

**CAPABILITY-AWARE SHARED HYPERNETWORKS FOR HETEROGENEOUS  
MULTI-AGENT COORDINATION**

A Dissertation  
Presented to  
The Academic Faculty

By

Kevin Fu

In Partial Fulfillment  
of the Requirements for the Degree  
Master of Science in Computer Science in the  
School of Computer Science  
Department of Computer Science

Georgia Institute of Technology

December 2024

© Kevin Fu 2024

**CAPABILITY-AWARE SHARED HYPERNETWORKS FOR HETEROGENEOUS  
MULTI-AGENT COORDINATION**

Thesis committee:

Dr. Harish Ravichandar  
School of Interactive Computing  
*Georgia Institute of Technology*

Dr. Sonia Chernova  
School of Interactive Computing  
*Georgia Institute of Technology*

Dr. Matthew Gombolay  
School of Interactive Computing  
*Georgia Institute of Technology*

Date approved: December 6, 2024

## ACKNOWLEDGMENTS

First, I would like to thank my advisor, Dr. Harish Ravichandar. Without his mentorship and guidance this thesis would not have been possible. I am truly grateful I was able to work in his lab and feel I have learned an enormous amount from him, both personally and professionally.

I would also like to thank the members of my thesis committee, Dr. Matthew Gombolay and Dr. Sonia Chernova, for their valuable feedback and guidance.

I would like to thank all of my research labmates for providing an enjoyable and intellectually stimulating environment to do research in. Special thanks to Max Rudolph, Reza Torbati, and especially Pierce Howell and Shalin Jain, without whom many parts of this thesis would not have been possible to complete.

I would like to thank all the friends and family who have been supportive of my endeavors throughout my time at Georgia Tech. It is truly bittersweet for me to realize this completion of this thesis marks the end of my time here. Finally, extra special thanks to my girlfriend for being supportive of me through the most stressful times of my master's program.

## TABLE OF CONTENTS

<b>Acknowledgments</b> . . . . .	iii
<b>List of Tables</b> . . . . .	vii
<b>List of Figures</b> . . . . .	viii
<b>List of Acronyms</b> . . . . .	xi
<b>Summary</b> . . . . .	xii
<b>Chapter 1: Introduction</b> . . . . .	1
<b>Chapter 2: Related Works</b> . . . . .	4
2.1 Learning for multi-robot teams . . . . .	4
2.2 Heterogeneity within MARL . . . . .	4
2.2.1 Shared-Parameter Architectures . . . . .	5
2.2.2 Learning for Heterogeneous Teams . . . . .	5
2.3 Hypernetworks . . . . .	7
2.3.1 Hypernetworks for multi-agent learning . . . . .	7
<b>Chapter 3: Capability-Aware Policy Architectures</b> . . . . .	8
3.1 Capability Awareness and Communication for Adaptive Teaming . . . . .	8

3.1.1	Modeling Heterogeneous Multi-Robot Teams . . . . .	8
3.1.2	Problem Description . . . . .	9
3.1.3	Policy Architecture . . . . .	10
3.1.4	Graph Neural Networks . . . . .	10
3.1.5	Policy Architectures . . . . .	11
3.1.6	Policy Training Hyperparameters . . . . .	12
3.1.7	Training Procedure . . . . .	13
3.2	Experimental Design . . . . .	13
3.3	Results . . . . .	15
3.3.1	Heterogeneous Material Transport . . . . .	15
3.3.2	Heterogeneous Sensor Network . . . . .	17
3.4	Conclusion . . . . .	19
<b>Chapter 4: Capability-Aware Shared Hypernetworks . . . . .</b>		<b>24</b>
4.1	Proposed Architecture . . . . .	24
4.1.1	RNN Encoder . . . . .	24
4.1.2	Adaptive Decoder . . . . .	26
4.1.3	Hyper Adapter . . . . .	26
4.2	Experimental Procedure . . . . .	27
4.2.1	Learning Paradigms . . . . .	27
4.2.2	Heterogeneous Multi-Agent Tasks . . . . .	28
4.2.3	Baseline Architectures . . . . .	30
4.2.4	Training and Testing Teams . . . . .	33

<b>Chapter 5: Results</b>	34
5.1 Sample and Parameter Efficiency	34
5.2 Zero-Shot Generalization	36
5.3 Behavioral Heterogeneity	37
5.4 Comparing Trends Across RL and IL	37
5.5 Comparing Trends Across Tasks	38
5.6 Layer Normalization	38
<b>Chapter 6: Conclusion and Discussion</b>	44
6.1 Limitations	44
6.2 Future Work	45
<b>Appendices</b>	47
Appendix A: Evaluating Behavioral Diversity with SND	48
Appendix B: Additional Training Details	49
<b>References</b>	54

## LIST OF TABLES

3.1	Hyperparameters used to train each of the policies with PPO. . . . .	12
B.1	Hyperparameters for MAPPO . . . . .	50
B.2	Hyperparameters for QMix . . . . .	50
B.3	Hyperparameters for DAgger . . . . .	51
B.4	Firefighting Training Teams . . . . .	52
B.5	Firefighting Testing Teams . . . . .	52
B.6	Transport Training Teams . . . . .	53
B.7	Transport Testing Teams . . . . .	53

## LIST OF FIGURES

3.1	We investigate the role of capability awareness and communication in generalizing decentralized heterogeneous multi-robot coordination policies to teams of new composition, size, and robots. . . . .	9
3.2	When evaluated on <i>teams seen during training</i> , capability-aware policies performed comparably to ID-based policies in terms of both average return (higher is better) and task-specific metrics (lower is better). . . . .	16
3.3	When generalizing to <i>new team compositions and sizes</i> in HMT, capability-based policies consistently outperformed ID-based policies in terms of average steps taken to meet the quota (lower is better). . . . .	20
3.4	When generalizing to <i>new robots with unseen values for capabilities</i> in HMT, policies that are only aware of capabilities (CA (MLP) and CA (GNN) ) outperformed policies that also communicated capabilities (CA+CC (GNN) ) in terms of average number of steps taken to transport the required material (lower is better). . . . .	21
3.5	When generalizing to <i>new team compositions and sizes</i> in HSN, capability-based policies consistently outperformed ID-based baselines in terms of average return (higher is better). Further, combining awareness and communication of capabilities resulted in the best generalization performance. . . . .	22
3.6	When generalizing to <i>teams comprised of new robots</i> in HSN, combining awareness and communication of capabilities (CA+CC (GNN) ) achieves higher average returns than baselines that are merely aware of capabilities, irrespective of whether they communicate observations (CA (GNN) ) or not (CA (MLP) ). . . . .	23
4.1	We present Capability-Aware Shared Hypernetworks (CASH), an architecture for flexible and decentralized heterogeneous multi-agent teaming. CASH leverages “soft parameter sharing” using hypernetworks to condition agent decision making on individual capabilities, collective capabilities, and observations. . . . .	25

4.2	Example still from the simulated <code>Firefighting</code> environment with three agents (green circles) and two fires (red circles). Agents aim to effectively allocate themselves to fires within a fixed time horizon. Radius of agent and fire circles represent firefighting capacity and fire severity, respectively, and a fire is considered extinguished if the sum of all agent firefighting capacity on a given fire is greater than the severity of said fire. . . . .	29
4.3	Example still taken from execution within <code>Transport</code> . Agents are represented as filled-in blue circles, with associated capacity for different materials listed in text following each agent. The larger hollow circles are sites of interest. The blue and green circles represent different sources of materials, and the purple circle is the depot which agents bring back materials to. . . . .	31
5.1	Training returns ( $\uparrow$ ) across two tasks and three learning paradigms. CASH consistently outperforms the baselines across learning paradigms and tasks despite having 20-40% of the learnable parameters (see Figure 5.2). For QMIX and MAPPO, results are shown from 10 random seeds. For DAGger, results are obtained over 3 random seeds. . . . .	35
5.2	Learnable parameter counts ( $\downarrow$ ) for each baseline architecture compared in our experiments. Variations between tasks are solely due to differences in observation and action space dimensions – hidden layers and depths are kept constant between tasks for the same baseline. All of the performance results of CASH presented in the other figures are obtained with less than half of the learnable parameters as the baseline methods. . . . .	36
5.3	Success rate ( $\uparrow$ ) on unseen team compositions with unseen agent capabilities across two tasks and three learning paradigms. CASH consistently outperforms baseline methods when generalizing to new teams or agents. . . . .	40
5.4	Additional task-specific performance metric across two tasks and three learning paradigms for unseen teams. Percentage of Fires Extinguished ( $\uparrow$ , on left) is for <code>Firefighting</code> . Makespan ( $\downarrow$ , on right) is for <code>Transport</code> . These metrics provide additional context beyond the success rates in Figure 5.3. . . . .	41
5.5	Behavioral diversity ( $\updownarrow$ ) across two tasks and three learning paradigms. In all cases but one (MAPPO/ <code>Firefighting</code> ), CASH exhibits more diverse behavior than the baseline approaches. Qualitative analysis of MAPPO/ <code>Firefighting</code> suggests that RNN-EXP is able to learn diverse behaviors, but the corresponding success rate in Figure 5.3 suggests these behaviors are unproductive for task success. Conversely, CASH is able to learn an appropriate level of heterogeneity for strong task success. . . . .	42

5.6 Impact of removing Layer Normalization from the Hyper Adapter of CASH when training with DAGger across two tasks. It is evident that LayerNorm is a crucial component in stabilizing the training of the hypernetwork within CASH. . . . . 43



## SUMMARY

Cooperative heterogeneous multi-agent tasks require agents to behave in a flexible and complementary manner that best leverages their diverse capabilities. Learning-based approaches to this challenge span a spectrum between two endpoints: i) shared-parameter methods, which assign an ID to each agent to encode diverse behaviors within a single architecture for sample-efficiency, but are limited in their ability to learn diverse behaviors; ii) independent methods, which learn a separate policy for each agent, enabling greater diversity at the cost of sample- and parameter-efficiency. Prior work on learning for heterogeneous multi-agent teams has already explored the middle ground of this spectrum by learning shared-parameter or independent policies for *classes* of agents, allowing for a compromise between diversity and efficiency. However, these approaches still do not reason over the impact of agent capabilities on behavior, and thus cannot generalize to unseen agents or team compositions. In this work, we aim to enable flexible and heterogeneous coordination without sacrificing diversity, sample efficiency or generalization to unseen agents and teams. First, inspired by work from trait-based heterogeneous task allocation, we explore how *capability-awareness* enables generalization to unseen agents and teams. We thoroughly evaluate our GNN-based capability-aware policy architecture, showing that it can more effectively generalize than existing work.

Then, inspired by recent work in transfer learning and meta-RL, we propose *Capability-Aware Shared Hypernetworks (CASH)*, a new *soft weight sharing* architecture for heterogeneous coordination that use hypernetworks to explicitly reason about continuous agent capabilities in addition to local observations. Intuitively, CASH allows the team to learn *shared* decision making strategies (captured by a shared encoder) that are readily *adapted* according to the team’s individual and collective capabilities (by a shared hypernetwork). Our design is agnostic to the underlying learning paradigm. We conducted detailed experiments across two heterogeneous coordination tasks and three standard learning paradigms

(imitation learning, value-based and policy-gradient reinforcement learning). Results reveal that CASH generates appropriately diverse behaviors that consistently outperform baseline architectures in terms of task performance and sample efficiency during both training and zero-shot generalization. Notably, CASH provides these improvements with only 20% to 40% of the learnable parameters used by baselines.

# CHAPTER 1

## INTRODUCTION

Consider a wildfire fighting scenario wherein multiple fire departments gather their fire-fighting robots into a team. The team is heterogeneous – robots have different capabilities (e.g., speed, water-capacity, sensing radius, etc.) due to different makes, models, and customization. To combat growing and distributed fires, the robots must reason about and leverage their individual and collective capabilities to adopt dynamic and adaptive roles within the team. In such challenging scenarios, we often do not know which specific agents are available until runtime, and agent capabilities can change due to damages or reinforcements. As such, an effective approach needs to efficiently encode coordination strategies and readily generalize to team variations without requiring manual engineering or hand-tuning every time a new team is assembled.

In this work, we address the challenge of learning *flexible* and *diverse* coordination strategies that enable heterogeneous agents to adapt their behaviors based upon the individual and collective *capabilities* within the team. We are particularly interested in *efficiently* learning coordination strategies that can *generalize* to changes in team composition and capabilities.

Compared to homogeneous teams, learning coordination strategies for heterogeneous teams is more challenging. In a homogeneous team, agents can learn shared decision making strategies in which individual agents are interchangeable as they perform identical roles. In stark contrast, scenarios involving heterogeneous teams require agents to adopt diverse and flexible roles depending on the context and how their individual capabilities interact with that of their teammates and the environment. Revisiting our firefighting example, agents' individual and relative capabilities (speeds and water capacities) share an intricate relationship with the environments' characteristics (fire intensities and locations). The

team’s collective effectiveness crucially depends on its ability to reason about such relationships.

Inspired by prior literature in trait-based task allocation for heterogeneous teams [1], we represent the heterogeneity of agents with a capability vector for each one which defines its key traits (e.g. payload capacity, top speed). We refer to an extension of the observation space with this capability vector as capability-awareness. In Chapter 3, we investigate the utility of capability-awareness for decentralized multi-agent teams, via a novel GNN-based policy architecture trained with MAPPO [2]. We show that capability-awareness greatly improves the ability of policies to zero-shot generalize to unseen agents and teams.

Then, to improve the efficiency of capability-aware policies, in Chapter 4, we propose a novel architecture, named *Capability-aware Shared Hypernetwork (CASH)*, to encode individual policies or value functions (see Figure 4.1). CASH consists of a RNN-based Encoder, a Hyper Adapter, and an Adaptive Decoder. The encoder helps learn *agent-agnostic* coordination strategies that are shared across agents regardless of their capabilities. The Hyper Adapter learns to map the current observations and the agents’ capabilities to the weights of the Adaptive Decoder. As such, CASH can encode *diverse* coordination strategies within a *parsimonious* architecture that *flexibly* adapts to current capabilities and context. Further, CASH can be trained using any paradigm (e.g., imitation learning, value-based or policy-based reinforcement learning).

A key motivation for our use of hypernetworks in CASH is their recent success in other learning problems (e.g., transfer learning [3], meta-RL [4]). A hypernetwork is a neural network that generates weights for a target network, and are often referred to as dynamic networks that enable *soft* parameter sharing. Our core technical problem requires learned policies (value networks) to encode the mapping from observations *and* capabilities to actions (value estimates). One approach to accomplish this is by simply appending capability information to observations and utilizing a single shared network. Indeed, this embedding technique appears in works on multi-task generalization for single-agent reinforcement

learning [5], as well as generalization to unseen agents in multi-agent reinforcement learning [6]. In both cases, the additional embedding allows a single architecture to adaptively choose distinct functions for different contexts. However, hypernetworks have shown to be a far superior alternative both in terms of efficiency and effectiveness when contextualizing neural policies on such embedded features [7]. CASH can thus be seen as a soft parameter sharing architecture that is capable of encoding *appropriately* diverse behaviors within a heterogeneous team while still leveraging the benefits of a parameter sharing.

We evaluate the effectiveness of CASH across two heterogeneous coordination tasks (wildfire suppression and multi-material transport) and three learning paradigms (imitation learning, value-based RL, and policy-based RL). We compare CASH’s performance against that of two strong baselines that reflect contemporary architectures to accommodate heterogeneous capabilities: i) implicit learning of capabilities with recurrent networks, and ii) concatenating capabilities to observations. Our results demonstrate that CASH consistently learns appropriate levels of heterogeneous behaviors that outperform the baseline architectures in terms of task performance and sample efficiency, both during training and when generalizing to new teams with out-of-distribution capabilities. Notably, CASH provides these improvements while using only 20% to 40% of the learnable parameters used by baselines. Our work suggests capability awareness when combined with hypernetworks – which are relatively simple to implement with modern libraries – represent a powerful architectural design choice that practitioners of heterogeneous multi-agent learning should consider.

## **CHAPTER 2**

### **RELATED WORKS**

In this section, we characterize and discuss works that are competing, complementary, or motivational to our own.

#### **2.1 Learning for multi-robot teams**

Recent advances in applying deep learning to multi-agent systems have shown promise in circumventing the challenges associated with classical control of multi-robot systems. Multi-agent reinforcement learning (MARL), in particular, has been shown to be capable of solving a wide variety of tasks, including simple tasks in the multi-agent particle environments (MPE) [8], complex tasks under partial observability [9], coordinating an arbitrary number of agents in video games [10], and effective predictive modeling of multi-agent systems [11]. These approaches are driven by popular MARL algorithms like QMIX [12], MADDPG [8], and MAPPO [2] – nontrivial extensions of their single agent counterparts DQN [13], DDPG [14], and PPO [15], respectively. Centralized training, decentralized execution (CTDE) is a commonly used framework in which decentralized agents learn to take actions based on local observations while a centralized critic provides feedback based on global information [16, 17]. We use the CTDE paradigm as it lends itself naturally to multi-robot teams since observation and communication are often restricted.

