ADAPTABLE AND SCALABLE MULTI-AGENT GRAPH-ATTENTION COMMUNICATION

A Dissertation
Presented to
The Academic Faculty

By

Yaru Niu

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in
Electrical and Computer Engineering
at College of Engineering

Georgia Institute of Technology

May 2022

ADAPTABLE AND SCALABLE MULTI-AGENT GRAPH-ATTENTION COMMUNICATION

Date approved: April 29th, 2022

Will robots inherit the earth? Yes, but they will be our children.

Marvin Minsky

ACKNOWLEDGMENTS

Over the past two years and eight months, I have received unfailing support from a group of people who I would like to acknowledge. Without them, this work would be impossible.

First I would like to express immense gratitude to my research advisor Dr. Matthew Gombolay. He has inspired me with insightful ideas, pushed me with constructive criticism, and encouraged me with unwavering support. Under his guidance, I grow up to be a capable researcher.

I am also grateful to my thesis committee members, Dr. Ayanna Howard and Dr. Sonia Chernova, for their passionate participation and critical input to this thesis. They have been my role models for AI researchers and roboticists.

I would also like to thank my research mentor and collaborator Rohan Paleja, who gives me golden advice on research projects, works and stays up with me during those long nights for paper deadlines.

I have to thank my parents and my friends who support and love me unconditionally through this journey. They are always my home and my harbor. I have the courage to pursue my dream and embark on adventures because of them.

I would gratefully acknowledge the support for this work offered by MIT Lincoln Laboratory under grant number 7000437192.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

High-performing teams learn effective communication strategies to judiciously share information and reduce the cost of communication overhead. Within multi-agent reinforcement learning, synthesizing effective policies requires reasoning about when to communicate, whom to communicate with, and how to process messages. Meanwhile, in real-world problems, training policies and communication strategies that are able to generalize to multiple tasks, and adapt to unseen tasks, can improve the learning efficiency in multi-agent systems. However, many methods in current literature suffer from efficiently learning a dynamic communication topology. At the same time, learning adaptable and scalable multi-agent communication remains to be a challenge. This thesis develops algorithms to tackle these two problems.

First, I propose a novel multi-agent reinforcement learning algorithm, Multi-Agent Graph-attention Communication (MAGIC), with a graph-attention communication protocol in which we learn 1) a Scheduler to help with the problems of when to communicate and whom to address messages to, and 2) a Message Processor using Graph Attention Networks (GATs) with dynamic graphs to deal with communication signals. The Scheduler consists of a graph attention encoder and a differentiable attention mechanism, which outputs dynamic, differentiable graphs to the Message Processor, which enables the Scheduler and Message Processor to be trained end-to-end. We evaluate our approach on a variety of cooperative tasks, including Google Research Football. Our method outperforms baselines across all domains, achieving $\approx 10.5\%$ increase in reward in the most challenging domain. We also show MAGIC communicates 27.4% more efficiently on average than baselines, is robust to stochasticity, and scales to larger state-action spaces. Finally, we demonstrate MAGIC on a physical, multi-robot testbed.

Second, based on MAGIC, I present a multi-agent multi-task reinforcement training scheme, MT-MAGIC, and develop a multi-agent meta-reinforcement learning framework, Meta-MAGIC. Both methods can generalize and adapt to unseen tasks with different team sizes. Meta-MAGIC initiatively explores using the RNN architecture to perform the adaptation process in multi-agent meta-reinforcement learning. Through experiments, we find that Meta-MAGIC and MT-MAGIC can beat the baseline by a notable margin in multi-task training and generalize well to new tasks. Meta-MAGIC is able to adapt quickly to new tasks and keeps an upper bound of the performance of all methods through the interactions with unseen scenarios in Predator-Prey. Fine-tuning from pre-trained models by MT-MAGIC quickly achieves better performance on new tasks compared to training from scratch, with only 11.28% of training epochs.

CHAPTER 1
INTRODUCTION

1.1 Background and Motivation

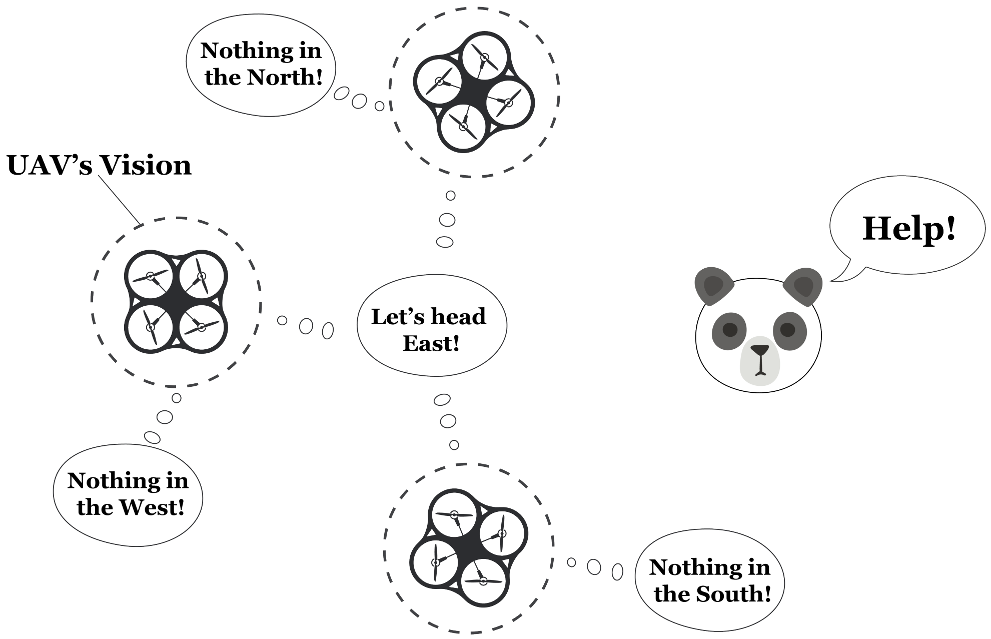


Figure 1.1: This figure displays a scenario where three unmanned aerial vehicles with limited visions (dash circles) are searching to rescue a lost animal (panda) in a forest.

Communication is a key component of successful coordination, enabling the agents to convey information and cooperate to collectively achieve shared goals [1, 2]. In high-performing human teams, human experts judiciously choose when to communicate and whom to communicate with, communicating only when beneficial [3, 4, 5]. Each team member exhibits the role of a communicator and message receiver, relaying information to the right teammates and incorporating received information effectively. What is more, multi-round (multi-hop) communication, i.e., multi-round of back-and-forth interactive message passing between agents, is usually required to establish complex cooperation strategies [6]. Multi-round (multi-hop) communication is also commonly employed in communication systems [7, 8].

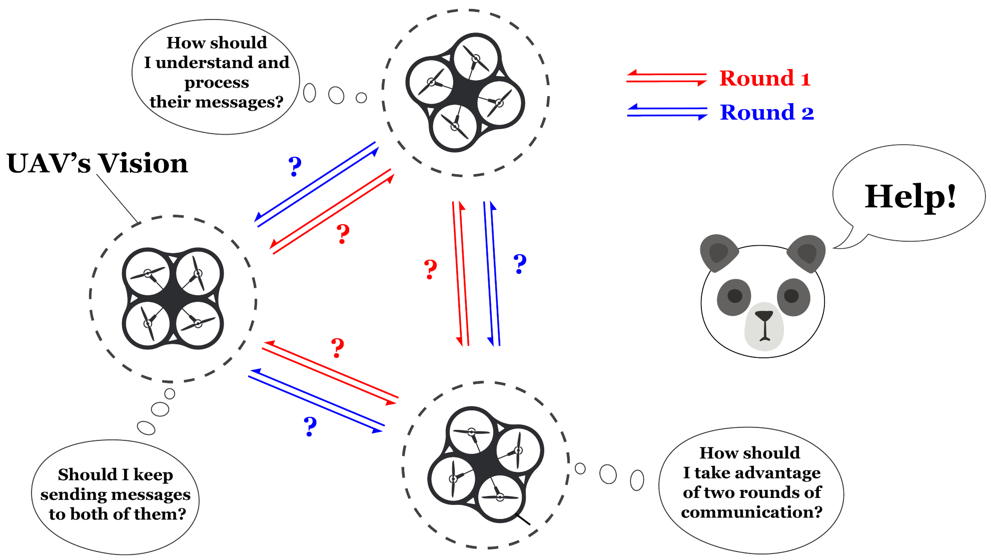

How should I understand and process their messages?
Round 1
Round 2
UAV's Vision
?
?
?
?
?
?
Help!
Should I keep sending messages to both of them?
How should I take advantage of two rounds of communication?

