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.
- [71] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
- [72] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

- [73] Matteo Bettini, Ajay Shankar, and Amanda Prorok. Heterogeneous multi-robot reinforcement learning. *arXiv preprint arXiv:2301.07137*, 2023.
- [74] Chao Yu, Akash Velu, Eugene Vinitzky, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955*, 2021.
- [75] Jennifer She, Jayesh K Gupta, and Mykel J Kochenderfer. Agent-time attention for sparse rewards multi-agent reinforcement learning. *arXiv preprint arXiv:2210.17540*, 2022.
- [76] Tim Franzmeyer, Mateusz Malinowski, and Joao F. Henriques. Learning altruistic behaviours in reinforcement learning without external rewards. In *International Conference on Learning Representations*, 2022.
- [77] Lucian Busoni, Robert Babuska, and Bart De Schutter. A comprehensive survey of multi-agent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.
- [78] Kaige Qu, Weihua Zhuang, Qiang Ye, Wen Wu, and Xuemin Shen. Model-assisted learning for adaptive cooperative perception of connected autonomous vehicles. *IEEE Transactions on Wireless Communications*, 23(8):8820–8835, 2024.
- [79] Zhi Zhang, Jiachen Yang, and Hongyuan Zha. Integrating independent and centralized multi-agent reinforcement learning for traffic signal network optimization. *arXiv preprint arXiv:1909.10651*, 2019.
- [80] Jianzhun Shao, Hongchang Zhang, Yun Qu, Chang Liu, Shuncheng He, Yuhang Jiang, and Xiangyang Ji. Complementary attention for multi-agent reinforcement learning. 2023.
- [81] Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. Tarmac: Targeted multi-agent communication. In *International Conference on Machine Learning*, pages 1538–1546. PMLR, 2019.
- [82] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In *Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS’16*, page 2252–2260, Red Hook, NY, USA, 2016. Curran Associates Inc.
- [83] Yen-Cheng Liu, Junjiao Tian, Chih-Yao Ma, Nathan Glaser, Chia-Wen Kuo, and Zsolt Kira. Who2com: Collaborative perception via learnable handshake communication. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6876–6883. IEEE, 2020.
- [84] Yaru Niu, Rohan R Paleja, and Matthew C Gombolay. Multi-agent graph-attention communication and teaming. In *Proceedings of the AAMAS Conference*, volume 21. International Foundation for Autonomous Agents and Multiagent Systems, 2021.

- [85] Esmaeil Seraj, Zheyuan Wang, Rohan Paleja, Matthew Sklar, Anirudh Patel, and Matthew Gombolay. Heterogeneous graph attention networks for learning diverse communication. *arXiv preprint arXiv:2108.09568*, 2021.
- [86] Siqi Shen, Yongquan Fu, Huayou Su, Hengyue Pan, Peng Qiao, Yong Dou, and Cheng Wang. Graphcomm: A graph neural network based method for multi-agent reinforcement learning. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3510–3514. IEEE, 2021.
- [87] Chuangchuang Sun, Macheng Shen, and Jonathan P How. Scaling up multiagent reinforcement learning for robotic systems: Learn an adaptive sparse communication graph. In *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 11755–11762. IEEE, 2020.
- [88] Shengchao Hu, Li Shen, Ya Zhang, and Dacheng Tao. Learning multi-agent communication from graph modeling perspective. *arXiv preprint arXiv:2405.08550*, 2024.
- [89] Yaodong Yang, Jianye Hao, Ben Liao, Kun Shao, Guangyong Chen, Wulong Liu, and Hongyao Tang. Qatten: A general framework for cooperative multiagent reinforcement learning, 2020.
- [90] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, pages 2244–2252, 2016.
- [91] Y. L. Lo, B. Sengupta, J. Foerster, and M. Noukhovitch. Learning multi-agent communication with contrastive learning. *arXiv preprint arXiv:2307.01403*, 2024.
- [92] Tonghan Wang, Jianhao Wang, Chongyi Zheng, and Chongjie Zhang. Learning nearly decomposable value functions via communication minimization. *arXiv preprint arXiv:1910.05366*, 2019.
- [93] Toru Lin, Jacob Huh, Christopher Stauffer, Ser Nam Lim, and Phillip Isola. Learning to ground multi-agent communication with autoencoders. *Advances in Neural Information Processing Systems*, 34:15230–15242, 2021.
- [94] Cong Guan, Feng Chen, Lei Yuan, Chenghe Wang, Hao Yin, Zongzhang Zhang, and Yang Yu. Efficient multi-agent communication via self-supervised information aggregation. *Advances in Neural Information Processing Systems*, 35:1020–1033, 2022.
- [95] Yali Du, Bo Liu, Vincent Moens, Ziqi Liu, Zhicheng Ren, Jun Wang, Xu Chen, and Haifeng Zhang. Learning correlated communication topology in multi-agent reinforcement learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 456–464. International Foundation for Autonomous Agents and Multiagent Systems, 2021.
- [96] Kejun Zhang, Jayesh K Gupta, Alfred O Hero III, and Mykel J Kochenderfer. Succinct and robust multi-agent communication with temporal message control. *arXiv preprint arXiv:2107.06609*, 2021.

- [97] A. van den Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.
- [98] Ruijie Zheng, Xiyao Wang, Yanchao Sun, Shuang Ma, Jieyu Zhao, Huazhe Xu, Hal Daumé, and Furong Huang. Taco: temporal latent action-driven contrastive loss for visual reinforcement learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2024. Curran Associates Inc.
- [99] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 18661–18673, 2020.
- [100] Zhiwei Xu, Bin Zhang, Dapeng Li, Zeren Zhang, Guangchong Zhou, Hao Chen, and Guoliang Fan. Consensus learning for cooperative multi-agent reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(10):11726–11734, Jun. 2023.
- [101] A. Singh, T. Jain, and S. Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. In *International Conference on Learning Representations*, 2019.
- [102] S. Q. Zhang, Q. Zhang, and J. Lin. Efficient communication in multi-agent reinforcement learning via variance based control. In *Proceedings of the NeurIPS 32nd Conference on Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [103] Frans A Oliehoek, Christopher Amato, et al. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016.
- [104] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. *arxiv e-prints*, page. *arXiv preprint arXiv:1803.11485*, 2018.
- [105] Kristine Steen-Tveit and Bjørn Erik Munkvold. From common operational picture to common situational understanding: An analysis based on practitioner perspectives. *Safety Science*, 142:105381, 2021.
- [106] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.
- [107] Changxi Zhu, Mehdi Dastani, and Shihan Wang. A survey of multi-agent reinforcement learning with communication. *arXiv preprint arXiv:2203.08975*, 2022.