#### **2.2 Heterogeneity within MARL**

Many MARL algorithms were originally designed for use in homogeneous multi-agent teams. However, truly homogeneous multi-robot teams are rare because of manufacturing differences, wear and tear, or task requirements. Most real-world multi-robot problems

such as search & rescue, agriculture, and surveillance require a diverse set of capabilities aggregated from heterogeneous robots [18, 19, 20]. Below we first discuss standard MARL algorithms, then discuss modifications prior works have made to these standard MARL algorithms to accurately capture heterogeneous teams.

### 2.2.1 Shared-Parameter Architectures

In MARL, one main design choice is whether or not to share the parameters and network architectures among agents. The main benefit to sharing parameters is that by using a single network for all agents, training is more sample-efficient as all agents effectively learn from one another’s experiences. Moreover, this is trivially possible under the assumption of CTDE training. However, without modification, sharing parameters unfortunately prohibits diverse agent behaviors and unique roles. While this limitation is immaterial in problems with homogeneous agents (e.g., formation control [21] and path planning [22, 23]), it renders classical shared-parameter architectures incompatible with heterogeneous multi-agent learning. In an effort to bring the benefits of shared-parameter architectures to heterogeneous coordination problems, prior work has developed a simple-but-effective design modification: append unique information about each individual agent (or type of agent) to the input [24, 6]. Typically, a *unique ID* is assigned to each agent and then appended to its observations before being passed to the shared-parameter architecture. While ID-based methods with full parameter sharing do tend to improve efficiency and scalability, recent work has demonstrated that they often fail to learn sufficient behavioral diversity and are not robust to noisy environments [25, 26].

### 2.2.2 Learning for Heterogeneous Teams

To overcome the limitations of shared-parameter architectures, *independent policy learning* [26] has recently emerged as a common approach to learn robust and diverse behaviors [27]. Naturally, this improvement comes at the cost of training efficiency, since in-

dependent learning involves significantly more learning parameters and does not allow for data reuse between agents.

One can view shared-parameter and independent learning as two ends of the same spectrum. Some prior work has explored a middle-ground approach in which only agents within a certain group (e.g., same class [28], same action space [29], same inferred role [25]) share parameters with one another, while there is no sharing between groups. We refer to these approaches as *selective* parameter sharing. Selective parameter sharing approaches are more efficient than independent architectures, while still being able to generate more diverse behaviors than shared-parameter architectures. However, they retain the downside of agent ID approaches in assuming that the same set of training classes will be retained in evaluation. Ideally, heterogeneous multi-agent policies would be able to generalize to unseen agents or teams.

Our proposed architecture is informed by existing architectures (ID-based, independent, and selective sharing), but establishes a new class of architectures for heterogeneous coordination: *soft* weight sharing. CASH shares the same encoder across agents to capture agent-agnostic but task-specific aspects of collaborating on a shared task, making it a parameter-shared approach. Simultaneously, it uses hypernetworks to flexibly adapt each agent’s decisions based on both context and capabilities. This combination affords CASH a number of unique benefits that are not shared by existing architectures. First, CASH can reason over agent capabilities defined in continuous spaces [1]. Second, CASH can use a *single* network to encode diverse behaviors, improving training efficiency. Third, CASH can generalize to variations in agent capabilities and team compositions as it learns to reason over a vector space of capabilities and does not treat each team as a separate problem instance.

## 2.3 Hypernetworks

Hypernetworks [30] are a class of neural architectures that tackle two common limitations of standard neural networks: i) lack of ability to adapt knowledge between different data contexts, and ii) performance drops due to distributional shifts. Our use of hypernetworks for flexible heterogeneous coordination is inspired by the fact that they can encode flexible decision making strategies that dynamically adapt to contextual information. They also offer a number of benefits. In particular, recent findings suggest that hypernetworks can encode optimal policies for multiple tasks within a single architecture [31, 32], improve learning efficiency by improving gradient estimation in Q-learning [33], and achieve better performance with significantly less number of learnable parameters [7, 34]. As our experiments reveal, these benefits of hypernetworks translate to heterogeneous coordination.

### 2.3.1 Hypernetworks for multi-agent learning

There is one notable use of hypernetworks in multi-agent learning: QMIX [35], a popular off-policy MARL algorithm. QMIX uses hypernetworks within the value-mixing network to process global state information in order to improve decentralized coordination via centralized training. In contrast, our novelty is to use hypernetworks to flexibly determine parameters within *individual* agents' policy or value networks, enabling a single architecture to encode diverse behaviors. To the best of our knowledge, no prior work in multi-agent coordination has employed hypernetworks in a similar fashion. Further, CASH and QMIX are compatible with one another, and this combination leads to significant improvements as we show in our results.

## CHAPTER 3

### CAPABILITY-AWARE POLICY ARCHITECTURES

In this chapter, we explore GNN-based methods for heterogeneous multi-agent teaming, with a specific focus on robotics applications and an exploration of the importance of *capability-awareness*. Our key insight, inspired by prior literature in trait-based task allocation for heterogeneous teams [1], is that adaptive teaming requires the understanding of how a team’s diverse capabilities combine to dictate the behavior of individual robots. For instance, consider an autonomous heterogeneous team responding to multiple concurrent wildfires. Effective coordination in such situations requires reasoning about the opportunities and constraints introduced by the robots’ individual and relative capabilities, such as speed, water capacity, and battery range. In general, robots must learn how their individual capabilities relate to those of others to determine their role in achieving shared objectives. We make the assumption that the robots are aware of their individual capabilities and must share them for effective coordination, and to this end, we develop a novel GNN-based policy architecture as detailed below.

#### 3.1 Capability Awareness and Communication for Adaptive Teaming

In this section, we first model heterogeneous teams and then introduce policy architectures that enable capability awareness and communication, along with the associated training pipeline.

##### 3.1.1 Modeling Heterogeneous Multi-Robot Teams

We model teams of  $N$  heterogeneous robots as a graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , where each node  $v_i \in \mathcal{V}$  is a robot, and each edge  $e_{ij} = (v_i, v_j) \in \mathcal{E}$  is a communication link. We use  $z_i$  to denote the observations of the  $i$ th robot, which includes its capabilities and its sensor

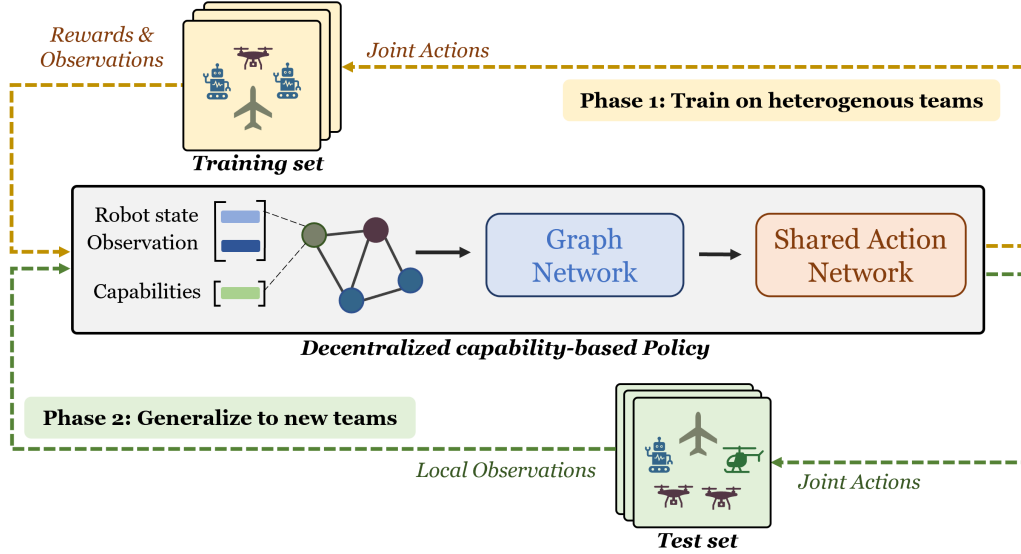


Figure 3.1: We investigate the role of capability awareness and communication in generalizing decentralized heterogeneous multi-robot coordination policies to teams of new composition, size, and robots.

readings of the environment. We assume that the robots’ heterogeneity can be captured by their capabilities. We represent the capabilities of the  $i$ th robot by a real-valued vector  $c_i \in \mathcal{C} \subseteq R_+^C$ , where  $\mathcal{C}$  is the  $C$ -dimensional space of all capabilities of the robots. An example of a multi-dimensional capability is a vector with elements representing payload, speed, and sensing radius. When robot  $i$  does not possess the  $k$ th capability, the  $k$ th element of  $c_i$  is set to zero.

### 3.1.2 Problem Description

We are interested in learning a decentralized control policy that can i) effectively coordinate a team of heterogeneous robots to achieve the task objectives, and ii) generalize readily to teams of both novel compositions and novel robots that are not encountered during training. Our problem can be viewed as a multi-agent reinforcement learning problem that can be formalized as a decentralized partially-observable Markov Decision Process (Dec-POMDP) [36]. We expand on the Dec-POMDP formulation to incorporate the *capabilities* of heterogeneous robots and arrive at the tuple  $\langle \mathcal{D}, \mathcal{S}, \{\mathcal{A}_i\}, \{\mathcal{Z}_i\}, \mathcal{C}, T, R, O \rangle$  where  $\mathcal{D}$  is

the set of  $N$  robots,  $\mathcal{S}$  is the set of global states,  $\{\mathcal{A}_i\}$  is a set of action spaces across all robots,  $\{\mathcal{Z}_i\}$  is a set of joint observations across all robots,  $\mathcal{C}$  is the multi-dimensional space of capabilities,  $R$  is the global reward function, and  $T$  and  $O$  are the joint state transition and observation models, respectively. Our objective is to learn decentralized action policies that control each robot to maximize the expected return  $E[\sum_{t=0}^{T_h} r_t]$  over the task horizon  $T_h$ . The decentralized policy of the  $i$ th robot  $\pi(a_i|\tilde{o}_i)$  defines the probability that Robot  $i$  takes Action  $a_i$  given its effective observation  $\tilde{o}_i$ . The effective observation  $\tilde{o}_i$  of the  $i$ th robot is a function of both its individual observation and that of others in its neighborhood.

### 3.1.3 Policy Architecture

To enable capability awareness and communication in multi-robot coordination policies, we designed a policy architecture that leverages graph convolutional networks (GCNs) since recent approaches attest to their ability to learn effective communication protocols and enable decentralized decision-making in multi-agent teams [37]. Further, operations are local to nodes and can therefore generalize to graphs of any topology (i.e., permutation invariance [38]). We illustrate our architecture in Figure 3.1 and explain its components below.

### 3.1.4 Graph Neural Networks

We employ a graph convolutional network (GCN) architecture for the decentralized policy  $\pi_i$ , which enables robots to communicate for coordination according to the robot communication graph  $\mathcal{G}$ .

A GCN is composed of  $L$  layers of graph convolutions, followed by non-linearity. In this work, we consider a single graph convolution layer applied to node  $i$  is given by

$$h_i^{(l)} = \sigma \left( \sum_{j \in \mathcal{N}(i) \cup i} \phi_{\theta}(h_j^{(l-1)}) \right)$$

where  $h_j^{(l-1)} \in R^F$  is the node feature of node  $j$ ,  $\mathcal{N}(i) = \{j | (v_i, v_j) \in \mathcal{E}\}$  are all nodes  $j$  connected to  $i$ ,  $\phi_\theta$  is node feature transformation function with parameters  $\theta$ ,  $\sigma$  is a non-linearity (e.g. Relu), and  $h_i^l \in R^G$  is the output node feature.

### 3.1.5 Policy Architectures

Each of the graph neural networks in the GNN-based policy architectures evaluated are composed of an input encoder network, a message passing network, and an action output network. The encoder network is a 2-layer MLP with hidden dimensions of size 64. For the message passing network, a single graph convolution layer composed of 2-layer MLPs with ReLU non-linear activations. The action output network is additionally a 2-layer MLP with hidden dimensions of size 64. The learning rate is 0.005.

MLP (ID) /MLP (CA) : The MLP architectures compose of a 4-layer multi-layer perceptron with 64 hidden units at each layer and ReLU non-linearities.

CA (GNN) /CA+CC (GNN) /ID (GNN) : Each of the graph neural networks compose of an input “encoder” network, a message passing network, and an action output network. The encoder network and the action output network are multi-layer perceptrons with hidden layers of size 64, ReLU non-linear activations, and with one and two hidden layers respectively. The message passing network is a graph convolution layer wherein the linear transformation of node features (i.e. observations) is done by a 2-layer MLP with ReLU non-linear activations and 64 dimensional hidden units, followed by a summation of the transformed neighboring node features. The output node features are concatenated with the output feature from the encoder network. This concatenated features is the input to the two out action network. The CA (GNN) network doesn’t communicate the robot’s capabilities with the graph convolution layers. Rather, the capabilities are appended to the output of the encoder network and output of node features of the graph convolution layer just before the the action network. Thus, the the action network is the only part of this model that is conditioned on robot capabilities.

### 3.1.6 Policy Training Hyperparameters

We detail the hyperparameters used to train each of the policies using MAPPO [39], which is a multi-agent extension of Proximal Policy Optimization (PPO) [40] in Table 3.1.

Table 3.1: Hyperparameters used to train each of the policies with PPO.

Hyperparameter	Value
Action Selection (Training)	soft action selection
Action Selection (Testing)	hard action selection
Critic Network Update Interval	200 steps
Learning Rate	0.0005
Entropy Coefficient	0.01
Epochs	4
Clip	0.2
Q Function Steps	5
Buffer Length	64
Number of training steps	$40 \times 10^6$ (HMT), $20 \times 10^6$ (HSN)

**Capability awareness:** We argue that the heterogeneity of robots can have a significant impact on how the team must coordinate to achieve a task. Specifically, individual and collective capabilities can affect the roles the robots play within the team. For instance, consider a heterogeneous mobile robot team responding to wildfire incidents at multiple locations. To effectively respond, a robot within the team must account for its speed and water capacity. Therefore, it is necessary that robots are aware of their capabilities. To enable such awareness, we append each robot’s capability vector  $c_i$  to its observations before passing them along as node features to the graph network. This information will help each robot condition its actions not just on observations, but also on its capabilities.

**Capability communication:** In addition to awareness, communicating capability information can enable a team to reason about how its collective capabilities impact task performance. Revisiting our wildfire example, robots in the response team can effectively coordinate their efforts by implicitly and dynamically taking on roles based on their relative speed and water capacity. But such complex decision making is only possible if robots communicate with each other about their capabilities. Each node in our GCN-based policy

receives capability information along with the corresponding robot’s local observations so the learned communication protocol can help the team communicate and effectively build representations of their collective capabilities. We do not enforce limits on how the GCN learns to communicate; that is, the learned communication latents are uninterpretable.

Note that our policy is robot-agnostic and learns the implicit and interconnected relationships between the observations, capabilities, and actions of all robots in the team. Further, as we demonstrate in our experiments, capability awareness and communication enables generalization to teams with new robot compositions, sizes, and even to entirely new robots as long as their capabilities belong to the same space of capabilities  $\mathcal{C}$ . An overview of the policy architecture is shown in Figure 3.1.

### 3.1.7 Training Procedure

We employ a CTDE paradigm to train the parameter-shared action policy for each agent. Parameter sharing is required for our problem so policies can transfer to new robots without the need for training new policies or assigning robots to already trained policies, allowing us to accurately assess generalization. We apply an actor-critic model, and train using MAPPO. The actor-critic model is composed of a decentralized actor network (i.e., shared action policy) that maps robots observations to control actions, and a centralized critic network [2], which estimates the value of the team’s current state based on centralized information about the environment and robots aggregated from individual observations. Finally, we trained our policies on multiple teams until they converged, with the teams changing every 10 episodes to stabilize training.

## **3.2 Experimental Design**

We conducted detailed experiments to evaluate how capability awareness and communication impact generalization to: i) new team sizes and compositions, and ii) new robots with unseen capabilities.