Figure 1.2: This figure displays a scenario where three unmanned aerial vehicles are reasoning about when, whom, and how to communicate with two rounds of communication to locate the lost animal (panda).

For example, in Figure 1.1, three unmanned aerial vehicles (UAVs) are searching to rescue a lost panda in a forest. Each UAV can only get access to the vision around it. At the start, none of them found the panda at their locations. After communication, they finally decide to expand their rescue area to the east side of the forest, which is where the panda locates at. Then the remaining problem is to decide how to communicate efficiently to make wise decisions. Figure 1.2 depicts a scenario in which agents are reasoning about when, whom, and how to communicate with two rounds of communication. Specifically, the UAV agents should dynamically choose targeted agents to send messages, processed the messages sent from other UAVs for later rounds of communication or the final decision.

Reinforcement learning provides a solution to multi-agent communication. Over the past few years, reinforcement learning has received great attention from researchers due to its success in multiple domains, games [9, 10], robotics [11, 12], traffic control [13, 14], healthcare [15], natural language processing [16, 17, 18], etc. Furthermore, there has been recent success in multi-agent reinforcement learning (MARL) for Multiplayer Online

Battle Arena (MOBA) games such as StarCraft II and Dota II [19, 20, 21]. MARL seeks to enable agents to share information to improve team performance [22, 23, 24, 6, 25, 26]. However, most prior work in MARL fails to capture the complex relations among agents, leading to low-performance and inefficient communication. While [22] and [27] are able to efficiently decide when to broadcast messages, agents will broadcast these messages to all other agents without targets, resulting in wasteful communication. Even with targeted communication [6], failure to assess when to communicate results in poor performance, as we display in section 3.6. However, determining when to communicate and whom to communicate with is not enough. Selectively utilizing received messages can significantly improve performance. Yet, none of these methods simultaneously address "when" and with "whom", and "how" to communicate while modeling agent interaction topology.

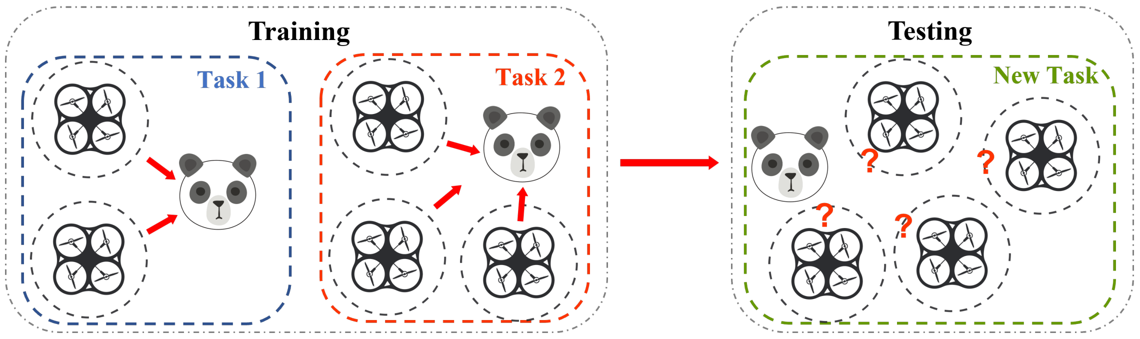


Figure 1.3: This figure depicts a simple multi-task learning and adaption task for the multi-agent search and rescue example used in this chapter.

Besides communication, learning and adapting fast is a hallmark of human intelligence [28]. People can usually quickly learn new things because they never learn from scratch. They can utilize their past experiences, and generalize and adapt learned skills to new tasks. To endow the intelligent agents with the same capability, multi-task learning [29, 30, 31], transfer learning [32, 33, 30], and meta-learning [34, 35, 36] have been studied for years. However, it remains to be challenging for multi-agent systems and MARL. In MARL, learning to adapt not only requires learning to generalize agent's own policy, but also requires learning to generalize strategies for inter-agent coordination or communication to new tasks. What is more, different tasks might require different team sizes. For example, in Figure 1.3, we present an example using the before-mentioned multi-agent search and rescue scenario. It can be difficult to perform multi-task learning on task one with two agents and task two with three agents, then quickly adapt to an unseen task with four agents during test time.

Recently, some work has explored solutions to such problems for MARL. Ad-hoc team play seeks to enable agents to collaborate with unfamiliar agents [37, 38, 39]. These works assume no communication protocol or only undirected message passing from a coach agent to other agents. Some approaches employ Graph Neural Networks to propagate information and communicate messages through a locomotion agent's different body parts, which can be structured as a graph and regarded as many sub-agents [40, 41, 42]. Their trained policies show good generalization and transfer ability to different agent types with different body sizes. Their tasks are limited to locomotion, and fix the communication topology among sub-agents according to the static body structure. However, a dynamic communication strategy (learning "when" and "whom") is usually required for high-performing coordination. In the field of meta-reinforcement learning for multi-agent settings, many methods focus on extending optimization-based methods [43, 36, 44] to a multi-agent team [43, 45, 46, 47]. However, most prior work handles different multi-agent tasks with the same size, and none of the previous work employs a black-box adaptation [48, 49], which is a kind of commonly-used meta-reinforcement learning method.

1.2 Thesis Outline

The goal of this thesis is to solve the main problems existing in MARL mentioned in section 1.1, and sketch a path to adaptable and scalable multi-agent communication. Accordingly, this thesis is structured in two parts, Multi-Agent Graph-Attention Communication (chapter 3), and Learning Scalable and Adaptable Multi-Agent Communication (chapter 4). We will introduce the related work of both parts in chapter 2.

1.2.1 Multi-Agent Graph-Attention Communication

The goal of the first part of this thesis (chapter 3) is to solve the existing “when”,“whom”, and “how” problems, and develop a novel high-performing communication-based MARL method, MAGIC. It focuses on designing a fully differentiable communication protocol that supports learning dynamic hierarchical inter-agent interaction topology in favor of more powerful multi-round communication. Specifically, we discuss our method within the assumption of Partially Observable Markov Game, with augmentation of communication signals. In the method section, I present the overall framework of the proposed protocol, and its two key components, the Scheduler, and the Message Processor. Then I evaluate and compare our method against several state-of-the-art communication-based approaches, in three domains, including two grid worlds limiting agents’ visions, and one challenging 3D football game simulator. I also provide a communication efficiency evaluation across different methods, and a physical robot demonstration at the end of chapter 3.

1.2.2 Scalable and Adaptable Multi-Agent Communication

Chapter 3 focuses on learning multi-agent communication for single tasks, without consideration of its generality and adaptability. The goal of the second part of this thesis (chapter 4) is to fill some above-mentioned gaps of current MARL literature, and extend MAGIC to multi-tasks and meta-learning settings. Specifically, I present a multi-task training scheme based on chapter 3’s assumption and framework, which enables multi-task learning across teams with different number of agents. Then I propose and describe a meta-learning for MARL based on black-box adaptation. Next, I design experiments in two domains to evaluate our methods. In the end of chapter 4, I discuss the results in three aspects, multi-task training, zero-shot/few-shot testing, and fine-tuning.

1.3 Thesis Contributions

This thesis proposes methodologies and and presents experiments of adaptable and scalable multi-agent graph-attention communication within the framework of MARL. I list both its theoretical/algorithmic and empirical contributions as follow.

1. We propose a novel communication protocol for MARL that enables the agents to decide to when to communicate, whom to communicate with, and how to process the received messages. Our method is able to dynamically learn multiple hierarchical communication graphs to support more powerful multi-rounds of communication.

2. We implement extensive experiments on two previously conquered domains and one challenging unsolved domain. Our proposed MAGIC achieves the best performance in all domains. We also release a public codebase [1] of our method and baselines in these domains.

3. We present a multi-agent multi-task reinforcement learning framework and design a black-box adaptation meta-reinforcement learning method based on MAGIC, which can handle tasks with different team sizes. To the best of our knowledge, no previous work has applied black-box adaption for meta-learning in an MARL framework.

4. We implement a series of multi-task and adaptation experiments. Our methods achieve good performance in most tested tasks. Importantly, we initiatively present and analyze the results of tentative multi-agent meta-reinforcement learning with RNN-based black-box adaptation.