**Environments:** We designed two heterogeneous multi-robot tasks for experimentation:

- **Heterogeneous Material Transport (HMT):** A team of robots with different material carrying capacities for lumber and concrete (denoted by  $c_i \in R^2$  for the  $i$ th robot) must transport materials from lumber and concrete depots to a construction site to fulfill a pre-specified quota while minimizing over-provision. We implemented this environment as a Multi-Particle Environment (MPE) [41] and leverage the infrastructure of EPyMARL [42].
- **Heterogeneous Sensor Network (HSN):** A robot team must form a single fully-connected sensor network while maximizing the collective coverage area. The  $i$ th robot’s capability  $c_i \in R$  corresponds to its sensing radius. We implemented this environment using the MARBLER [43] framework which enables hardware experimentation in the Robotarium [44], a well-established multi-robot test bed.

**Policy architectures:** In order to systematically examine the impact of capability awareness and communication, we consider the following policy architectures:

- **ID (MLP):** Robot ID-based MLP
- **ID (GNN):** Robot ID-based GNN
- **CA (MLP):** Capability-aware MLP
- **CA (GNN):** Capability-aware GNN without communication of capabilities,
- **CA+CC (GNN):** Both capability awareness and communication.

The ID-based baselines stand in for SOTA approaches that employ behavioral typing to handle heterogeneous teams [45, 46], and, as such, question the need for capabilities. The MLP based baselines help us investigate the need for communication. Finally, the CA (GNN) enables communication of observations but does not communicate capability information.

**Metrics:** For both environments, we compare the above policies using *Average Return*: the average joint reward received over the task horizon (higher is better). Additionally, we use environment-specific metrics. In HMT, we terminate the episodes when the quotas for both materials are met. Therefore, we consider *Average Steps* taken to meet the quota (lower is better). For HSN, we consider *Pairwise Overlap*: sum of pairwise overlapping area of robots’ coverage areas (lower is better).

**Training:** For each environment, we used five teams with four robots each during training. We selected the training teams to ensure diverse compositions and degree of heterogeneity. For HSN, we sampled robots’ sensing radius from the uniform distribution  $U(0.2, 0.6)$ . For HMT, we sampled robots’ lumber and concrete carrying capacities from the uniform distribution  $U(0, 1.0)$ . We also assigned each robot a one-hot ID to train ID-based policies. We trained each policy with 3 random seeds. We resampled robot teams every 10 episodes to stabilize training.

### 3.3 Results

Below, we report i) performance on the training team, ii) zero-shot generalization to new teams, and iii) zero-shot generalization to new robots with unseen values of capabilities for each environment.

#### 3.3.1 Heterogeneous Material Transport

We first focus on the Heterogeneous Material Transport (HMT) environment.

**Performance on training set:** To ensure considering capabilities does not negatively impact training, we first evaluate trained policies on the training set in terms of average return and average steps (see Figure 3.2). We find that all policies resulted in comparable average returns (Figure 3.2 (a)). However, the average number of steps per episode better captures performance in HMT, since episodes terminate early when the quota is filled. Compared to agent-ID policies, capability-based policies took fewer steps per episode (Fig-

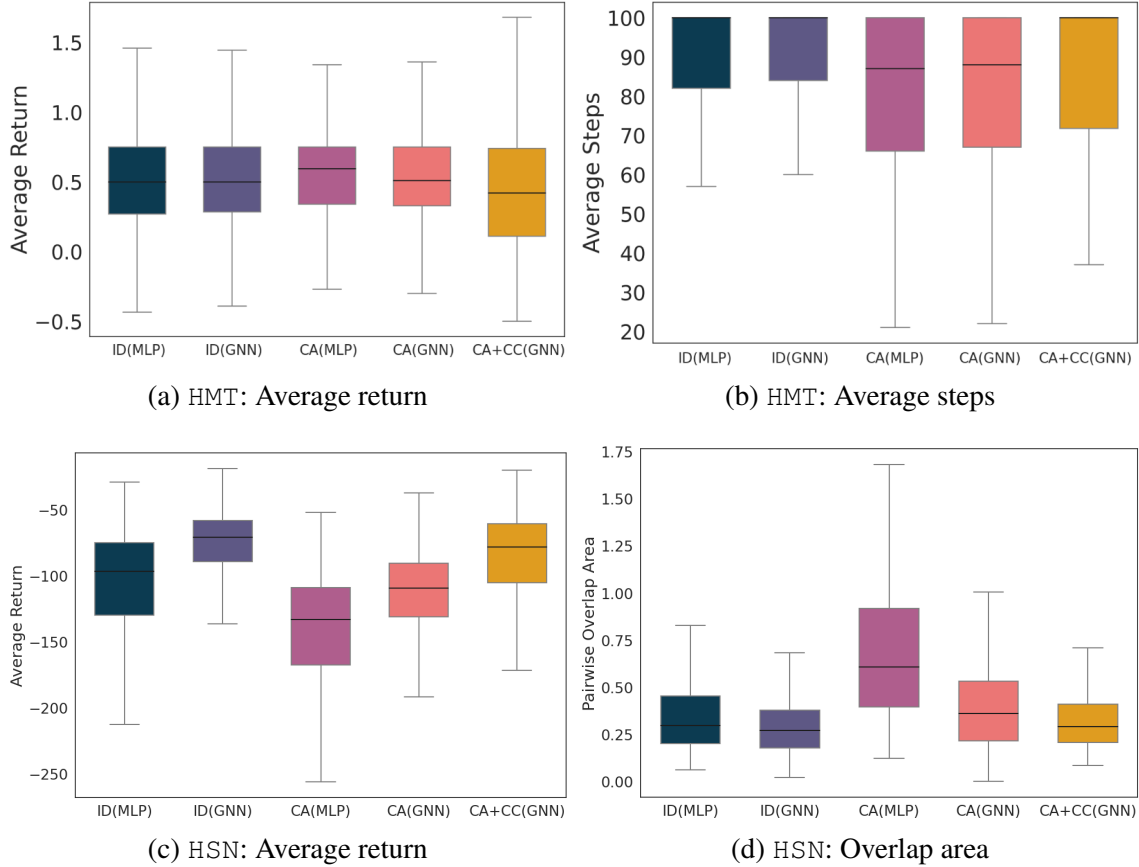


Figure 3.2: When evaluated on *teams seen during training*, capability-aware policies performed comparably to ID-based policies in terms of both average return (higher is better) and task-specific metrics (lower is better).

ure 3.2 (b)) to achieve comparable rewards, suggesting that reasoning about capabilities can improve task efficiency performance.

**Zero-shot generalization to new team compositions and sizes:** We next evaluated how trained policies generalize to team compositions and sizes not encountered during training. Figure 3.3 shows plots quantifying performance on new compositions with team sizes of 3, 4, and 5 robots. To ensure a fair comparison, we evaluated all policies on the same set of 100 teams by randomly sampling novel combinations of robots from the training set. We evaluated each policy on each test team across 10 episodes per seed. Given that this evaluation involved no new individual robots, we reused each robot’s ID from the training set to facilitate the evaluation of ID-based policies.

We find that all capability-aware methods outperformed ID-based methods in return and task-specific metrics. This is likely due to capability-aware methods’ ability to capture the relationship between robots’ carrying capacities and the material quota. In contrast, ID-based methods must learn to implicitly reason about how much material each robot can carry. Interestingly,  $CA(MLP)$  resulted in fewer steps per episode (lowest mean and variance) across all team sizes, and outperformed both the other capability-based and communication-enabled baselines:  $CA(GNN)$  and  $CA+CC(GNN)$ . This suggests that mere awareness of capabilities is sufficient to perform well in the HMT environment. Indeed, communication is not as essential in this task as robots can directly observe relevant information (e.g. material demands) and implicitly coordinate as long as they are aware of their own capabilities. It might be possible to further improve performance by learning to better communicate, but that would be significantly more challenging since the task can be mostly solved without communication.

**Zero-shot generalization to new robots:** In Figure 3.4, we show the policies’ ability to generalize to teams composed of new robots. Since the robots’ capabilities in this evaluation are different from those of the robots in the training set, we could not evaluate agent-ID methods since there is no trivial way to assign IDs to the new robots. These results clearly demonstrate that reasoning about capabilities can enable generalization to teams with entirely new robots. Further, we again see that capability awareness without communication is sufficient to generalize in the HMT environment.

### 3.3.2 Heterogeneous Sensor Network

Below, we discuss results on the Heterogeneous Sensor Network (HSN) environment.

**Performance on training set:** In Figure 3.2 (c) and (d), we report the performance of trained policies in HSN on teams in the training set in terms of average return and pairwise overlap. All policies except  $CA(MLP)$  performed comparably and were able to effectively learn to maximize expected returns, achieve a fully connected sensor network, and

minimize the pairwise overlap in coverage area. Further, CA+CC (GNN) and ID (GNN) perform similarly but marginally better than the other baselines. CA (MLP) ’s suboptimal performance indicates that capability awareness in isolation without any communication hurts performance in the HSN task. Indeed, while it is possible to achieve good performance in HMT without communicating, HSN requires robots to effectively communicate and reason about their neighbors’ sensing radii in order to form effective networks.

Taken together, these results suggest that capability awareness and communication can lead to effective training in heterogeneous teams. ID-based methods are able to perform at a similar level. This is to be expected given that we conducted these evaluations on the training set and IDs are sufficient to implicitly assign roles and coordinate heterogeneous robots within known teams.

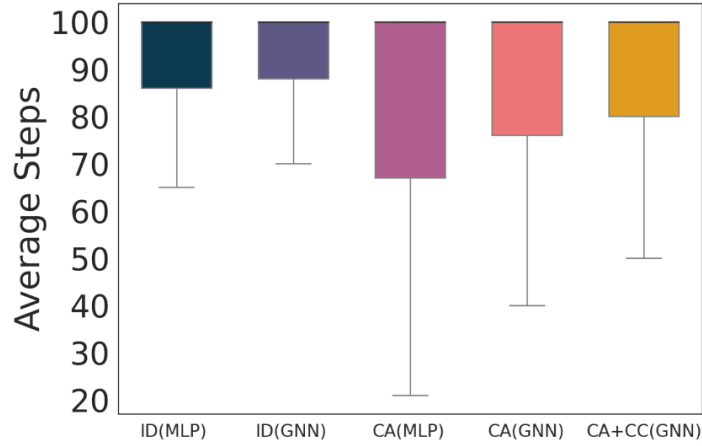
**Zero-shot generalization to new team compositions and sizes:** In Figure 3.5, we report the performance of the training policies in HSN when evaluated on teams of different compositions and sizes. We found that CA+CC (GNN) achieved the best average returns (highest mean and lowest variance) across all team sizes. Both ID-based methods (ID (MLP) and ID (GNN)) resulted in lower returns compared to all three capability-awareness baselines. Note that this is in stark contrast to the results for the training set in Figure 3.2, demonstrating that IDs alone might help train heterogeneous teams but tend to generalize poorly to new team compositions and sizes. This is likely because ID-based policies fail to reason about robot heterogeneity, and instead overfit the relationships between robot IDs and behavior in the training set. Further, CA+CC (GNN) in particular consistently outperformed all other policies across metrics and variations, suggesting that both capability awareness and communication are necessary to enable generalization in HSN.

**Zero-shot generalization to new robots:** We evaluated the trained policies’ ability to generalize to teams of different sizes which are composed of entirely new robots whose sensing radii are different from those encountered in training. Similar to HMT, we cannot evaluate ID-based policies on teams with new robots since there is no obvious way

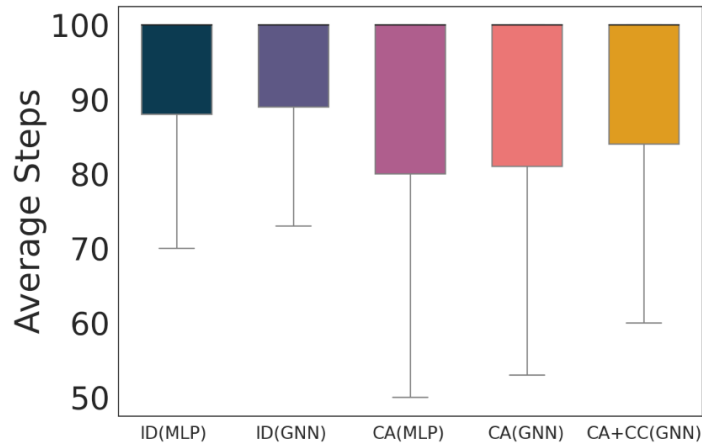
to assign IDs to the new robots. In Figure 3.6, we report the performance of all three capability-based policies in terms of average return. Both GNN-based policies ( $CA (GNN)$  and  $CA+CC (GNN)$ ) considerably outperform the  $CA (MLP)$  policy, underscoring the importance of communication in generalization to teams with new robots. However, we also see that communication of observations alone is insufficient, as evidenced by the fact that  $CA+CC (GNN)$  (which communicates both observations and capabilities) consistently outperforms  $CA (GNN)$  (which only communicates observations).

### 3.4 Conclusion

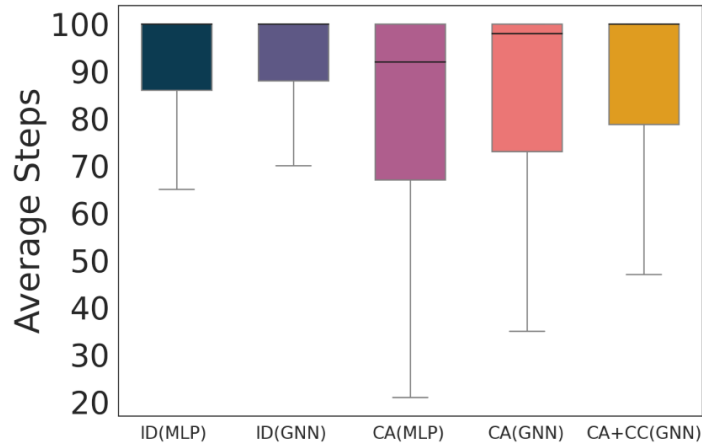
Our work here shows the utility of awareness and communication of robot capabilities to allow multi-robot policies to generalize effectively to new teams. To effectively account for and communicate capabilities, we developed a GNN-based decentralized, shared-parameter policy architecture that allows individual robots to reason about and communicate their observations and capabilities to achieve adaptive teaming. Our detailed experiments with two heterogeneous multi-robot tasks illustrate the importance and the need for reasoning about capabilities as opposed to merely agent IDs.



(a) 3 Robots

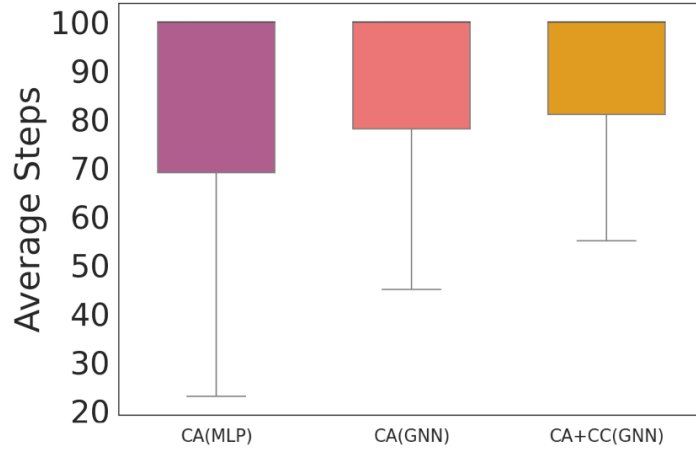


(b) 4 Robots

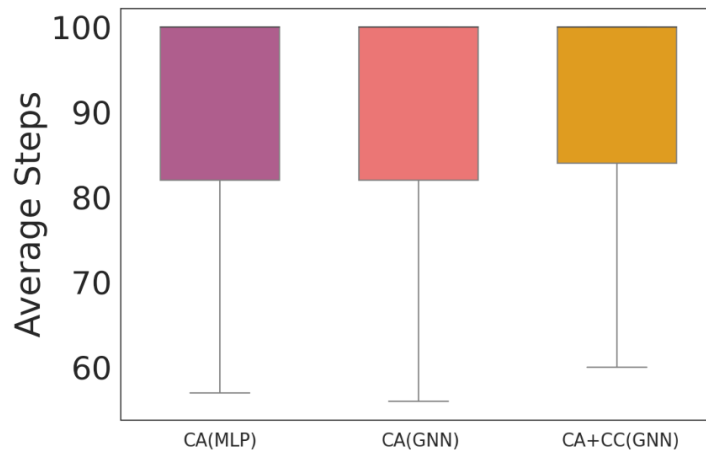


(c) 5 Robots

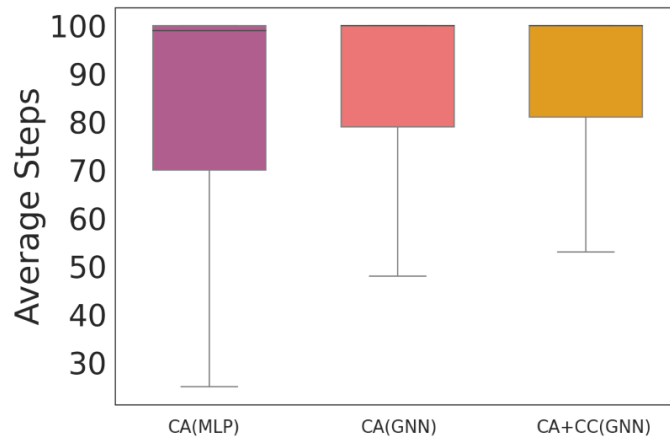
Figure 3.3: When generalizing to *new team compositions and sizes* in HMT, capability-based policies consistently outperformed ID-based policies in terms of average steps taken to meet the quota (lower is better).



(a) 3 robots

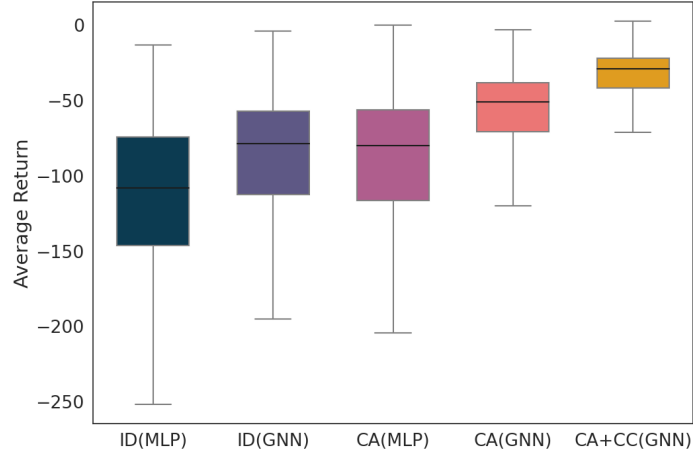


(b) 4 robots

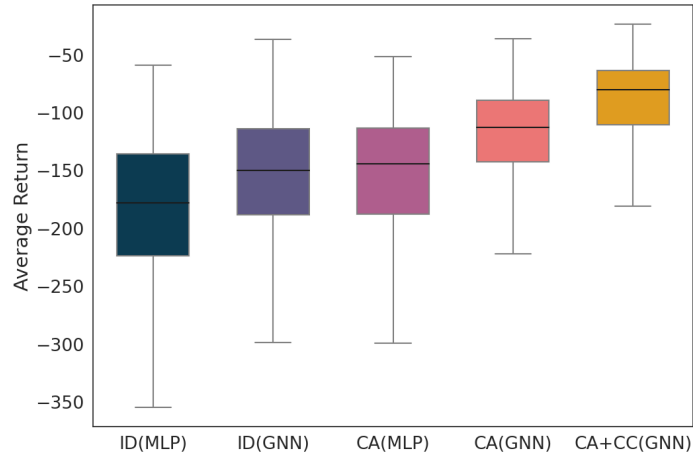


(c) 5 robots

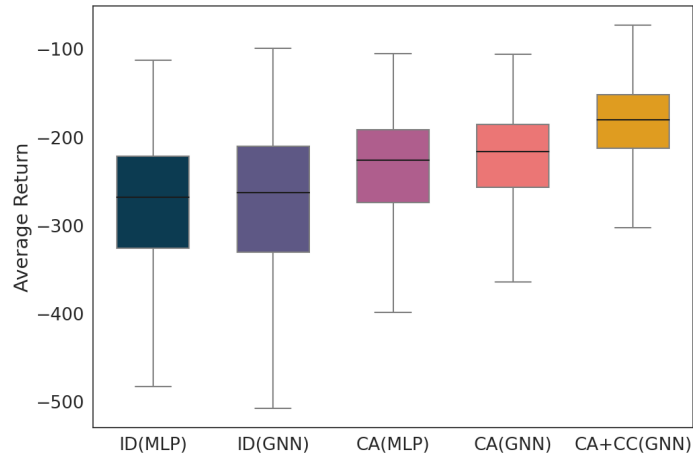
Figure 3.4: When generalizing to *new robots with unseen values for capabilities* in HMT, policies that are only aware of capabilities (CA (MLP) and CA (GNN) ) outperformed policies that also communicated capabilities (CA+CC (GNN) ) in terms of average number of steps taken to transport the required material (lower is better).



(a) 3 Robots

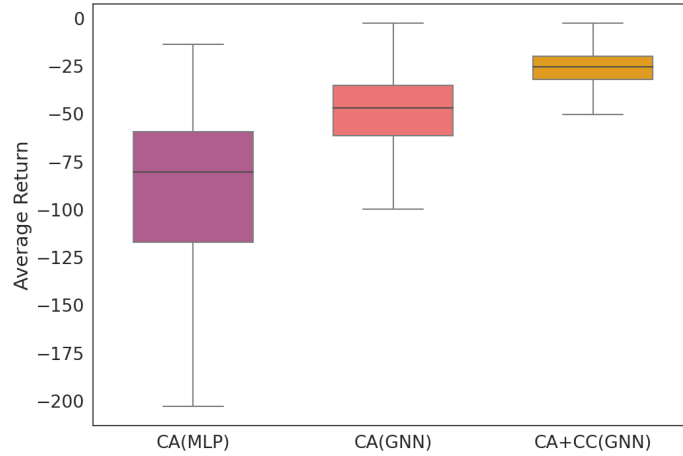


(b) 4 Robots

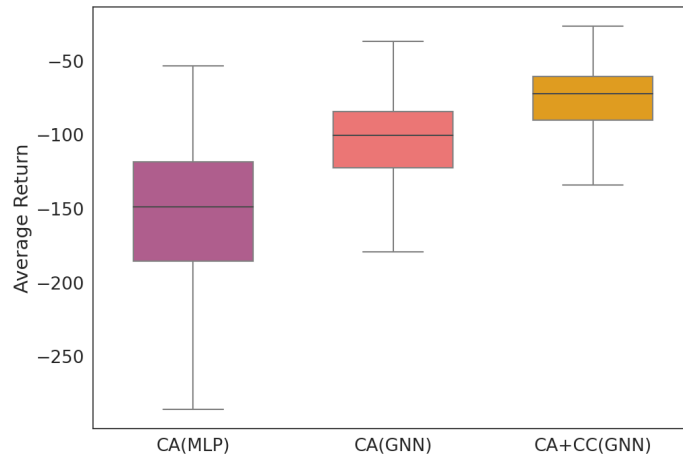


(c) 5 Robots

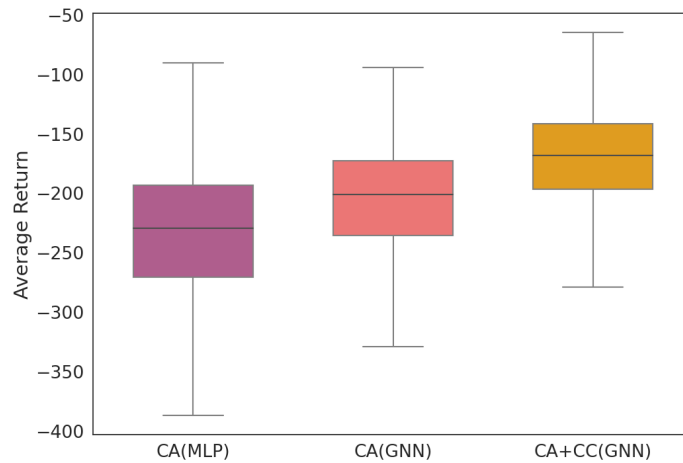
Figure 3.5: When generalizing to *new team compositions and sizes* in HSN, capability-based policies consistently outperformed ID-based baselines in terms of average return (higher is better). Further, combining awareness and communication of capabilities resulted in the best generalization performance.



(a) 3 Robots



(b) 4 Robots



(c) 5 Robots

Figure 3.6: When generalizing to *teams comprised of new robots* in HSN, combining awareness and communication of capabilities (CA+CC (GNN) ) achieves higher average returns than baselines that are merely aware of capabilities, irrespective of whether they communicate observations (CA (GNN) ) or not (CA (MLP) ).

## CHAPTER 4

### CAPABILITY-AWARE SHARED HYPERNETWORKS

In this chapter, we retain the heterogeneous DEC-POMDP problem formulation from Chapter 3, and propose an improved hypernetwork-based policy architecture. This presents a novel soft weight-sharing solution to the problem, which we show experimentally both retains sufficient diversity for task success while remaining sample- and parameter- efficient, and generalizing to unseen agents and teams. We make one simplifying assumption from the previous chapter, which is that each decentralized agent now has not only its own capabilities as a vector, but also the capabilities of its teammates, as well as their positions in the observation vector (defined as relative to ego-agent). As such, we do not include a GNN component in the CASH architecture; future work could explore the combination of the work from the preceding chapter and this one. Our main improvement on the preceding chapter’s policy architecture is that we explicitly design CASH to be more parameter- and sample-efficient, while also conducting an explicit investigation into how shared-parameter policies can maintain high behavioral diversity with our soft weight-sharing approach.

#### 4.1 Proposed Architecture

We present Capability-Aware Shared Hypernetworks (CASH), a new class of decentralized shared-parameter architectures that enable flexible heterogeneous coordination. As shown in Figure 4.1, CASH contains three separate neural network modules: i) RNN Encoder, ii) Adaptive (action) Decoder and iii) Hyper Adapter.

##### 4.1.1 RNN Encoder

CASH employs a shared-parameter GRU-based recurrent network [47] to process agents’ local observations. We use a gated recurrent network to encode observations since they

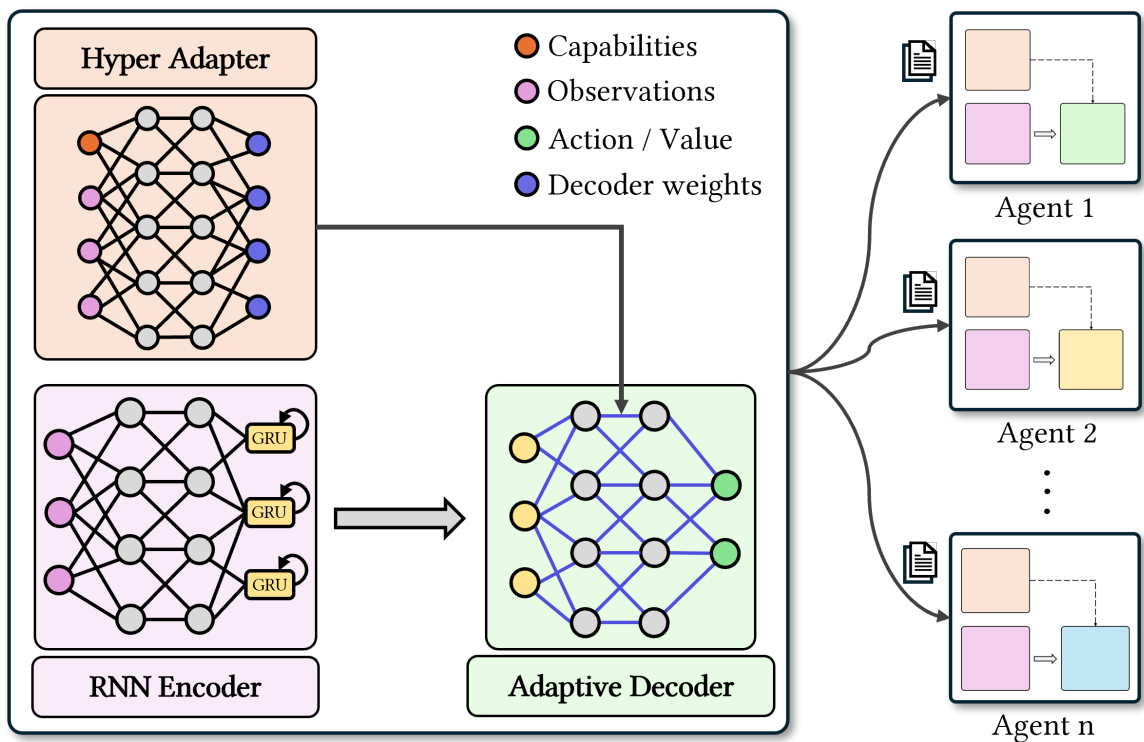


Figure 4.1: We present Capability-Aware Shared Hypernetworks (CASH), an architecture for flexible and decentralized heterogeneous multi-agent teaming. CASH leverages “soft parameter sharing” using hypernetworks to condition agent decision making on individual capabilities, collective capabilities, and observations.

have been shown to be particularly effective in challenging environments that involve long time horizons and partial observability [48, 35]. Recurrent networks are commonly used in multi-agent settings to i) keep a history of agent experiences and ii) learn a state transition model for the environment. The RNN Encoder within CASH can be denoted using  $g_\sigma : \mathcal{Z}_i \rightarrow \mathcal{Y}$ , where  $\{\mathcal{Z}_i\}$  is the joint space of all agents’ observations and  $\mathcal{Y}$  is the latent embedding space. CASH provides the embedding produced by the RNN Encoder as input to the Adaptive Decoder.

#### 4.1.2 Adaptive Decoder

In many problems involving heterogeneous teams, each agent must adapt its behavior based on its own capabilities as well as those of others. We accomplish this in CASH by ensuring that the network module that generates each agent’s final output (referred to as the *Adaptive Decoder*) is controlled by a hypernetwork conditioned on the agent’s own capability vector  $c_i$  and of its teammates  $\mathcal{C}_{/i} = \{c_j | j \neq i\}$ , as well as local observations  $o_i(t)$ . This design allows CASH to encode flexible decision making strategies that vary across heterogeneous agents within a single network. However, decision making that adapts to capabilities alone might not be sufficient to achieve the necessary flexibility. Agents might also need to consider the current state of the environment in conjunction with capabilities. Thus, we also ensure that the Adaptive Decoder reasons about the current observation of the agent.

#### 4.1.3 Hyper Adapter

To achieve sufficient flexibility and dynamism with the Adaptive Decoder, we leverage a hypernetwork  $h_\phi(\cdot)$  to generate the neural network weights  $\theta_i(t)$  of the  $i$ th agent’s Adaptive Decoder  $f_{\theta_i(t)}$  at each time step  $t$ :  $\theta_i(t) = h_\phi(c_i, \mathcal{C}_{/i}, o_i(t))$ , where  $o_i(t)$  the  $i$ th agent’s observations at time  $t$ . One way to view the relationship between Hyper Adapter and Adaptive Decoder is to consider the Adaptive Decoder as a parameterized distribution over actions conditioned on local observations:  $f_{\theta_i(t)} = p(a_i(t) | o_i(t); \theta_i(t))$ , where  $a_i(t)$  is the action

output of the  $i$ th agent at time  $t$ . Note how this conditional distribution dynamically varies as a function of both capabilities and observations owing to the fact that its parameters are dictated by a hypernetwork conditioned on the same variables. This enables “soft” parameter sharing [49] in CASH. The Adaptive Decoder can take on different parameters according to both agents and observations despite the fact that all predetermined parameters of CASH are shared among all agents.

We find empirically that a deeper four-layer hypernetwork is better at simultaneously reasoning over capabilities and observations. However, consistent with prior work using hypernetworks in RL [4], we found it difficult to optimize a standard four-layer hypernetwork, both in reinforcement and imitation learning regimes. We believe this is due to the combination of instabilities inherent in the learning paradigms as well as hypernetworks. Fortunately, Layer Normalization [50] has been shown recently to improve training stability for deeper networks across a wide range of reinforcement learning domains [51, 52]. Inspired by these findings, we add LayerNorm before each ReLU activations in our hypernetwork and find that it stabilizes learning and substantially improves performance.

## 4.2 Experimental Procedure

Our experimental procedure compares CASH to two other strong baselines trained with three common multi-agent learning paradigms, across two cooperative tasks with heterogeneous agents. We aim to show that CASH effectively grounds agents in their capabilities, which improves generalization to new agents and teams as well as substantial sample-efficiency improvements during training.

### 4.2.1 Learning Paradigms

We evaluate CASH on three standard approaches for multi-agent learning, to demonstrate that the benefits of CASH translate across multiple learning paradigms.

*QMIX*: a popular value-based multi-agent reinforcement learning algorithm where de-

centralized agent networks estimate the values of state/action pairs for each agent. These estimates are then aggregated to estimate the overall team’s value with a hypernetwork-controlled central network [35].

*MAPPO*: a popular policy gradient approach to multi-agent reinforcement learning where individual agents are controlled by separate policies, but the policy gradient update step is regularized by a shared centralized critic [48].

*Dagger*: an imitation-learning algorithm that improves standard Behavioral Cloning by adding expert demonstrations during training to mitigate distributional shift [53]. We extend *Dagger* to the multi-agent case by using a single shared-parameter policy across all agents and giving the expert full state information to select the optimal action for each agent at each timestep. For *MAPPO* and *QMIX*, we adapt *JaxMARL*’s existing implementations [54]. We implemented *Dagger* from scratch for *JaxMARL* based on the existing *MARL* implementations.