[1] https://github.com/CORE-Robotics-Lab/MAGIC

CHAPTER 2
RELATED WORK

In this chapter, we present related work and discuss existing problems with regards to MARL and adaptable MARL, with a special focus on communication-based methods.

Coordinating multi-agent teams is a challenging computational problem [50, 51, 52, 53, 54, 55, 56, 57]. In multi-agent settings, each agent observes other agents as part of the environment, causing the environment to appear dynamic and non-stationary. Further difficulty arises due to the issue of credit assignment, where it is difficult for each agent to deduce its own contribution to the team's success (especially when there are only global rewards). To solve these multi-agent challenges, many researchers in MARL [58] have pursued centralized training and decentralized execution. Further extensions allow agents to exchange messages during execution, allowing for increased performance. To improve efficiency of MARL, recently, some works are focused on incorporating multi-task learning, transfer learning, or meta-learning into MARL frameworks.

In this chapter, we first depict the background of MARL with centralized critic in section 2.1. Next, section 2.2 introduces and discusses the existing challenges of communication-based MARL. Then, we introduce works that utilize Graph Neural Networks for graph representations in multi-agent problems in section 2.3. In section 2.4, we present recent advances and discuss problems in multi-agent multi-task and transfer reinforcement learning. At last, we introduce multi-agent meta-reinforcement learning in section 2.5.

2.1 MARL with Centralized Critic

Some works extend variants of actor-critic algorithms to multi-agent settings and learn decentralized policy through centralized critics without explicit communication channels [59, 60, 61]. MADDPG [59] is a MARL framework based on Deep Deterministic Policy Gradient, and can be applied in both cooperative and competitive scenarios. COMA [60] extends on-policy actor-critic and proposes a counterfactual baseline to address the credit assignment problem. MAAC [61] is developed from Soft-Actor-Critic [62] and takes advantage of the idea of the counterfactual baseline from COMA. The authors propose a specialized attention mechanism over agents when training the critic, which allows for better scalability as its input space increases linearly, instead of exponentially, with respect to the number of agents. While these works present critical improvements in the field of MARL, the ability to communicate and process information can further increase performance. As we show in subsection 3.6.4, the ability to communicate results in a 88.9% performance gain for our MAGIC.

2.2 MARL with Communication

Recent works have enabled agents to communicate and exchange messages during runtime. Differentiable Inter-Agent Learning (DIAL) [24] builds up limited-bandwidth differentiable discrete communication channels among agents. CommNet [23] extends to a continuous communication protocol designed for fully cooperative tasks. Agents receive averaged encoded hidden states from other agents and use the messages to make informed decisions. However, utilizing a sum or average of messages results in some information loss. IC3Net [22] uses a gating mechanism to enable the agents to decide when to communicate, and thus is amenable to competitive scenarios. However, both IC3Net and CommNet process messages with a simple average. The proper integration of these messages is critically important for communication, as displayed by the performance of our results in section 3.6. TarMAC [6] achieves targeted communication with a signature-based soft-attention mechanism. The integrated signal for each agent is the weighted mean of values generated by all agents. IMAC [26] uses a scheduler to aggregate compact messages through reweighting all agents' messages, and achieves the state-of-the-art performance on multi-agent communication under limited bandwidth. TarMAC and IMAC do not explicitly consider "when" and "whom" or the topology of agent interactions, which can help save communication resources and process messages efficiently. ATOC [63], employs an attention mechanism to decide if an agent should communicate in its observable field. SchedNet [27] proposes a weight-based scheduler to pick agents who should broadcast their messages. However, both ATOC and SchedNet have to manually configure their communication groups. Our communication protocol proposed in chapter 3 intelligently decides when and with whom to communicate through a graph-attention based Scheduler resulting in a large performance gain ($\approx 38\%$ average increase in reward in our most difficult domain) compared to prior work.

2.3 MARL using Graph Neural Networks

Graph Neural Networks (GNNs) are powerful tools for learning from data with graph structures [64, 65, 66, 67]. To model the interactions between agents, MARL has utilized GNNs to allow for a graph-based representation [68, 69, 70, 71]. DGN [69] represents the multi-agent environment as a graph and employ multi-head dot-production attention as the convolutional kernel to extract relational features between agents. MAGNet [70] learns multi-agent policies in the Pommerman game by utilizing a relevance graph and message passing mechanism. The graphs are static and constructed based on heuristic rules. [68] learns a hierarchical topology of the communication structure dynamically by electing central agents. HAMA [72] designs a hierarchical graph attention network to model the hierarchical relationships between agents in both cooperative and competitive scenarios. G2ANet [73] combines a hard-attention and a soft attention mechanism to dynamically learns interactions between agents. In this work, we modify standard graph attention networks to be compatible with a differentiable directed graph, allowing us to represent the interactions among agents more accurately during communication.

In chapter 3, we improve upon prior frameworks by utilizing a Scheduler to solve the problems of when to communicate and whom to address messages to, and a Message Processor using GATs with dynamic directed graphs to integrate and process messages. In this way, we achieve efficient and targeted message sending and high-performance message comprehension.

2.4 Multi-Agent Multi-Task and Transfer Reinforcement Learning

While most literature for multi-task and transfer RL has been focused on single-agent cases, recently GNNs have shown strong generalization capability and superior performance in multi-task and transfer settings for MARL. Within the GNN framework, agents coordinate with each other through message passing. Wang et al. propose a GNN-based method NerveNet, where information over the structure of an agent can be propagated, and then the actions for different parts of the agent can be predicted [40]. Here, different parts of an agent are modeled using a graph, and can be regarded as multiple sub-agents. NerveNet outperforms other methods in a few transfer and multi-task learning tasks, where the tasks vary in agent sizes (number of sub-agents) and components disabled. In this work, they focus on the locomotion control problems of an agent with a time-invariant underlying graph structure for bodies and joints, instead of a real multi-agent cooperative task with dynamic topology. Snowflake [41] addresses the overfitting problems of NerveNet and greatly improves its asymptotic performance on tasks of larger scales (tasks with larger numbers of sub-agents), by freezing the parameters of encoder, decoder, and message function during training. Consequently, the policy can be trained more stably with smaller batch sizes, thus the sample efficiency is also improved. Similarly, Huang et al. propose Shared Modular Policies for agent-agnostic locomotion with a two-way message-passing mechanism [42]. Representing agent morphologies as graphs, Shared Modular Policies can control agents with different skeletal structures, and perform zero-shot generalization to unseen agents from a similar distribution and out-of-distribution agents. Still, the task is limited to locomotion and the topology of agent limbs (sub-agents) is static across time.

Some other works aim to learn adaptable policies with no communication or limited communication. Omidshafiei et al. [74] propose a fully-decentralized two-phase method for partially observable domains, where the first phase enables coordination for single-task MARL and the second phase distills agent-specific action-value networks into a generalized multi-task network. In [75], a tensor and action space with fixed dimensionality is designed to make zero-shot transfer to tasks with a different number of agents and other entities possible. Cui et al. introduce a distributed scalable MARL method for networked traffic control problems, and the trained polices can be successfully transferred with zero shot to larger networks with more agents [76]. Ad-hoc teamwork [77] represents a challenging problem that requires agents can adapt quickly to collaborate with unfamiliar teammates without pre-coordination. Liu et al. investigates the adaptive MARL and communication for ad-hoc team playing by employing a coach with omniscient broadcasting team-level strategies periodically [39]. During training, the tasks are sampled from a fixed set of team compositions, and the trained model can remain high performance with sparse communication for zero-shot generalization on unseen tasks with more agents. Ad-hoc teamwork problems are also explored and solved through reward attribution decomposition [37] and teamwork-conditional marginal utility estimation [38].

The first goal of chapter 4 is to present a communication-based MARL framework that can perform multi-task learning across teams with different numbers of agents, and can be generalized to unseen tasks with different team sizes. Distinguished from the previous works, our communication protocol supports dynamic patterns across timesteps and multiple rounds in the multi-task setting.

2.5 Multi-Agent Meta-Reinforcement Learning

Most meta-reinforcement learning methods are designed for single-agent domains, and can be grouped into three categories by the way of adaptation: Black-box adaptation methods [78, 48, 49], optimization-based methods [36, 79, 80], and the inference-based methods [81, 82, 83].