#### 4.2.2 Heterogeneous Multi-Agent Tasks

We evaluate our method on two heterogeneous cooperative tasks. Both environments are implemented with *JaxMARL*’s variant of the Multi-Agent Particle Environment [54, 55].

*Firefighting*: A team of three agents spawn in a fire depot and aim to put out two fires. Agents have varying speed levels and water capacity, and fires spawn with varying sizes. As a simplification of true firefighting, fires are fixed in space and do not grow over time, reducing the problem to effective allocation of agents to fires given the distances and sizes. Task success is defined as both fires being successfully extinguished within the time limit. The team is rewarded when fires are successfully extinguished and penalized at a rate proportional to the severity of active fires for each active fire. Each agent is also given a small shaping reward to encourage them to reduce their distance to any fire at the start of training. For an example of this environment, see Figure 4.2.

*Transport*: A team of four agents transport construction materials between two ma-

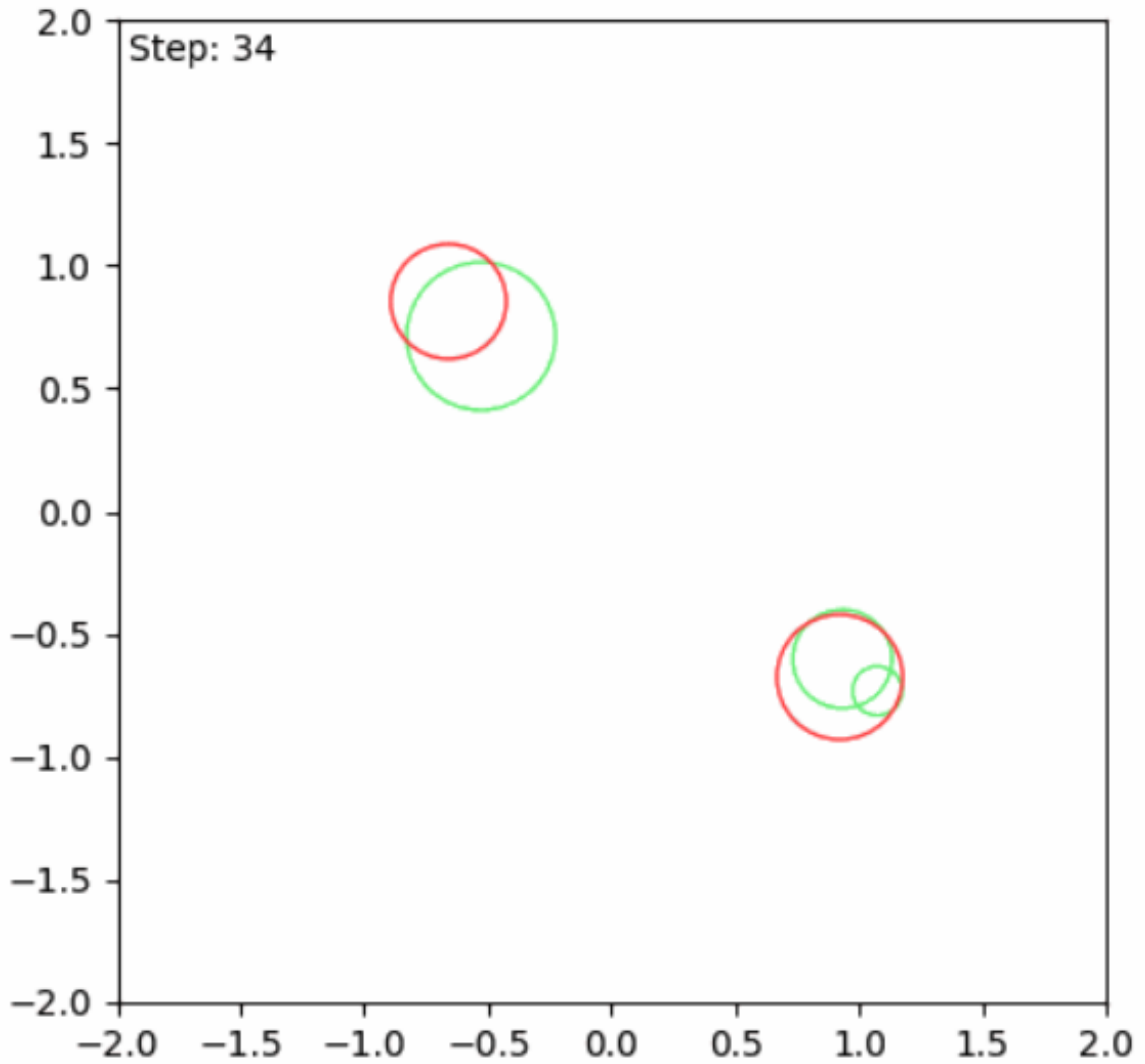


Figure 4.2: Example still from the simulated `Firefighting` environment with three agents (green circles) and two fires (red circles). Agents aim to effectively allocate themselves to fires within a fixed time horizon. Radius of agent and fire circles represent firefighting capacity and fire severity, respectively, and a fire is considered extinguished if the sum of all agent firefighting capacity on a given fire is greater than the severity of said fire.

terial depots back to a shared construction site. Agents vary in their carrying capacity for each type of material. The agents must fulfill a specific quota of material requested by the construction site, without delivering too much or too little of each material. Agents succeed if they meet the quota within the time limit, without exceeding it. We adapt this task from the HMT task in the prior work on capability-awareness in multi-agent teams described in the preceding chapter [6]. Agents are rewarded for picking up a material when they have capacity and for dropping off material in the depot if it contributes productively to the quota (i.e. does not exceed the quota). For an example of our implementation of this environment, see Figure 4.3.

**Metrics:** For both tasks, we use the following metrics:

- *Training returns* ( $\uparrow$ ), the sum of all rewards collected during training episodes.
- *Success rate* ( $\uparrow$ ), where success is as defined for each task above.
- *SND* ( $\updownarrow$ ), a metric developed to quantify the behavioral heterogeneity of independent policies [56]. We slightly modify SND to accommodate shared-parameter policies conditioned on capabilities (see Appendix for details).

In addition to the above, we define one more task-specific metric for each task. In *Firefighting*, we report the *percentage of fires extinguished* across all episodes. In *Transport*, we report the *makespan* (number of timesteps taken to fulfill the quota).

### 4.2.3 Baseline Architectures

In all experiments, we compare CASH against two baseline architectures that reflect current standard practices to handle agent heterogeneity while sharing parameters. A common architecture in MARL is a single linear layer, followed by a GRU cell whose latent embedding is decoded by an MLP into actions or values (e.g., [35, 48]). In our implementation of the standard RNN architecture, we keep the encoding linear layer and GRU identical across learning paradigms and modify only the action decoder. For MAPPO, our decoder

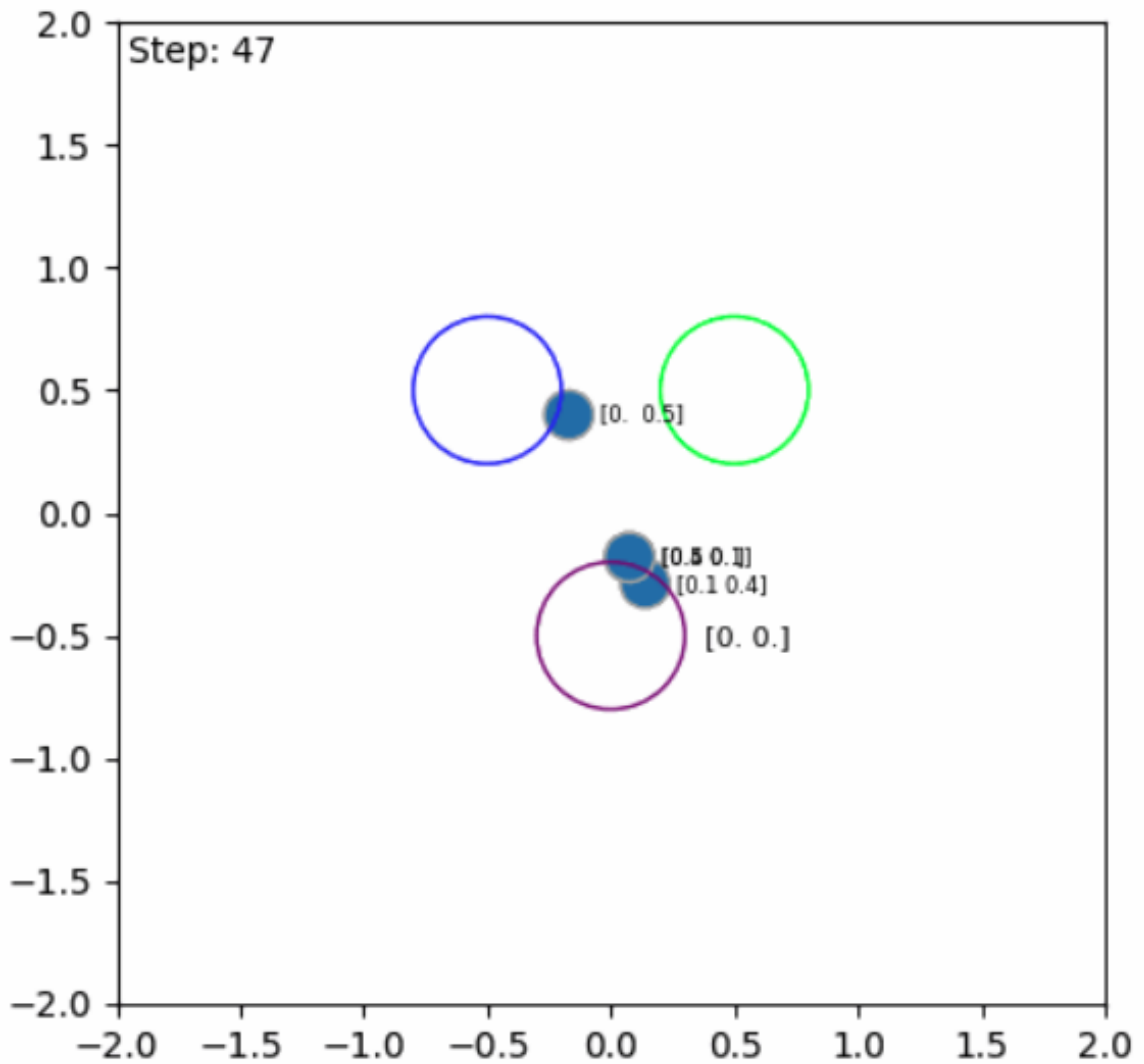


Figure 4.3: Example still taken from execution within `Transport`. Agents are represented as filled-in blue circles, with associated capacity for different materials listed in text following each agent. The larger hollow circles are sites of interest. The blue and green circles represent different sources of materials, and the purple circle is the depot which agents bring back materials to.

MLP is two linear layers with an intermediate ReLU activation. For QMIX and DAgger, our decoder is a single linear layer. These choices match the default architectures defined in JaxMARL.

**RNN-IMP:** The standard RNN architecture that *implicitly* encodes and reasons about agent heterogeneity. Indeed, some capabilities (e.g. speed) can be inferred solely from observations over time. This baseline questions the need for explicit conditioning on capabilities.

**RNN-EXP:** A modified version of the standard RNN architecture, which *explicitly* considers capability information by concatenating it to observations. This baseline is similar in spirit to prior techniques that append agent-specific information to the observations [24, 6]. This baseline questions the need for conditioning via hypernetworks.

**CASH:** Our proposed architecture also explicitly considers capability information, but leverages a hypernetwork to dynamically adapt the action decoder to each agent and context as detailed in section 4.1.

To ensure a fair comparison of architecture variants within each learning paradigm, we tune the **RNN-EXP** baseline over a range of widths and select the best performing width as measured by task success rate. Details on this ablation are in the Appendix. We then use the same width for the **RNN-IMP** baseline, and select a considerably smaller width for the RNN Encoder of CASH to showcase its improved representational power with fewer learnable parameters. Note that despite **RNN-EXP** having more inputs than **RNN-IMP**, we zero-pad the input to force the parameter count of both to be identical. This provides additional relative representational power to **RNN-IMP** as it needs to learn to implicitly reason about capabilities from observations, which is a more challenging task. Together, this set of baselines is representative of state-of-the-art approaches in shared-parameter approaches in learning decentralized policies for heterogeneous teams.

#### 4.2.4 Training and Testing Teams

To ensure fair comparisons, we standardize the teams observed during training and evaluation across all evaluated architectures.

*Train teams:* We sample a fixed number of train teams from a pool of agents whose capabilities uniformly cover a set range. Each episode rollout during training is then gathered with a randomly-selected train team. We find that limiting training teams to a small number of specific teams stabilizes learning while still allowing networks to generalize.

*Test teams:* During evaluation, we sample a fixed set of 10 test teams. Two of the three agents in the Firefighting task are sampled from an out of distribution capability range, while the third is constructed to ensure feasibility of the task. For HMT, agents are sampled from a wider range of capabilities than the training teams. At each evaluation step, one of the ten test teams is randomly selected for evaluation.

For additional details, including specific ranges for each task and the train/test teams used, see Appendix.

## CHAPTER 5

### RESULTS

In this chapter, we present and discuss our results from our comparison of CASH against the two baselines architectures, RNN-IMP and RNN-EXP, across two heterogeneous coordination tasks and three learning paradigms. We generated all results involving QMIX and MAPPO over 10 random seeds, and all results involving DAGGER results over 3 random seeds. The smoothed curves for the figures in this chapter are obtained by downsampling the original mean and standard deviation over all seeds, then taking a rolling average. We detail training hyperparameters and additional training details in the Appendices.

#### 5.1 Sample and Parameter Efficiency

We show the returns gathered by each architecture during training across all tasks and learning paradigms in Figure 5.1. As can be seen, CASH is consistently the most sample-efficient across the board. Further, CASH also consistently receives the highest returns at the end of training, albeit the difference between RNN-EXP and CASH is not significant when training with QMIX.

Notably, CASH achieves such superior performance with merely 20% to 40% of the learnable parameters utilized in both baselines (see bar charts in Figure 5.2). Given that we control for all other aspects of the architectural design, CASH’s improved efficiency can be attributed to its use of a hypernetwork. When comparing the two baselines against each other, we find that RNN-EXP consistently outperforms RNN-IMP in terms of sample efficiency likely because RNN-IMP requires more data to effectively capitalize on the implicit relationship between capabilities and observations. However, it is worth noting that RNN-IMP achieves comparable returns at the end of training in both RL paradigms across the two tasks. These observations suggest that implicit (RNN-IMP) and explicit

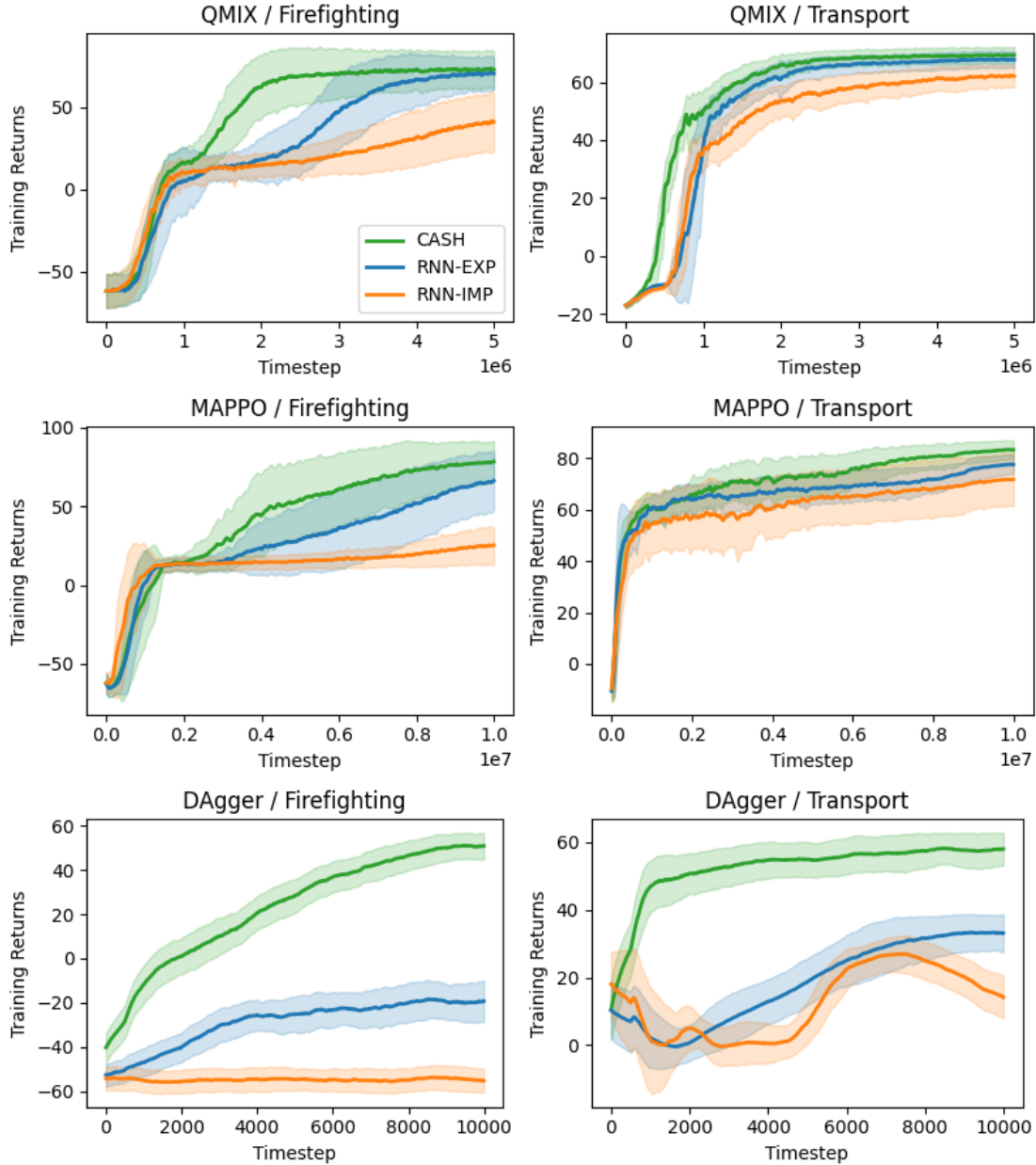


Figure 5.1: Training returns ( $\uparrow$ ) across two tasks and three learning paradigms. CASH consistently outperforms the baselines across learning paradigms and tasks despite having 20-40% of the learnable parameters (see Figure 5.2). For QMIX and MAPPO, results are shown from 10 random seeds. For DAgger, results are obtained over 3 random seeds.