Existing multi-agent meta-reinforcement learning methods are under exploration, and are mostly optimization-based. In [43], Reptile [84], which performs a first-order approximation of MAML [36], is applied to MADDPG [59] to achieve meta-reinforcement learning (meta-RL) for Trans-Conditions and Trans-Scenarios tasks. This approach attempts to transfer the meta knowledge from single-agent tasks to a simple three-agent cooperative navigation task. Hu et al. propose a distributed value-decomposition-based RL solution to the problem of the trajectory design for drone base stations [45]. MAML is employed by each agent's individual policy and value networks for different user request realizations. Dif-MAML [85] proposes a decentralized MAML-based meta-RL method that enables distributed agents to explore tasks stemming from potentially different task distribution. It can match the performance of a centralized solution while enjoying a few advantages of decentralized methods such as robustness and privacy. Huang et al. [86] treat each agent as an independent task, and design a meta actor-critic with a two-layer optimization process to improve the performance of MADDPG [59] and MATD3 [87]. Some works use recurrent structures like LSTM [88] to generate guidance for the optimization processes of multiple agents. In MAMRL [46], meta-agent acts as a optimizer that learns to optimize the networks of local agents (optimizee), using LSTM that is inspired from [44]. The local agents in this architecture are distributed and each local agent interacts with an individual energy environment. Zhang et al. employ a LSTM structure to provide the update direction for each agent's communication network which determines the agent-agent communication topology [47]. Meta-PG [89] models the nonstationary environment as a sequence of stationary tasks, and the agent can meta-learn to anticipate the changes and update when playing against an opponent that changes its strategy incrementally. Meta-MAPG [90] extends Meta-PG by modeling the learning process of all agents and adding peer learning gradients. A very recent work falls into the category of inference-based methods [91]. They use the history information of communication message to infer the task-specific communication pattern. However, none of the previous work explores using a black-box model such as RNN to adapt to new tasks for multi-agent meta-RL. And in most of the above works, the number of agents across different multi-agent tasks are limited to keeping fixed, while many real-world problems require the model to be deployed in size-variant and scalable teams.

To fill the gaps in existing research mentioned above, the second goal of chapter 4 is to propose and evaluate a RNN-based multi-agent meta-RL method that can be meta-trained and meta-tested on tasks with different team sizes.

CHAPTER 3
MAGIC: MULTI-AGENT GRAPH-ATTENTION COMMUNICATION

3.1 Introduction

In this chapter, we propose Multi-Agent Graph-attentIon Communication (MAGIC) [1], a novel graph communication protocol that determines "when" and "whom" with to communicate via an end-to-end framework. We set a new state-of-the-art in communication-based multi-agent reinforcement learning by modeling the topology of interactions among agents (the local and global characterization of connections between agents [93]) as a dynamic directed graph that accommodates time-varying communication needs and captures the relations between agents. Our proposed framework emulates the features of an effective human-human team through its key components, 1) the Scheduler, which helps each agent to decide when it should communicate and whom it should communicate with, and 2) the Message Processor, which integrates and processes received messages in preparation for decision making. We find MAGIC produces high-performance, cooperative behavior through its efficient communication protocol.

Our Scheduler consists of a graph attention encoder and a differentiable hard attention mechanism to decide when to communicate and whom to communicate with. This information is encoded within a directed graph, allowing us to represent the interaction among agents precisely. The Message Processor, consisting of a Graph Attention Network (GAT), utilizes received messages and the directed graph to intelligently and efficiently process messages. The encoded messages are then used in each agent's policy, leading to high-performance cooperation and efficient communication, as shown in section 3.6. We

[1]This work was published by ACM at the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS) [92]. It also won the best paper award at Multi-Agent Interaction and Relational Reasoning Workshop in ICCV 2021.

provide the following detailed contributions:

1. Develop a novel graph-attention communication protocol for MARL that utilizes 1) a Scheduler to solve the problems of when to communicate and whom to address messages to, and 2) a Message Processor using GATs with dynamic directed graphs to integrate and process messages.

2. Enable GATs in the Message Processor to maintain gradients from graph-based operations, which is not supported by standard GATs. In this way, the framework is fully differentiable and can be trained in an end-to-end manner.

3. Outperform prior methods across three domains, including the Google Research Football environment, achieving a 10.5% increase in reward. Further, MAGIC learns to communicate 27.4% more efficiently than the average baseline. These results set a new state-of-the-art in communication-based MARL.

4. Demonstrate our algorithm on physical robots in a 3-vs.-2 soccer scenario on a physical, multi-robot testbed.

In this chapter, we start with section 3.2 by introducing preliminaries of Partially Observable Markov Game, Policy Gradients and Graph Neural Networks. Next, we propose and describe our method MAGIC in section 3.3. Then, section 3.4 and section 3.5 provide details in evaluation environments and training, respectively. In section 3.6, we present and discuss the experiment results. Then, a physical robot demonstration is shown in section 3.7. At last, we conclude this chapter in section 3.8.

3.2 Preliminaries

3.2.1 Partially Observable Markov Game

A Markov Game [94] is the multi-agent version of Markov Decision Process (MDP). We are primarily concerned with a partially observable Markov game. A partially observable

Markov game (POMG) for N agents can be defined by a set of global states, S, a set of private observations for each agent, $O_1, O_2, \ldots, O_N$, a set of actions for each agent, $A_1, A_2, \ldots, A_N$, and the transition function, $T : S \times A_1 \times \ldots \times A_N \mapsto S$. In each time step, agent i chooses action, $a_i \in A_i$, obtains reward as a function of state, S, and its action $r_i : S \times A_i \mapsto \mathbb{R}$, and receives a local observation $o_i : S \mapsto O_i$. The initial state is defined by a initial state distribution ρ. Agent i aims to maximize its discounted reward $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$, where $\gamma \in [0, 1]$ is a discounted factor. Our work is based on the framework of POMG augmented with communication.

3.2.2 Reinforcement Learning: Policy Gradients

The Policy Gradient method (Equation 3.1) is widely used in reinforcement learning (RL) tasks to perform gradient ascent on the agent policy parameters, θ, to optimize the total discounted reward, $J(\theta) = \mathbb{E}_{s \sim p^{\pi}, a \sim \pi_\theta}[R]$. p^{π} is the state distribution, π_θ is the policy distribution, and $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r(s_{t'}, a_{t'})$.

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim p^{\pi}, a \sim \pi_\theta} \big[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t | s_t) R_t \big] \tag{3.1}$$

In lieu of R_t, we use the advantage function, $A^{\pi}(s_t, a_t) = R_t - V(s_t)$, to decrease the variance of the estimated policy gradient, where $V(s_t)$ is the value function.

3.2.3 Graph Neural Networks

In Graph Neural Networks (GNNs), each GNN layer computes the node representation by message passing, where each node aggregates the feature vectors from its neighboring nodes in the graph at the previous layer. The update rule for node representations by a GNN layer is displayed in Equation 3.2.

$$h_i^{(l)} = \sigma \left(\sum_{j \in N_i} \frac{1}{\sqrt{d_i d_j}} (h_j^{(l-1)} W^{(l)}) \right) \tag{3.2}$$

Here, $h_i^{(l)}$ represents the features of node i, at layer l. N_i is the set of neighboring nodes of node i, $d_i = |N_i|$ is the degree of node i, $W^{(l)}$ is a learnable weighting matrix for layer l, and $\sigma(\cdot)$ is a nonlinear activation function. In this work, we enable our GATs within the Message Processor to function with differentiable graphs (subsection 3.3.3), allowing for end-to-end training.

3.3 Method

In this section, we introduce our proposed Multi-Agent Graph-attentIon Communication protocol, MAGIC. We consider a partially observable setting of N agents, where agent i receives local observation, o_i^t, at time, t, containing local information from the global state, S. The agent, i, learns a communication-based policy, π_i, to output a distribution over actions, $a_i^{(t)} \sim \pi_i$, at each time step, t. Here, we present an overview of our framework, the description of our protocol's key components (i.e., the Scheduler and Message Processor), and our training procedure.