(RNN-EXP) conditioning of capabilities can be effective in both value-based and policy-based RL, but only at the cost of significantly more parameters and poorer sample efficiency. In stark contrast, CASH’s improved grounding of capability information is able to simultaneously achieve better performance and remain the most sample and parameter

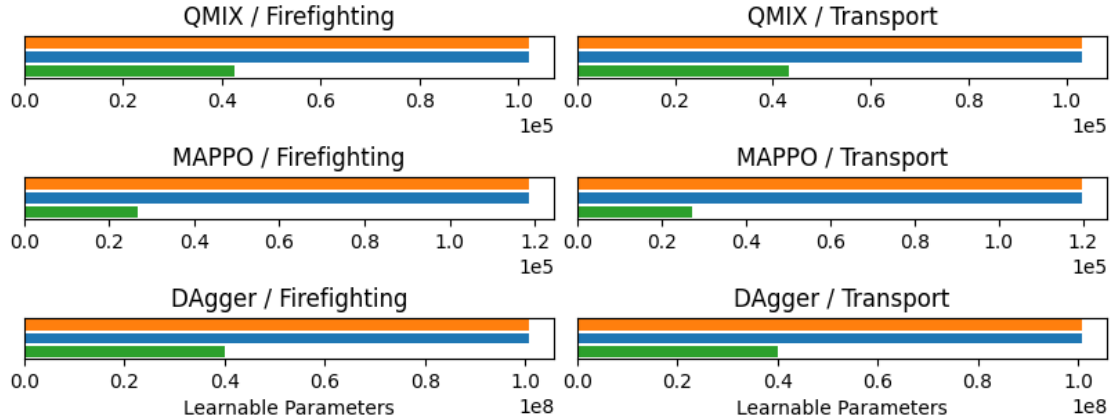


Figure 5.2: Learnable parameter counts ( $\downarrow$ ) for each baseline architecture compared in our experiments. Variations between tasks are solely due to differences in observation and action space dimensions – hidden layers and depths are kept constant between tasks for the same baseline. All of the performance results of CASH presented in the other figures are obtained with less than half of the learnable parameters as the baseline methods.

efficient.

## 5.2 Zero-Shot Generalization

We evaluated all trained policies on the unseen test teams, each involving out-of-distribution capabilities as explained previously. We show the success rates and task-specific metrics achieved by each architecture across the two tasks and three learning paradigms in Figure 5.3 and Figure 5.4, respectively. Across all reported metrics, we once again observe that CASH consistently scores higher than the baseline architectures when generalizing to new teams. Naturally, all architectures see a small drop in performance when generalizing to new teams, when compared to their performance during training. These findings suggest that all three architectures learn strategies during training that generalize to new teams. However, CASH learns the best-performing strategies with the fewest samples and maintains its edge when evaluated on generalization.

### 5.3 Behavioral Heterogeneity

In general, as seen in the SND plots (Figure 5.5), agents trained with CASH generally exhibit higher behavioral diversity than the baseline architectures. Naturally, the `RNN-IMP` architecture does not exhibit any diversity given that it has no access to capability information. This suggests that CASH can encode higher behavioral heterogeneity than existing techniques to bolster shared-parameter architectures. The one exception to this trend is in the SND values for Firefighting when trained with MAPPO: `RNN-EXP` produces policies that are slightly more heterogeneous. While this might appear to be disadvantageous to CASH at first glance, a careful qualitative analysis of episode rollouts reveals a different picture. We observe that both CASH and `RNN-EXP` are able to learn to move different agents to different fires even if they are currently in the same location. However, `RNN-EXP` is less adept at sending the agents with the appropriate capabilities to the best-suited fires in a way that benefits the overall task. Indeed, as we noted before, the success rate of CASH is far higher than `RNN-EXP` on MAPPO/Firefighting (see Figure 5.3). Together, these results suggest that `RNN-EXP` is learning *unproductive* heterogeneity on this specific condition. We conclude that CASH is better able to learn an *appropriate* level of heterogeneity that improves task performance, rather than merely maximizing heterogeneity.

### 5.4 Comparing Trends Across RL and IL

When combined with DAgger, CASH consistently resulted in remarkable improvements over the baseline architectures with respect to every metric that we considered, both during training as well on generalization across both tasks. Unlike RL, IL exposes architectures to optimal, but fewer, environment interactions from an expert policy. CASH’s flexible architecture likely benefits more from expert data since it can isolate and encode the expert’s agent-agnostic and capability-aware behaviors in dedicated modules.

## 5.5 Comparing Trends Across Tasks

When contrasting the performance of the architectures across the two tasks, we observe that the disparity between the three architectures is more pronounced for `Firefighting` compared to `Transport`. We hypothesize that this is due to tasks exhibiting varying levels of complex relationships between capabilities, task performance, and the need for behavioral heterogeneity. In this case, `Firefighting` demands tighter coordination and a more complex capability-based adaptation compared to `Transport`.

## 5.6 Layer Normalization

Finally, to show the importance of adding Layer Normalization [50] when training hypernetworks for multi-agent coordination problems, we present an ablation over the inclusion of LayerNorm in CASH’s Hyper Adapter in Figure 5.6. Across all tasks and learning algorithms, the inclusion of LayerNorm improves training stability and decreases variance between seeds. In all cases except one (MAPPO / `Transport`), it results in a significant improvement on the final converged returns. Once again, `DAGger` is a notable outlier in terms of performance gap between our method and the baselines: we see that when training with `DAGger`, the performance of CASH without LayerNorm in `Transport` plummets around timestep 4000 and never recovers, while in `Firefighting`, CASH without LayerNorm never demonstrates better than random performance.

Taken together, these results show that Layer Normalization improves training stability for hypernetworks, and that this improved stability is crucial for our architecture. We note that adding Layer Normalization to hypernetworks was suggested by the original Hypernetworks paper [30] to improve gradient flow, which our findings here corroborate. Surprisingly, prior work on using hypernetworks in meta-RL for generalization to new tasks [4, 34] does not include LayerNorm in the hypernetwork, even though we found it to be critically important for CASH. Both of these prior works are able to achieve strong perfor-

mance with a shallow two-layer hypernetwork by providing the input to the hypernet as a latent embedding from a pretrained encoder. However, in our setting we do not have access to a pretrained encoder, instead conditioning the Hyper Adapter on raw observations and capabilities. Empirically, we found that a deeper four-layer hypernetwork was necessary to achieve strong performance on this more challenging task. We leave a more thorough investigation of normalization schemes for training deep hypernetworks to future work.

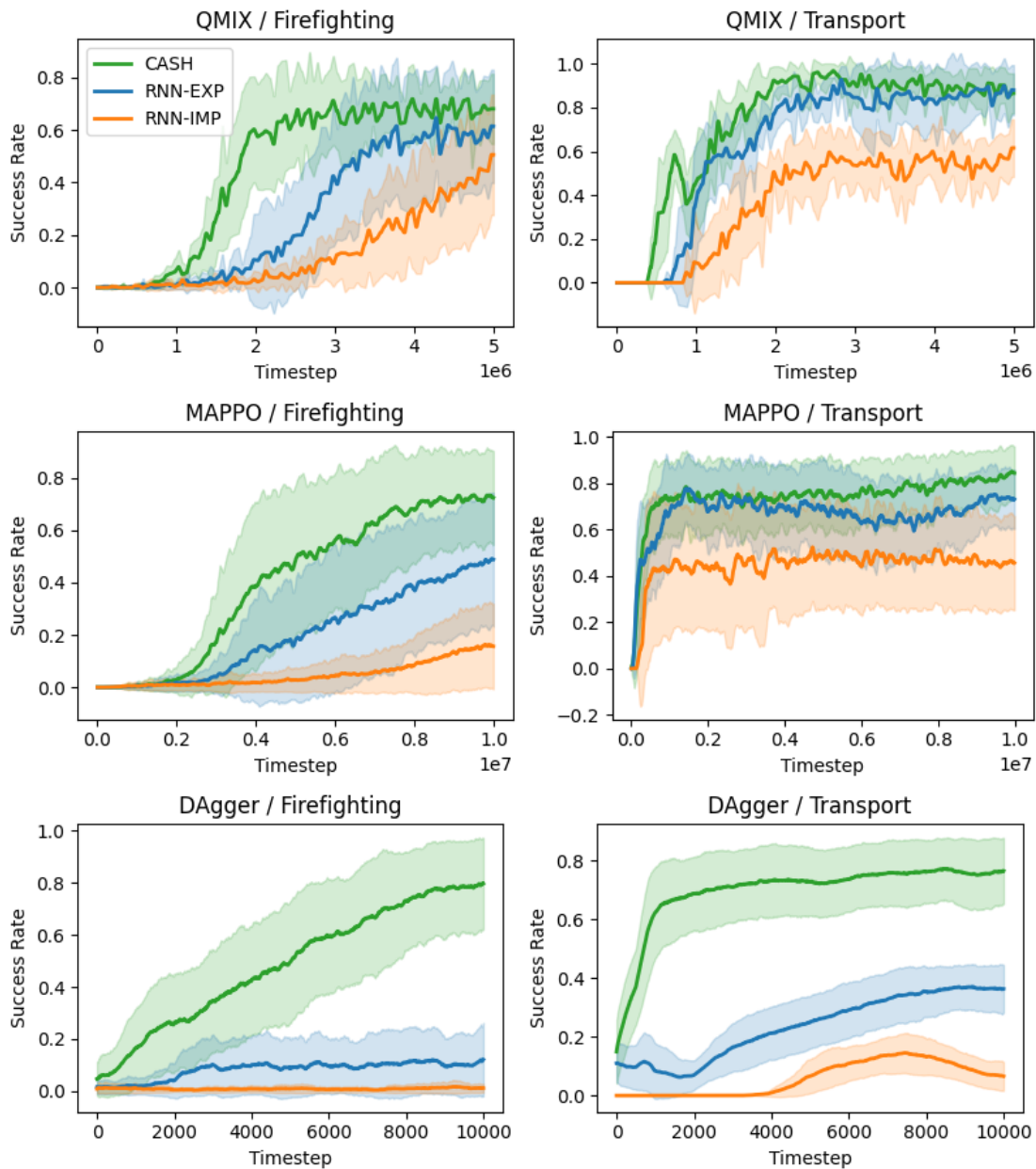


Figure 5.3: Success rate ( $\uparrow$ ) on unseen team compositions with unseen agent capabilities across two tasks and three learning paradigms. CASH consistently outperforms baseline methods when generalizing to new teams or agents.

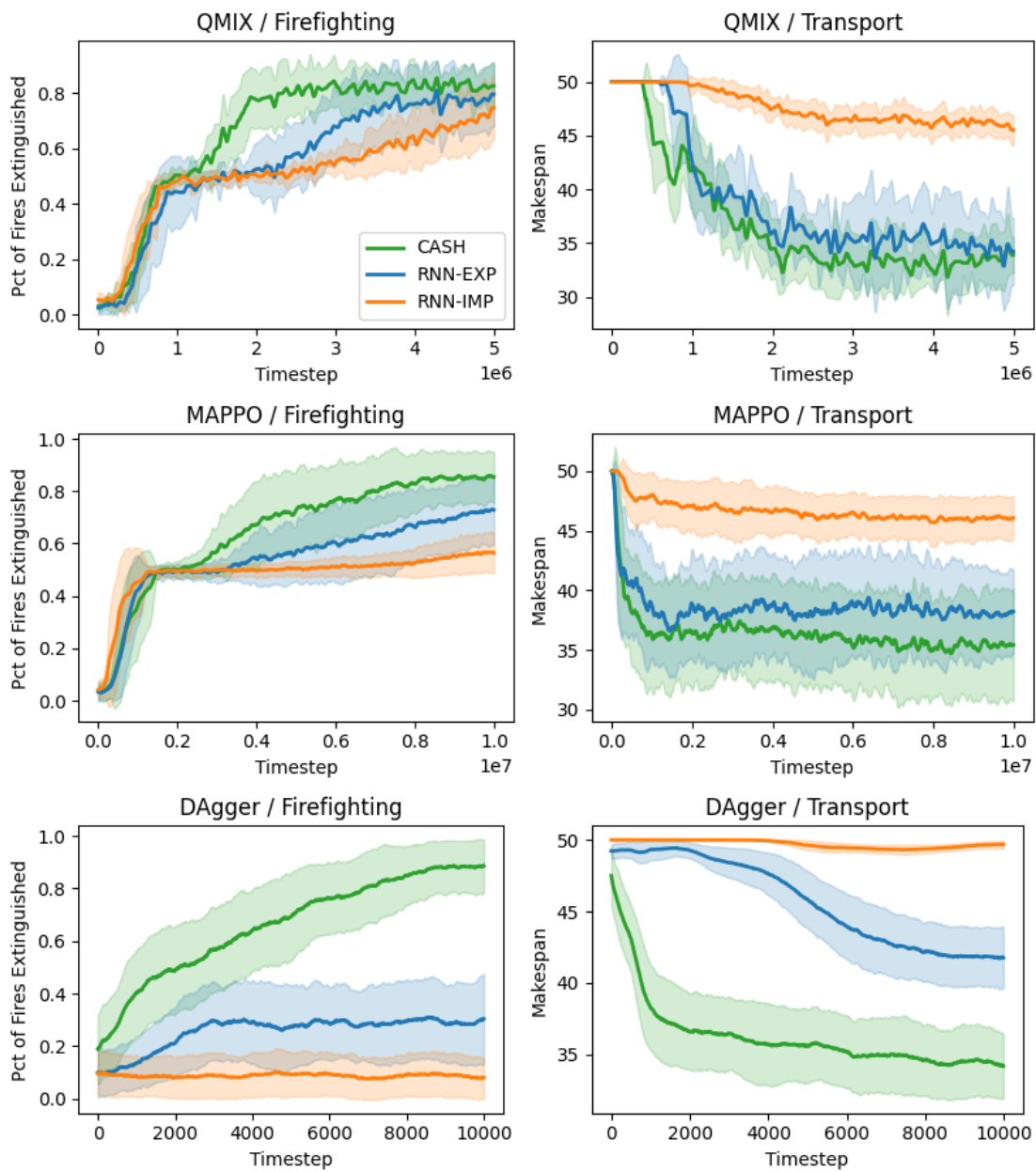


Figure 5.4: Additional task-specific performance metric across two tasks and three learning paradigms for unseen teams. Percentage of Fires Extinguished ( $\uparrow$ , on left) is for Firefighting. Makespan ( $\downarrow$ , on right) is for Transport. These metrics provide additional context beyond the success rates in Figure 5.3.