3.3.1 Overview

Our proposed graph-attention communication protocol is displayed in Figure 3.1. At time step, t, the observation for each agent, o_i^t, is first encoded using an agent-specific fully-connected layer (FC). The encoded observation is passed into an agent-specific LSTM cell to generate a hidden state, h_i^t, as shown in Equation 3.3.

$$h_i^t, c_i^t = LSTM(e(o_i^t), h_i^{t-1}, c_i^{t-1}) \tag{3.3}$$

In this equation, c_i^t is the cell state for agent, i, at time step, t, and $e(\cdot)$ is a fully-connected layer acting as an encoder for the observation. The hidden state, h_i^t, is then encoded as a message, $m_i^{t(0)} = e_m(h_i^t)$, through the encoder, $e_m(\cdot)$ (a fully-connected layer). Here, the exponent notation for the message, $m_i^{t(0)}$, denotes that message is for agent i, and is prior to any message aggregation or processing. We refer to this stage, where the message has

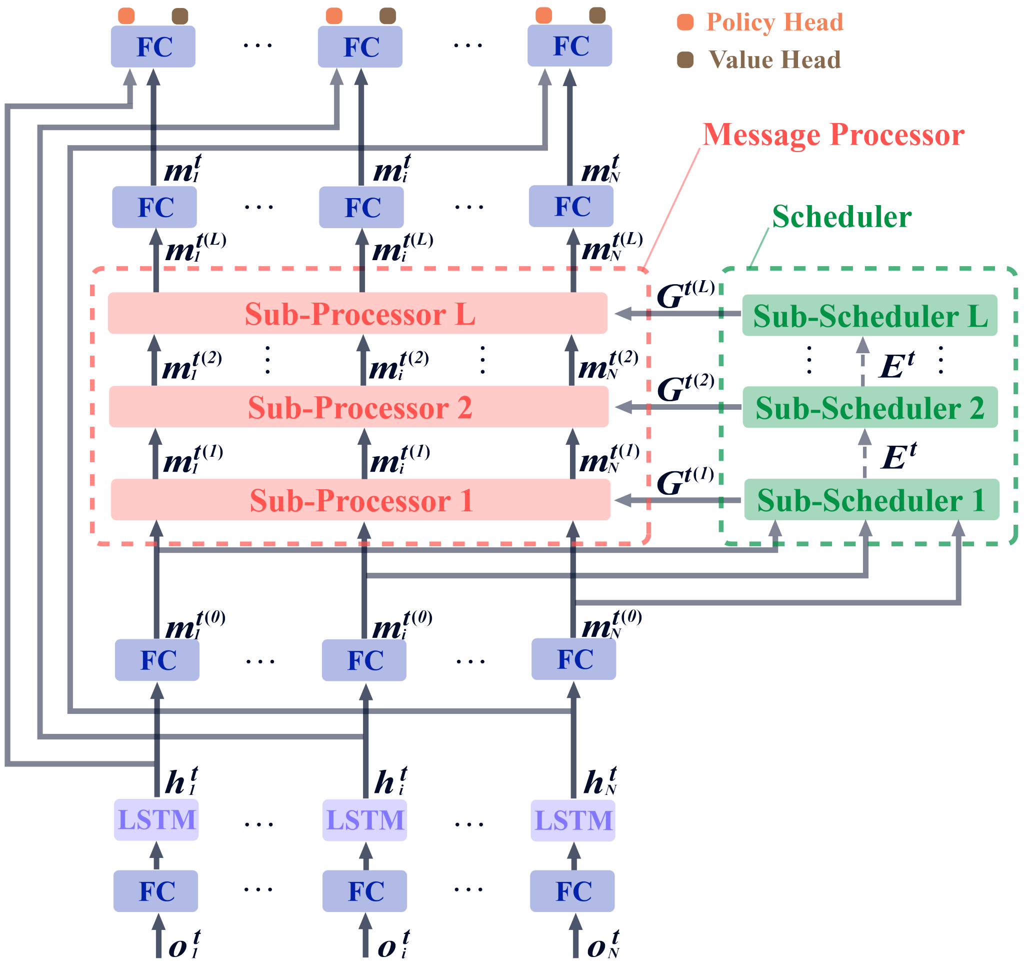


Figure 3.1: This figure displays the framework of our multi-agent graph-attention communication protocol.

not been processed, as round 0, giving the exponent notation, $t(0)$.

As shown in Figure 3.1, we define the function module to help agents decide whom to send messages at each time step as the "Scheduler" and define the function module to process messages as "Message Processor." The Scheduler and the Message Processor may include multiple sub-schedulers and sub-processors, respectively. Prior work has termed the procedure of processing messages for multiple iterations as multi-round communication [6]. As multi-round communication has been shown to improve performance, our protocol supports L rounds of communication, where $L \in \mathbb{N}$. A round of communication, l, is defined as a forward pass through a sub-scheduler and sub-processor. As shown in Figure 3.1, the encoded messages, $m_i^{t(0)}$ are passed into Sub-Scheduler 1 and Sub-Processor 1 (i.e., the

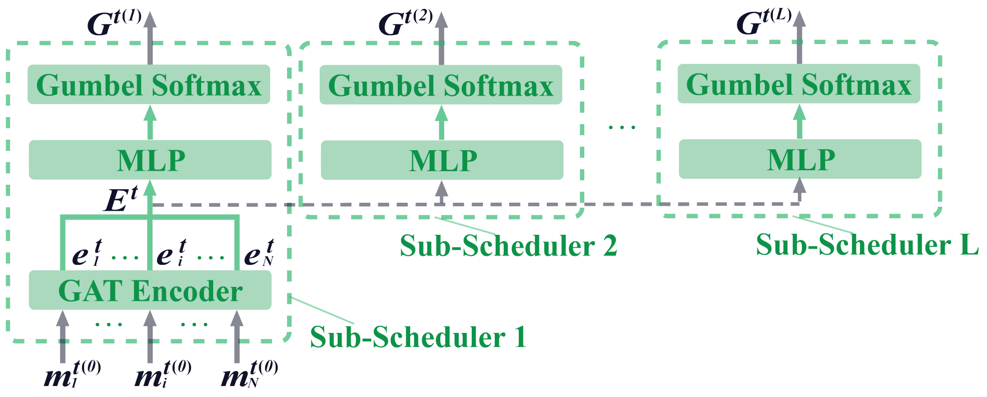


Figure 3.2: This figure displays the details and components of the Scheduler.

sub-scheduler and sub-processor at round 1).

The Sub-Scheduler l (at round, $l \in L$) will output an adjacency matrix, $G^{t(l)}$. $G^{t(l)}$ is a directed graph that indicates the targeted receivers for each agent at time step, t. $G^{t(l)}$ is utilized by the Sub-Processor, l, to produce a set of integrated messages, $\{m_i^{t(l)}\}_1^N$, where $m_i^{t(l)}$ is the integrated message for agent, i, at time step, t. The integrated messages for each agent, i, can be incorporated into agent i's policy (in the case where we are on the last round of communication, $l = L$) or be further processed by more rounds of communication ($l < L$). If the messages are to be further processed, the set of messages outputted from round l, $\{m_i^{t(l)}\}_1^N$, are passed into the Sub-Scheduler $l + 1$ and the Sub-Processor $l + 1$, producing adjacency matrix $G^{t(l+1)}$ and messages $\{m_i^{t(l+1)}\}_1^N$ respectively.

The message outputted from the Message Processor, $m_i^{t(L)}$, for agent, i, is encoded through a fully-connected layer, $e'_m(\cdot)$, to produce an intelligently integrated message, $m_i^t = e'_m(m_i^{t(L)})$. m_i^t is concatenated with the hidden state, h_i^t, to produce the input feature to the policy head and the value head. The policy head is a fully-connected layer followed by a softmax function. We sample the action for the i-th agent at time step, t, from the policy output distribution: $a_i^t \sim \pi_i(a_i^t|o_i^t)$. The value head is a single fully-connected layer and serves as a baseline function.

3.3.2 The Scheduler

The Scheduler decides when each agent should send messages and whom each agent should address messages to, as shown in Figure 3.2. As a black-box, the Scheduler takes as input the encoded messages, $\{m_i^{t(0)}\}_1^N$, and outputs the directed graphs, $\{G^{t(l)}\}_1^L$, as represented in $f_{Sched}(\cdot)$ shown in Equation 3.4.

$$\{G^{t(l)}\}_1^L = f_{Sched}\left(m_1^{t(0)}, \cdots, m_N^{t(0)}\right) \tag{3.4}$$

As noted in subsection 3.3.1, the Scheduler consists of L Sub-Schedulers, each producing an adjacency matrix $G^{t(l)}$. A Sub-Scheduler consists of a GAT encoder and a hard attention mechanism that uses a multi-layer perceptron (MLP) and a Gumbel Softmax function [95]. The GAT encoder helps encode local or global information for an agent efficiently, and it is only used in the first Sub-Scheduler. We adopt the same form of GATs as proposed in [96], where the attention mechanism is expressed in Equation 3.5.

$$\alpha_{ij}^S = \frac{exp\left(LReL\left(a_S^T[W_S m_i^{t(0)} || W_S m_j^{t(0)}]\right)\right)}{\sum_{k \in N_i^t \cup i} exp\left(LReL\left(a_S^T[W_S m_i^{t(0)} || W_S m_k^{t(0)}]\right)\right)} \tag{3.5}$$

Here, $LReL(\cdot)$ is the LeakyReLU [97], $a_S \in R^{D'}$ is a weighting vector, $N_i^t \cup i$ is the set of neighboring agents for the i-th agent, including agent i, at time step, t, and $W_S \in R^{D' \times D}$ is a weighting matrix, where D' and D refer to the output feature cardinality and message cardinality, respectively. The node features of each agent are obtained through Equation 3.6.

$$e_i^t = ELU\left(\sum_{j \in N_i^t \cup i} \alpha_{ij}^S W_S m_j^{t(0)}\right) \tag{3.6}$$

Here, $ELU(\cdot)$ is the Exponential Linear Unit (ELU) function. After concatenating the node feature vectors pairwise, supposing $E_{i,j}^t = (e_i^t || e_j^t)$, we obtain a matrix $E^t \in \mathbb{R}^{N \times N \times 2D}$, where $E_{i,j}^t$ represents a high-level representation of relational features between the i-th and j-th agent.

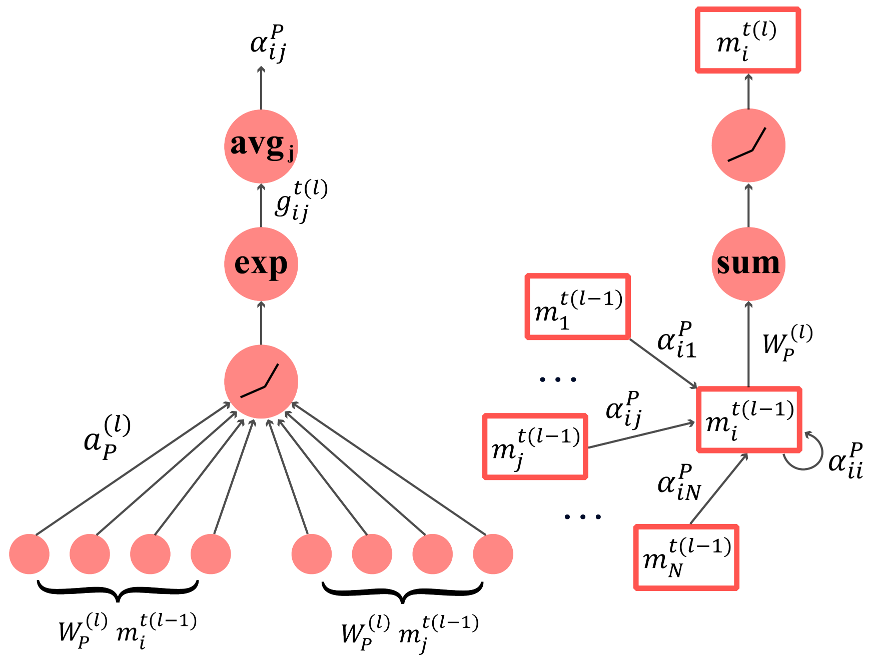


Figure 3.3: This figure displays the detailed structure of a Sub-Processor.

Setting E^t as the input to an MLP followed by a Gumbel Softmax function, we can get an adjacency matrix $G^{t(l)}$, consisting of binary values, representing a directed graph. If the element $g_{ij}^{t(l)}$ in $G^{t(l)}$ is 1, the j-th agent will send a message to i-th agent. Otherwise ($g_{ij}^{t(l)} = 0$), the j-th agent will not send any message to i-th agent.

3.3.3 The Message Processor

The Message Processor helps agents integrate messages for intelligent decision making. As a black-box, it takes in the encoded messages, $\{m_i^{t(0)}\}_1^N$, and the graphs generated by the Scheduler, $G^{t(1)} \cdots, G^{t(L)}$, and outputs the processed messages, $\{m_i^{t(L)}\}_1^N$. We represent the Message Processor in Equation 3.7.

$$\{m_i^{t(L)}\}_1^N = f_{MP}\left(m_1^{t(0)}, \cdots, m_N^{t(0)}, G^{t(1)}, \cdots, G^{t(L)}\right) \tag{3.7}$$

As stated in subsection 3.3.1, the Message Processor consists of L Sub-Processors, each producing a set of encoded messages, $\{m_i^{t(l)}\}_1^N$. A Sub-Processor, for a single round of communication, includes a designed GAT layer receiving messages, $\{m_i^{t(l-1)}\}_1^N$, from all agents and the adjacency matrix, $G^{t(l)}$, as input, as shown in Figure 3.3. The Sub-Processor helps agents process received messages. In Equation 3.8, we display the calculation of the attention coefficient in our designed GAT layer, which maintains the gradient from the Scheduler.

$$\alpha_{ij}^{P} = \frac{g_{ij}^{t(l)} exp\left(LReL\left((a_P^{(l)})^\top [W_P^{(l)} m_i^{t(l-1)} || W_P^{(l)} m_j^{t(l-1)}]\right)\right)}{\sum_{k=1}^{N} g_{ik}^{t(l)} exp\left(LReL\left((a_P^{(l)})^\top [W_P^{(l)} m_i^{t(l-1)} || W_P^{(l)} m_k^{t(l-1)}]\right)\right)} \tag{3.8}$$

Here, l is the round of communication, $g_{ij}^{t(l)} \in \{0,1\}$is a binary value in the adjacency matrix, $G^{t(l)}$, $W_P^{(l)} \in R^{D'' \times D}$is a weighting matrix, and $a_P^{(l)} \in R^{D''}$is a weighting vector. It should be noted that our graphs are capable of a self-loop, where an agent will "send" a message to itself, and use its own message in the integration of received messages. While the calculation of the coefficient for a standard GAT layer is a non-differentiable operation for the graph, using Equation 3.8 allows us to retain the gradient of $g_{ij}^{t(l)}$. Thus, the Scheduler can preserve the gradient flow for end-to-end training, avoiding the need to design an extra loss function to train the Scheduler. In practice, we find using multi-head attention [96] and adding a bias to the output message to be beneficial. The output message of sub-processor l can be obtained via Equation 3.9.

$$m_i^{t(l)} = ELU\left(\sum_{j=1}^{N} \alpha_{ij}^{P} W_P^{(l)} m_j^{t(l-1)}\right) \tag{3.9}$$

3.3.4 Training

In our experiments, the parameters of the fully-connected layers and LSTM in the policy network are shared across homogeneous agents to improve training efficiency. We employ a multi-threaded synchronous multi-agent policy gradient [22] and utilize an extra value head in the policy network to estimate the value function, $V_\phi(o_i^t)$, at observation o_i^t, which will serve as a state-independent baseline. In addition to optimizing the discounted total reward with policy gradient, the model also minimizes the squared error between the estimated value and the Monte-Carlo estimate. The two loss functions are balanced by a coefficient, β. We define the overall loss function as $\mathcal{L}(\cdot)$ and the policy function denoted as $\pi_\theta(a_i^t|o_i^t)$. Parameters, θ, of the policy and, ϕ, of the value function, share most of their parameters except the parameters in the policy and value heads. Our model is updated via minimizing the loss function displayed in Equation 3.10.

$$\begin{aligned}\nabla_{\theta,\phi}\mathcal{L}(\theta,\phi) = \frac{1}{t_{max}}\sum_{i=1}^{N}\sum_{t=1}^{t_{max}}[-\nabla_\theta \log \pi_\theta(a_i^t|o_i^t)(R_i^t - V_\phi(o_i^t)) \\ +\beta\nabla_\phi(R_i^t - V_\phi(o_i^t))^2]\end{aligned} \tag{3.10}$$

Here, R_i^t is the discounted total reward for agent i starting from time step t in an episode, and t_{max} is the number of steps taken within a batch. Different threads in training share the parameters, θ, and ϕ, and calculate their own gradients. The threads synchronously accumulate gradients and update θ and ϕ within each batch. A summary of the training procedure of our multi-agent graph-attention communication model and the algorithm is described in Algorithm 1. In Algorithm 1, we start by initializing the number of agents alongside several training parameters, including threads, batch size, maximum steps in an episode, and maximum steps in a batch, shown in Step 1. For each update per thread, we start by initializing a set of thread parameters and a replay buffer, D, as displayed in Step 4. After receiving the initial hidden state and observation for each agent (shown in Steps 7 and 8), we can utilize the Scheduler (containing L sub-schedulers) to output adjacency graphs, determining the communication pattern. The Message Processor (containing L sub-processors) can use these graphs and the encoded messages from round zero to produce messages for each agent, as shown in Steps 13 and 14. Once each agent has its selected message inputs, we can determine an action probability distribution from the policy and perform the action sampled from this distribution, shown in Steps 15 and 16. Storing this information in our replay buffer, we can then complete the episode and proceed to compute gradients with Equation 3.10, as shown in Step 26. Accumulating gradients across threads, we can update our policy and value function, as shown in Steps 29 and 31.