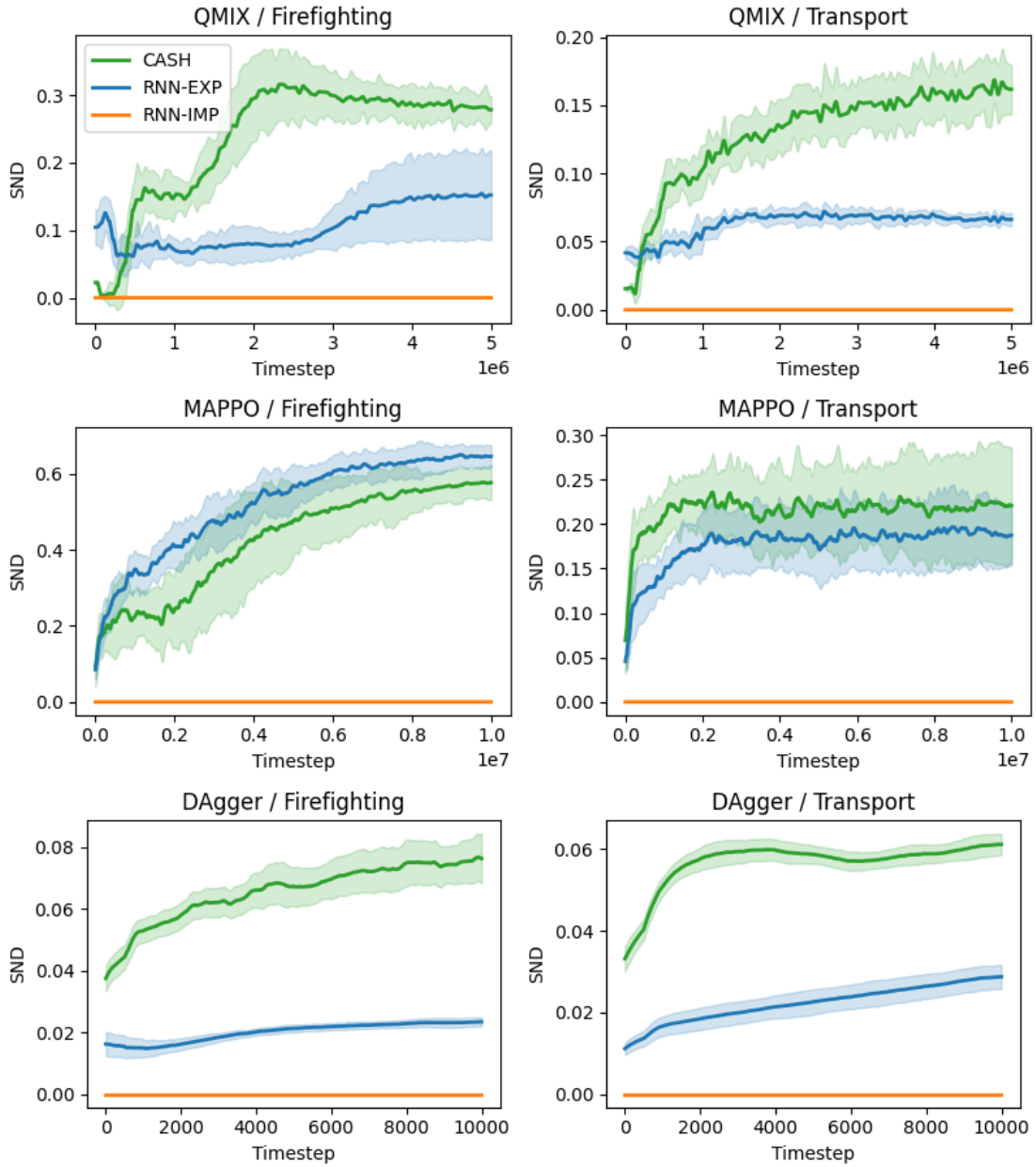


Figure 5.5: Behavioral diversity ( $\updownarrow$ ) across two tasks and three learning paradigms. In all cases but one (MAPPO/Firefighting), CASH exhibits more diverse behavior than the baseline approaches. Qualitative analysis of MAPPO/Firefighting suggests that RNN-EXP is able to learn diverse behaviors, but the corresponding success rate in Figure 5.3 suggests these behaviors are unproductive for task success. Conversely, CASH is able to learn an appropriate level of heterogeneity for strong task success.

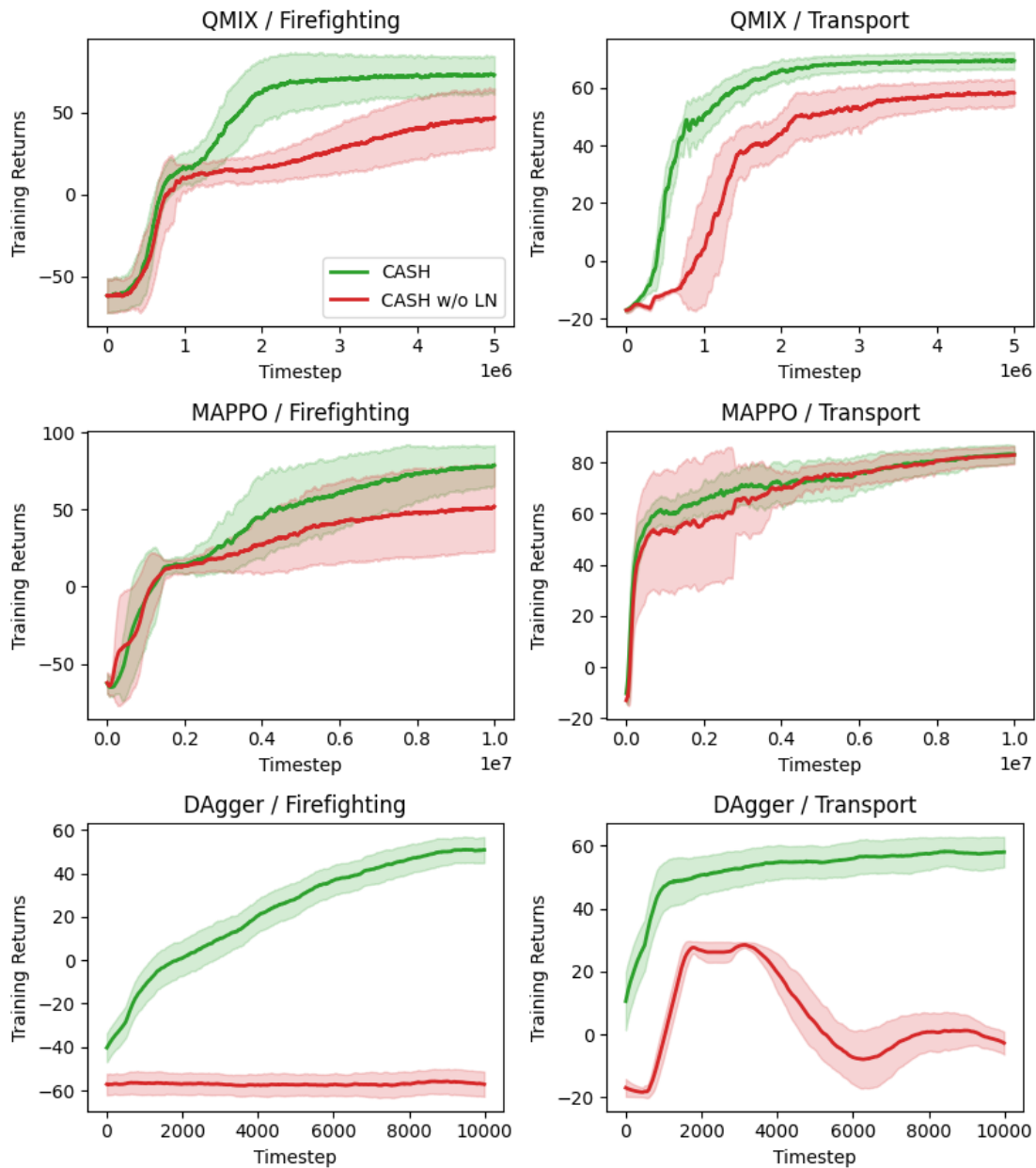


Figure 5.6: Impact of removing Layer Normalization from the Hyper Adapter of CASH when training with DAGger across two tasks. It is evident that LayerNorm is a crucial component in stabilizing the training of the hypernetwork within CASH.

## CHAPTER 6

### CONCLUSION AND DISCUSSION

In this work, we propose a new architecture for heterogeneous multi-agent coordination called *Capability-Aware Shared Hypernetworks (CASH)*, inspired by recent works in hypernetworks for transfer learning and meta reinforcement learning. Our experiments first illustrate the power of conditioning heterogeneous multi-agent policies on their capabilities on two complex multi-agent coordination tasks, then show that a hypernetwork-based architecture allows shared-parameter policies to demonstrate improved heterogeneity and task success as compared to policies without hypernetworks across two popular MARL algorithms and a multi-agent extension of DAgger. Furthermore, CASH outperforms these baselines despite having less than half of the learnable parameters across all tasks and learning paradigms. We believe our work represents a substantial contribution in the field of heterogeneous multi-agent learning and that future work should consider incorporating heterogeneity-conditioned hypernetworks into their architecture designs.

#### 6.1 Limitations

We acknowledge there are some limitations with the work presented in this thesis. For instance, though theoretically, CASH can be extended to any number of agents, prior work has showed that practically speaking, many decentralized MARL algorithms may not effectively handle large team sizes due to increases in training complexity. This may be especially problematic for CASH as the Hyper Adapter can be unstable to train even on the 3-4 agent teams we evaluated CASH on. Future work should investigate whether the gains introduced by CASH hold for larger agent teams, or if any modifications are needed.

Another limitation is that CASH assumes naively that agents can observe their neighbors fully, by directly appending relative teammate positions to the observations of each

agent. While fine in a small simulated environment, this assumption limits the applicability of CASH to larger-scale multi-robot systems. In Chapter 3 we explored the use of GNNs to combat imperfect information; future work on CASH could relax this assumption by incorporating a GNN into the encoder.

Finally, the tasks we evaluated CASH on are fairly simple compared to state-of-the-art tasks in heterogeneous multi-agent coordination (e.g. our proposed Firefighting task is a simplification of a more complex firefighting environment like FireCommander [28]). We chose these tasks as they allowed us to iterate quickly on the design of our policy architecture, as JaxMARL’s implementation of MPE is extremely lightweight and easy to use. Moreover, our experiments showed that despite the apparent simplicity of these tasks, capability-awareness improves the performance of policies on all of them, suggesting each task has a strong need for agents to accurately ground their actions in their individual heterogeneity to achieve team success. In our opinion, this is the most important characteristic for a heterogeneous multi-agent task to have, and abstracting away other details (e.g. more realistic communication, continuous action spaces, more dynamic environmental features) allowed us to focus on effective capability grounding and tease out how this can be done in an efficient way. However, we concede that a more thorough evaluation of CASH on a wider array of more complex tasks would be more persuasive.

## 6.2 Future Work

Finally, over the course of the research done for this thesis, we investigated several threads which did not make it into the final document, but all of which could become the basis for fruitful future research. For instance, one trend we noticed but did not investigate thoroughly was that including a hypernetwork in the policy architecture improved performance even *without* capability information sent to the hypernetwork; that is, a hypernetwork conditioned solely on observations to feed the parameters of a decoder layer showed some initial promise in terms of outperforming non-hypernet approaches. However, we did not

do our due diligence in running further experiments to explain why this would be the case, as the focus of this work was on heterogeneous teams where the capabilities are known. Another idea which we could have investigated more thoroughly is the idea of different mechanisms for adapting a shared encoder to the heterogeneity of individual agents, aside from a hypernetwork. As our experiments showed, hypernetworks can be difficult to optimize and tune, and their modifications to behavior can be hard to analyze. One architecture we tried was a residual network, which is a network whose outputs are meant to be summed with the outputs of the encoder. We abandoned this fairly early on as it did not outperform our hypernetwork architecture, but being able to analyze the exact changes made by the outputs of a residual network would allow for a far more explainable heterogeneity-aware decoder than a hypernetwork does.

Perhaps the most interesting unexplored thread is whether or not the idea of using hypernetworks for rapid adaptation can be used in MARL with different notions of heterogeneity. In this work we make the assumption that heterogeneity is known ahead of time and defined by a single capability vector. Future work could consider what happens when measures of heterogeneity are learned during training via observation and/or communication between other agents, or even learned during a pretraining phase. More ambitious still is the idea of using hypernetworks to adapt policies trained with single-agent RL with *team context* as the heterogeneity to condition a hypernetwork on. If possible, this would deconstruct the multi-agent learning problem into two clear phases: learning individual skills and learning effective team coordination, much like how many human teams operate. As single-agent RL is more sample-efficient than MARL by virtue of having less confounding agents in the environment, this should also hopefully have the positive benefit of increasing sample efficiency. We hope that future work will dive deeper into one of these paths which we have left unexplored.

# **Appendices**

## APPENDIX A

### EVALUATING BEHAVIORAL DIVERSITY WITH SND

To quantify the behavioral heterogeneity of the learned policies, we use the System Neural Diversity metric (SND) [57]. While SND was introduced to evaluate the heterogeneity of independent policies trained without parameter sharing, we modify SND to evaluate the heterogeneity of parameter-shared policies conditioned on heterogeneous capabilities. Specifically, given an observation, we append each agent’s capability to the observation and find the average pairwise distances between the policy outputs when conditioned on differing capabilities. We define pairwise distance as

$$d(i, j) = \frac{1}{|\mathcal{O}|} \sum_{o_t \in \mathcal{O}} TVD(\pi_\theta(o_t || c^i), \pi_\theta(o_t || c^j))$$

where  $\mathcal{O}$  is a set of observations obtained from several rollouts,  $c^i$  is the capability vector of agent  $i$ , and  $c^j$  is the capability vector of agent  $j$ . These pairwise distances are aggregated into a distance matrix, and the average of the upper triangular portion of this matrix is the final SND value. For value-based methods, individual networks output estimates of value per each action an agent can take. We interpret these Q-Values as a categorical distribution by taking a softmax over the values, and use Total Variational Distance when computing pairwise distances. Similarly, for discrete stochastic policies, we use Total Variational Distance between the categorical distributions over actions predicted by the policies.

## APPENDIX B

### ADDITIONAL TRAINING DETAILS

For CASH, we implemented the Hyper Adapter as two separate hypernetworks, one which generates the weights of the target linear layer and the other which generates the biases. We found that initializing both hypernetwork with orthogonal weights, zero-bias was conducive to good task success, which is common in other works. However, we noticed a slight performance increase when setting the *scale* of those weights to be 0 for the bias-generating hypernet and 0.2 for weight-generating hypernet. We could not find a reference for this odd initialization scheme in the literature.

#### **B.1 MAPPO Training Details**

All networks are trained with the hyperparameters in Table B.1. We ablated over four widths (32, 64, 128, 256) of RNN hidden dimension with the RNN-IMP architecture, and found 128 to be the best-performing variant as measured by test return. Based on this, we then designed a CASH architecture of lower parameter count than the baselines by selecting an RNN width of 32 and a hypernetwork width of 16.

#### **B.2 QMIX Training Details**

All networks are trained with the hyperparameters in Table B.2. We ablated over four widths (32, 64, 128, 256) of RNN hidden dimension with the RNN-IMP architecture, and found 128 to be the best-performing variant as measured by test return. Based on this, we then designed a CASH architecture of lower parameter count than the baselines by selecting an RNN width of 64 and a hypernetwork width of 32.

Table B.1: Hyperparameters for MAPPO

<b>Hyperparameter</b>	<b>Value</b>
Total timesteps	10e6
Learning rate	2e-3
Anneal learning rate	True
Update epochs	4
Number of minibatches	4
Gamma	0.99
GAE lambda	0.95
Clip epsilon	0.2
Scale clip epsilon	False
Entropy coefficient	0.01
Value function coefficient	0.5
Max gradient norm	0.5
Optimizer	Adam [58]

Table B.2: Hyperparameters for QMix

<b>Hyperparameter</b>	<b>Value</b>
Total timesteps	10e6
Learning rate	0.005
Learning rate linear decay	True
Buffer size	5000
Buffer batch size	32
Epsilon start	1.0
Epsilon finish	0.05
Epsilon anneal time	100000
Mixer embedding dim	32
Mixer hypernetwork hidden dim	64
Mixer initialization scale	0.00001
Max gradient norm	25
Target update interval	200
Optimizer	AdamW [59]
Epsilon Adam	0.001
Weight decay Adam	0.00001
TD lambda loss	True
TD lambda	0.6
Gamma	0.9

Table B.3: Hyperparameters for DAgger

Hyperparameter	Value
Expert buffer size	10000
Initial expert trajectories	1000
Iterations	10
Trajectories per iteration	1000
Learning rate	1e-4
Learning rate linear decay	False
Beta	1.0
Beta linear decay	True
Updates per iteration	100
Update batch size	64
Max gradient norm	1
Optimizer	AdamW [59]
Epsilon Adam	0.001
Weight decay Adam	0

### B.3 DAgger Training Details

All networks are trained with the hyperparameters in Table B.3. We ablated over four widths (512, 1024, 2048, 4096) of RNN hidden dimension with the RNN-IMP architecture, and found 4096 to be the best-performing variant as measured by test return. We were unable to include larger width RNNs due to memory constraints. Based on this, we then designed a CASH architecture of lower parameter count than the baselines by selecting an RNN width of 2048 and a hypernetwork width of 1024.

### B.4 Firefighting Train/Test Teams

In Table B.5 and Table B.4 we detail our randomly selected train and test teams for `Firefighting` and `Transport`. More details on how we chose these teams and how they were used can be found in the main body of Chapter 3. For context, in `Firefighting` the fires range from 0.2-0.3 strength and the training distribution of agents ranged between 0.1-0.3 fire-fighting capacity and 1-3 acceleration. For testing teams, we sample agents from out of bounds in both firefighting and acceleration, but match other agents to ensure task feasibil-

Table B.4: Firefighting Training Teams

Agent	(radius, acceleration) of agent
0	(0.3, 1)
1	(0.2, 2)
2	(0.1, 3)
3	(0.1, 3)
4	(0.2, 2)

Table B.5: Firefighting Testing Teams

Team	(radius, accel) for each agent in team
0	(0.09, 3.43), (0.21, 2.94), (0.42, 0.75)
1	(0.09, 3.41), (0.21, 3.00), (0.48, 0.63)
2	(0.05, 3.46), (0.25, 2.76), (0.44, 0.60)
3	(0.08, 3.23), (0.23, 2.80), (0.50, 0.61)
4	(0.09, 3.14), (0.21, 1.16), (0.47, 0.86)
5	(0.06, 3.45), (0.24, 2.08), (0.46, 0.76)
6	(0.08, 3.06), (0.22, 1.08), (0.48, 0.56)
7	(0.07, 3.04), (0.23, 2.37), (0.45, 0.56)
8	(0.09, 3.36), (0.21, 2.20), (0.49, 0.64)
9	(0.09, 3.26), (0.21, 2.59), (0.47, 0.64)

ity.

## B.5 Transport Train/Test Teams

In `Transport` we have a training range between 0-0.5 for each capacity, with each agent having a total capacity across the two materials that sums to 0.5. During testing the random sampling is altered such that the total capacity of each agent sums to 1.0, to present an out of distribution challenge. Exact hyperparameters can be found in Table B.6 and Table B.7. More details can be found in the main body of Chapter 3.

Table B.6: Transport Training Teams

Team	(1st material capacity, 2nd material capacity) for each agent of team
0	(0.1, 0.4), (0.2, 0.3), (0.3, 0.2), (0.5, 0.0)
1	(0.2, 0.3), (0.3, 0.2), (0.4, 0.1), (0.5, 0.0)
2	(0.0, 0.5), (0.1, 0.4), (0.3, 0.2), (0.5, 0.0)
3	(0.0, 0.5), (0.1, 0.4), (0.2, 0.3), (0.4, 0.1)
4	(0.0, 0.5), (0.1, 0.4), (0.2, 0.3), (0.5, 0.0)
5	(0.1, 0.4), (0.3, 0.2), (0.4, 0.1), (0.5, 0.0)
6	(0.0, 0.5), (0.2, 0.3), (0.4, 0.1), (0.5, 0.0)
7	(0.0, 0.5), (0.2, 0.3), (0.3, 0.2), (0.5, 0.0)
8	(0.0, 0.5), (0.1, 0.4), (0.4, 0.1), (0.5, 0.0)
9	(0.1, 0.4), (0.2, 0.3), (0.4, 0.1), (0.5, 0.0)

Table B.7: Transport Testing Teams

Team	(1st material capacity, 2nd material capacity) for each agent of team
0	(0.72, 0.28), (0.98, 0.02), (0.17, 0.83), (0.12, 0.13)
1	(0.63, 0.37), (0.88, 0.12), (0.56, 0.44), (0.17, 0.08)
2	(0.04, 0.96), (0.24, 0.76), (0.29, 0.71), (0.25, 0. )
3	(0.01, 0.99), (0.56, 0.44), (0.18, 0.82), (0.05, 0.2 )
4	(0.26, 0.74), (0.65, 0.35), (0.95, 0.05), (0.16, 0.09)
5	(0.37, 0.63), (0.52, 0.48), (0.77, 0.23), (0.04, 0.21)
6	(0.41, 0.59), (0.96, 0.04), (0.12, 0.88), (0.05, 0.2 )
7	(0.86, 0.14), (0.97, 0.03), (0.38, 0.62), (0.12, 0.13)
8	(0.68, 0.32), (0.87, 0.13), (0.73, 0.27), (0.15, 0.1 )
9	(0.4, 0.6), (0.41, 0.59), (0.54, 0.46), (0.02, 0.23)

## REFERENCES

- [1] H. Ravichandar, K. Shaw, and S. Chernova, “Strata: Unified framework for task assignments in large teams of heterogeneous agents.,” *Autonomous Agents and Multi-Agent Systems*, 2020.
- [2] C. Yu, A. Velu, E. Vinitzky, Y. Wang, A. M. Bayen, and Y. Wu, “The surprising effectiveness of MAPPO in cooperative, multi-agent games,” *CoRR*, 2021.
- [3] J. von Oswald, C. Henning, B. F. Grewe, and J. Sacramento, “Continual learning with hypernetworks,” no. arXiv:1906.00695, Apr. 2022.
- [4] J. Beck, R. Vuorio, Z. Xiong, and S. Whiteson, “Recurrent hypernetworks are surprisingly strong in meta-rl,” in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36, Curran Associates, Inc., 2023, pp. 62 121–62 138.
- [5] J. Oh, S. Singh, H. Lee, and P. Kohli, “Zero-shot task generalization with multi-task deep reinforcement learning,” in *Proceedings of the 34th International Conference on Machine Learning*, D. Precup and Y. W. Teh, Eds., ser. Proceedings of Machine Learning Research, vol. 70, PMLR, Jun. 2017, pp. 2661–2670.
- [6] P. Howell, M. Rudolph, R. J. Torbati, K. Fu, and H. Ravichandar, “Generalization of heterogeneous multi-robot policies via awareness and communication of capabilities,” in *Proceedings of The 7th Conference on Robot Learning*, J. Tan, M. Toussaint, and K. Darvish, Eds., ser. Proceedings of Machine Learning Research, vol. 229, PMLR, Jun. 2023, pp. 2772–2790.
- [7] T. Galanti and L. Wolf, “On the modularity of hypernetworks,” in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 10 409–10 419.
- [8] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, “Multi-agent actor-critic for mixed cooperative-competitive environments,” *Neural Information Processing Systems (NIPS)*, 2017.
- [9] S. Omidshafiei, J. Papis, C. Amato, J. P. How, and J. Vian, “Deep decentralized multi-task multi-agent reinforcement learning under partial observability,” in *International Conference on Machine Learning*, 2017.
- [10] B. Ellis, S. Moalla, M. Samvelyan, M. Sun, A. Mahajan, J. N. Foerster, and S. Whiteson, *Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning*, 2022.

- [11] Y. Hoshen, “Vain: Attentional multi-agent predictive modeling,” in *Advances in Neural Information Processing Systems*, 2017.
- [12] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson, *Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning*, 2018. eprint: 1803.11485.
- [13] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, *Playing atari with deep reinforcement learning*, 2013.
- [14] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, *Continuous control with deep reinforcement learning*, 2019.
- [15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal policy optimization algorithms*, 2017.
- [16] V. Egorov and A. Shpilman, *Scalable multi-agent model-based reinforcement learning*, 2022. eprint: 2205.15023.
- [17] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, *Counterfactual multi-agent policy gradients*, 2017. eprint: 1705.08926.
- [18] T. Dang, M. Tranzatto, S. Khattak, F. Mascarich, K. Alexis, and M. Hutter, “Graph-based subterranean exploration path planning using aerial and legged robots,” *Journal of Field Robotics*, 2020.
- [19] T. Dang, M. Tranzatto, S. Khattak, F. Mascarich, K. Alexis, and M. Hutter, “Graph-based subterranean exploration path planning using aerial and legged robots,” *Journal of Field Robotics*, 2020.
- [20] G. Gil, D. E. Casagrande, L. P. Cortés, and R. Verschae, “Why the low adoption of robotics in the farms? challenges for the establishment of commercial agricultural robots,” *Smart Agricultural Technology*, 2023.
- [21] A. Khan, E. Tolstaya, A. Ribeiro, and V. Kumar, “Graph Policy Gradients for Large Scale Robot Control,” in *Proceedings of the Conference on Robot Learning*, PMLR, 2020.
- [22] G. Sartoretti, J. Kerr, Y. Shi, G. Wagner, T. K. S. Kumar, S. Koenig, and H. Choset, “Primal: Pathfinding via reinforcement and imitation multi-agent learning,” *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2378–2385, 2019.
- [23] Q. Li, F. Gama, A. Ribeiro, and A. Prorok, “Graph neural networks for decentralized multi-robot path planning,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 11 785–11 792.

- [24] J. K. Terry, N. Grammel, S. Son, B. Black, and A. Agrawal, *Revisiting parameter sharing in multi-agent deep reinforcement learning*, 2023. arXiv: 2005.13625 [cs.LG].
- [25] F. Christianos, G. Papoudakis, M. A. Rahman, and S. V. Albrecht, “Scaling multi-agent reinforcement learning with selective parameter sharing,” in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 139, PMLR, 2021.
- [26] M. Bettini, A. Shankar, and A. Prorok, “Heterogeneous multi-robot reinforcement learning,” in *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS ’23, 2023.
- [27] C. Li, T. Wang, C. Wu, Q. Zhao, J. Yang, and C. Zhang, “Celebrating diversity in shared multi-agent reinforcement learning,” in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34, Curran Associates, Inc., 2021, pp. 3991–4002.
- [28] E. Seraj, Z. Wang, R. Paleja, D. Martin, M. Sklar, A. Patel, and M. Gombolay, “Learning efficient diverse communication for cooperative heterogeneous teaming,” in *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS ’22, Virtual Event, New Zealand: International Foundation for Autonomous Agents and Multiagent Systems, 2022, pp. 1173–1182, ISBN: 9781450392136.
- [29] C. Wakilpoor, P. J. Martin, C. Rebhuhn, and A. Vu, *Heterogeneous multi-agent reinforcement learning for unknown environment mapping*, 2020. arXiv: 2010.02663 [cs.MA].
- [30] D. Ha, A. Dai, and Q. V. Le, “Hypernetworks,” no. arXiv:1609.09106, Dec. 2016.
- [31] Y. Huang, K. Xie, H. Bharadhwaj, and F. Shkurti, “Continual model-based reinforcement learning with hypernetworks,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 799–805.
- [32] S. Rezaei-Shoshtari, C. Morissette, F. R. Hogan, G. Dudek, and D. Meger, “Hypernetworks for zero-shot transfer in reinforcement learning,” in *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI’23/IAAI’23/EAAI’23, AAAI Press, 2023, ISBN: 978-1-57735-880-0.
- [33] E. Sarafian, S. Keynan, and S. Kraus, “Recomposing the reinforcement learning building blocks with hypernetworks,” in *Proceedings of the 38th International Con-*

- ference on Machine Learning*, M. Meila and T. Zhang, Eds., ser. Proceedings of Machine Learning Research, vol. 139, PMLR, 18–24 Jul 2021, pp. 9301–9312.
- [34] J. Beck, M. T. Jackson, R. Vuorio, and S. Whiteson, “Hypernetworks in meta-reinforcement learning,” in *Proceedings of The 6th Conference on Robot Learning*, K. Liu, D. Kulic, and J. Ichnowski, Eds., ser. Proceedings of Machine Learning Research, vol. 205, PMLR, 14–18 Dec 2023, pp. 1478–1487.
- [35] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, “Monotonic value function factorisation for deep multi-agent reinforcement learning,” *Journal of Machine Learning Research*, vol. 21, no. 178, pp. 1–51, 2020.
- [36] F. A. Oliehoek and C. Amato, *A Concise Introduction to Decentralized POMDPs*. SpringerBriefs in Intelligent Systems, 2016.
- [37] A. Agarwal, S. Kumar, and K. Sycara, *Learning transferable cooperative behavior in multi-agent teams*, 2019.
- [38] A. Khan, E. Tolstaya, A. Ribeiro, and V. Kumar, *Graph policy gradients for large scale robot control*, 2019.
- [39] L. Yu, J. Song, and S. Ermon, “Multi-agent adversarial inverse reinforcement learning,” in *International Conference on Machine Learning*, 2019.
- [40] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal Policy Optimization Algorithms,” 2017.
- [41] I. Mordatch and P. Abbeel, “Emergence of grounded compositional language in multi-agent populations,” *CoRR*, 2017.
- [42] G. Papoudakis, F. Christianos, L. Schäfer, and S. V. Albrecht, “Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks,” in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS)*, 2021.
- [43] R. Torbati, S. Lohiya, S. Singh, M. S. Nigam, and H. Ravichandar, *Marbler: An open platform for standardized evaluation of multi-robot reinforcement learning algorithms*, 2023.
- [44] S. Wilson, P. Glotfelter, L. Wang, S. Mayya, G. Notomista, M. Mote, and M. Egerstedt, “The robotarium: Globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multirobot systems,” *IEEE Control Systems Magazine*, 2020.

- [45] J. Foerster, I. A. Assael, N. De Freitas, and S. Whiteson, “Learning to communicate with deep multi-agent reinforcement learning,” *Advances in neural information processing systems*, 2016.
- [46] J. K. Gupta, M. Egorov, and M. Kochenderfer, “Cooperative multi-agent control using deep reinforcement learning,” in *Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops, Best Papers, São Paulo, Brazil, May 8-12, 2017, Revised Selected Papers 16*, Springer, 2017.
- [47] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, *Learning phrase representations using rnn encoder-decoder for statistical machine translation*, 2014. arXiv: 1406.1078 [cs.CL].
- [48] C. Yu, A. Velu, E. Vinitzky, J. Gao, Y. Wang, A. Bayen, and Y. WU, “The surprising effectiveness of ppo in cooperative multi-agent games,” in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35, Curran Associates, Inc., 2022, pp. 24 611–24 624.
- [49] V. K. Chauhan, J. Zhou, P. Lu, S. Molaei, and D. A. Clifton, “A brief review of hypernetworks in deep learning,” no. arXiv:2306.06955, Aug. 2023.
- [50] J. L. Ba, J. R. Kiros, and G. E. Hinton, *Layer normalization*, 2016. arXiv: 1607.06450 [stat.ML].
- [51] P. J. Ball, L. Smith, I. Kostrikov, and S. Levine, “Efficient online reinforcement learning with offline data,” in *International Conference on Machine Learning*, PMLR, 2023, pp. 1577–1594.
- [52] M. Gallici, M. Fellows, B. Ellis, B. Pou, I. Masmitja, J. N. Foerster, and M. Martin, “Simplifying deep temporal difference learning,” *arXiv preprint arXiv:2407.04811*, 2024.
- [53] S. Ross and D. Bagnell, “Efficient reductions for imitation learning,” in *Proceedings of the thirteenth international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings*, 2010, pp. 661–668.
- [54] A. Rutherford, B. Ellis, M. Gallici, J. Cook, A. Lupu, G. Ingvarsson, T. Willi, A. Khan, C. S. de Witt, A. Souly, S. Bandyopadhyay, M. Samvelyan, M. Jiang, R. T. Lange, S. Whiteson, B. Lacerda, N. Hawes, T. Rocktaschel, C. Lu, and J. N. Foerster, “Jaxmarl: Multi-agent rl environments in jax,” no. arXiv:2311.10090, Dec. 2023, arXiv:2311.10090 [cs].
- [55] R. Lowe, Y. WU, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, “Multi-agent actor-critic for mixed cooperative-competitive environments,” in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach,

R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30, Curran Associates, Inc., 2017.

- [56] M. Bettini, A. Shankar, and A. Prorok, “System neural diversity: Measuring behavioral heterogeneity in multi-agent learning,” *arXiv preprint arXiv:2305.02128*, 2023.
- [57] M. Bettini, A. Shankar, and A. Prorok, “System neural diversity: Measuring behavioral heterogeneity in multi-agent learning,” no. arXiv:2305.02128, May 2023.
- [58] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.
- [59] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” in *International Conference on Learning Representations*, 2019.