Algorithm 1 Training Multi-Agent Graph-attentIon Communication (MAGIC)

```
 1: Initialize max updates $M$, agents $N$, threads $L$, max steps in an episode $T_e$, max steps
    in a batch $T_b$
 2: for update $= 1$ to $M$ do $d\theta \leftarrow 0$, $d\phi \leftarrow 0$
 3:   for thread $k = 1$ to $K$ do in parallel
 4:     Initialize params $\theta_k \leftarrow \theta$, $\phi_k \leftarrow \phi$, buffer $D$, step-count $t \leftarrow 1$
 5:     while $t < T_b$ do
 6:       Initialize thread step counter $t' \leftarrow 1$
 7:       Initialize $h_i^{t-1}$, $c_i^{t-1}$ for each agent $i$
 8:       Reset environment and get $o_i^t$ for each agent $i$
 9:       while $t' < T_e$ and not terminal do
10:         $h_i^t, c_i^t = LSTM(e(o_i^t), h_i^{t-1}, c_i^{t-1})$ for each agent $i$
11:         $m_i^{t(0)} = e_m(h_i^t)$ for each agent $i$
12:         $\{G^{t(l)}\}_1^L = f_{Sched}\left(m_1^{t(0)}, \cdots, m_N^{t(0)}\right)$
13:         $\{m_i^{t(L)}\}_1^N = f_{MP}\left(m_1^{t(0)}, \cdots, m_N^{t(0)}, G^{t(1)}, \cdots, G^{t(L)}\right)$
14:         $m_i^t = e'_m(m_i^{t(L)})$ for each agent $i$
15:         Calculate $\pi_{\theta_k}(a_i^t|o_i^t)$ and $V_{\phi_k}(o_i^t)$ for each agent $i$
16:         Perform $a_i^t \sim \pi_{\theta_k}(a_i^t|o_i^t)$ for each agent $i$
17:         Receive $r_i^t$ and $o_i^{t+1}$ for each agent $i$
18:         Store $(o_i^t, a_i^t, \pi_{\theta_k}(a_i^t|o_i^t), V_{\phi_k}(o_i^t), r_i^t, o_i^{t+1})$ in $D$
19:         $t \leftarrow t + 1, t' \leftarrow t' + 1$
20:       end while
21:     end while
22:     $t_{max} \leftarrow t$
23:     for $t = t_{max}, t_{max} - 1, \cdots, 1$ do
24:       $R_i^t = 0$ if $o_i^{t+1}$ is terminal else $R_i^t = r_i^t + \gamma R_i^{t+1}$ using $D$
25:     end for
26:     Calculate $d\theta_k$ and $d\phi_k$ using $D$ with Equation 3.10
27:   end for
28:   for thread $k = 1$ to $K$ do
29:     Accumulate gradients: $d\theta \leftarrow d\theta + d\theta_k$, $d\phi \leftarrow d\phi + d\phi_k$
30:   end for
31:   Perform update of $\theta$ using $d\theta$, and of $\phi$ using $d\phi$
32: end for
```

3.4 Evaluation Environments

We utilize three domains, including the Predator-Prey [22], Traffic Junction [23] and complex Google Research Football environment [98], to evaluate the utility of proposed communication protocol. Predator-Prey and Traffic Junction are common MARL benchmarks [23, 6]. Google Research Football (GRF) presents a difficult challenge, as it has sparse rewards, stochasticity, and adversarial agents.

3.4.1 Predator-Prey

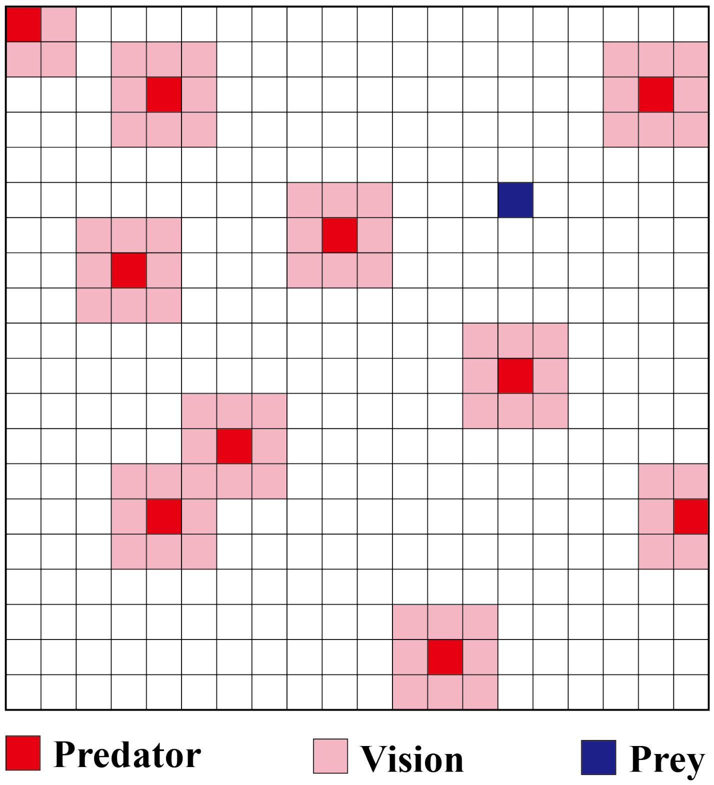


Figure 3.4: The visualization of the 10-agent Predator-Prey task. The predators (in red) with limited visions (light red region) of size 1 are searching for a randomly initialized fixed prey (in blue).

We utilize the predator-prey environment from [22]. Here, there are N predators with limited visions searching for a stationary prey. A predator or a prey occupies a single cell within the grid world at any time, and its location is initialized randomly at the start of each episode. The state at each point in the grid is the concatenation of a one-hot vector which represents its own location and binary values indicating the presence of predator and prey at this point. The observation of each agent is a concatenated array of the states of all points within the agent's vision. The predators can take actions up, $down$, $left$, $right$ or $stay$. We utilize the 'mixed' mode of Predator-Prey in which the predator incurs a reward -0.05 for each time step until the prey is found. An episode is defined as successful if all the predators find the prey before a predefined maximum time limit. We test two levels of difficulty in this environment. The difficulty varies as the grid size, and the number of predators increases, as more coordination is required to achieve success. The corresponding grid sizes and the number of predators are set to 10×10 with 5 predators and 20×20 with 10 predators. The 10-agent task is shown in Figure 3.4. We set the maximum steps for an episode (i.e., termination condition) to be 40 and 80, respectively. The vision is set to a unit length. We define a higher-performing algorithm in this domain as one that minimizes the average steps to complete an episode.

3.4.2 Traffic Junction

The second domain we utilize is the Traffic Junction environment. This environment, composed of intersecting routes and cars (agents) with limited vision, requires communication to avoid collisions. Cars enter the traffic junction from all entry points at each time step with a probability p_{arrive}, and are randomly assigned a route at the start. The maximum number of cars in the environment at a specific time is denoted by N_{max}, which varies across difficulty levels. A car occupies one cell at a time step and can take action "gas" or "brake" on its route. The state of each cell is the concatenation of a one-hot vector representing its location, and a value indicating the number of cars in this cell. The observation of each car is the concatenation of its previous action, route identifier, and all states of the cells within its vision. Two cars collide if they are in the same location, resulting in a reward of -10 for each car. The simulation terminates once all agents reach the end of its route

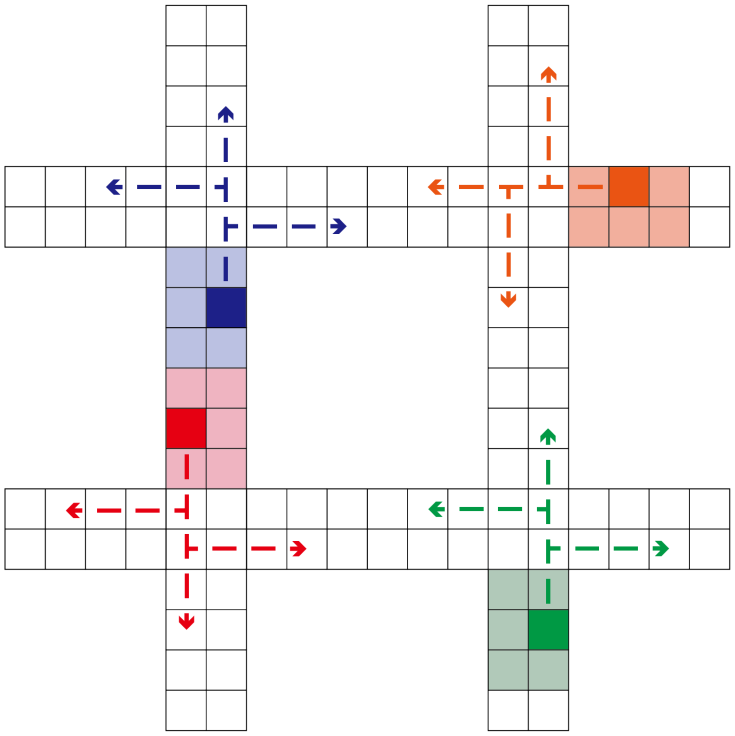


Figure 3.5: The visualization of the hard level Traffic Junction task. This task consists of four, two-way roads on a 18×18 grid with eight arrival points, each with seven different routes. Each agent is with a limited vision of size 1.

or if the time surpasses the predefined timeout parameter. Collisions will not incur "death" of agents or terminate the simulation. The agents will only be "dead" when it reaches the end of its route. There is a time penalty -0.01τ at each time step, where τ is the number of time steps that have passed since the agent's entry. An episode is considered successful if there are no collisions within the episode.

We validate our algorithm on three difficulty levels. The easy level consists of two, one-way roads on a 7×7 grid. There are two arrival points and two possible routes for each arrival point, and at most five agents ($N_{max} = 5$, $p_{arrive} = 0.3$). For the medium level, the junction consists of two, two-way roads on a 14×14 grid with four arrival points, each with three different routes. Here, there are at most ten agents ($N_{max} = 10$, $p_{arrive} = 0.2$). The hard level, as shown in Figure 3.5, consists of four, two-way roads on a 18×18 grid with eight arrival points, each with seven different routes, and there are at most twenty agents ($N_{max} = 20$, $p_{arrive} = 0.05$). The average success rate (i.e., no collisions within an episode) is used in our evaluation. We set the limited vision parameter to 1 for both levels. Similar to [22], in Traffic Junction, we fix the gating action to be 1 for IC3Net and TarMAC-IC3Net, set all the hard attention outputs in GA-Comm to be 1, and set all the graphs used by the Message Processor in our method to be complete.

3.4.3 Google Research Football

Our final domain of Google Research Football [98] presents a challenging, mixed cooperative-competitive, multi-agent scenario with high stochasticity and sparse rewards. Google Research Football (GRF) is a physics-based 3D soccer simulator for reinforcement learning. This domain presents an additional challenge as there are opponent artificial agents (AIs), significantly increasing the complexity of the state-action space. We present a depiction of this environment in Figure 3.6. To align with the partially observable setting, we extract the local observations from the provided global observations. The local observations include the relative positions of the players on both teams, the relative position of the ball, and

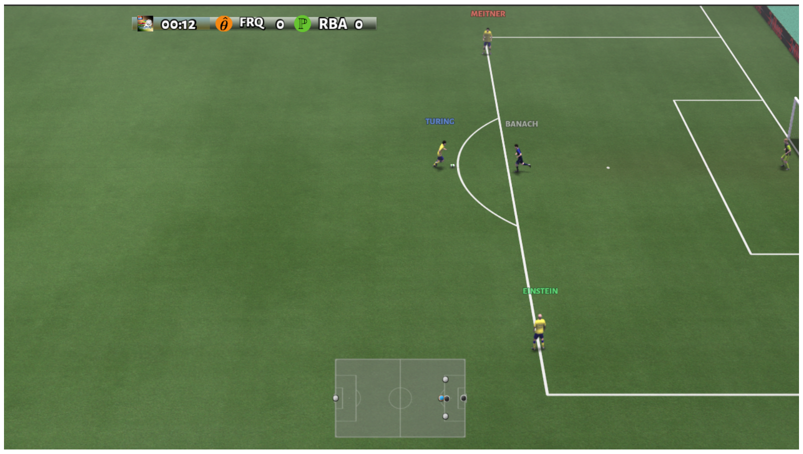


Figure 3.6: The visualization of 3 vs. 2 in Google Research Football. The five people shown in this figure are three offending players, one defending player and the goalie (left to right).

one-hot encoding vectors which represent the ball-owned team and the game mode. GRF provides 19 actions including moving actions, kicking actions, and other actions such as dribbling, sliding and sprint. GRF provides several pre-defined reward signals, consisting of a scoring and a penalty box proximity reward. The penalty box proximity reward is shaped to push attackers to move forward towards certain locations. Many MARL frameworks have required these highly shaped rewards functions to perform well [98]. However, we choose to use only the scoring reward to verify the ability of our algorithm and baselines to function in a high-complexity stochastic domain with sparse rewards. Accordingly, the only reward all agents will receive in our evaluation is $+1$ when scoring a goal. The termination criterion is the team scoring, ball out of bounds, or possession change. We evaluate algorithms in a standard scenario 3 vs. 2 from Football Academy [98], as shown in Figure 3.6, where we have 3 attackers vs. 1 defender, and 1 goalie. The three offending agents are controlled by the MARL algorithm, and the two defending agents are controlled by a built-in AI. We find that utilizing a 3 vs. 2 scenario challenges the robustness of MARL algorithms to stochasticity and sparse rewards. In this domain, we seek to maximize the average success rate (i.e., a goal is scored) and minimize the average steps taken to complete an episode, thereby scoring a goal in the shortest amount of time. We show in subsection 3.6.3 that our method outperforms all prior state-of-the-art baselines.

3.5 Training Details

We distribute the training over 16 threads and each thread runs batch learning with a batch size of 500. The threads share the parameters of the policy network and update synchronously. There are 10 updates in one epoch. We use RMSProp with a learning rate of 0.001 in all the domains except the ten-agent one in Predator-Prey, where we use 0.0003. The value coefficient, β, and discount factor, λ, are set to 0.01 and 1 respectively. The size of each agent's hidden state for LSTM is 128. The sizes of original encoded messages and the final messages for decision making are 128. 2 or 3 layers of GNNs have been used in practice and shown to work well [96]. Empirically, we find that two rounds of communication achieve the best performance with comparable training speeds to simpler methods such as CommNet and IC3Net. As such, we use two rounds of communication to test the performance of our method in all domains, and the number of heads for the first GAT layer (sub-processor 1) is set to be 4, 4, 1 in Predator-Prey, Traffic Junction and GRF respectively, and the number of heads for the output GAT layer (sub-processor 2) is set to be 1. We use one round of communication for efficiency evaluation for fair comparison, and the number of heads for the GAT layer is 1. The output size of the GAT encoder in the Scheduler is set to 32. We implement our method and baselines on each task over 5 random seeds and average the results.

3.6 Results and Discussion

In this section, we evaluate the performance of our proposed method on three environments, including Predator-Prey [22], Traffic Junction [23], and Google Research Football [98]. We benchmark our approach against a variety of state-of-the-art communication-based MARL baselines, including CommNet [23], IC3Net [22], GA-Comm [73], and TarMAC-IC3Net [6]. We implement our method and baselines on each task, averaging the best performance at convergence over 5 random seeds. Following an analysis of performance, we evaluate MAGIC's communication efficiency, concluding MAGIC presents a new state-of-the-art in both performance and efficiency in communication-based MARL. We provide additional training details within section 3.5 of the supplementary.

3.6.1 Predator-Prey

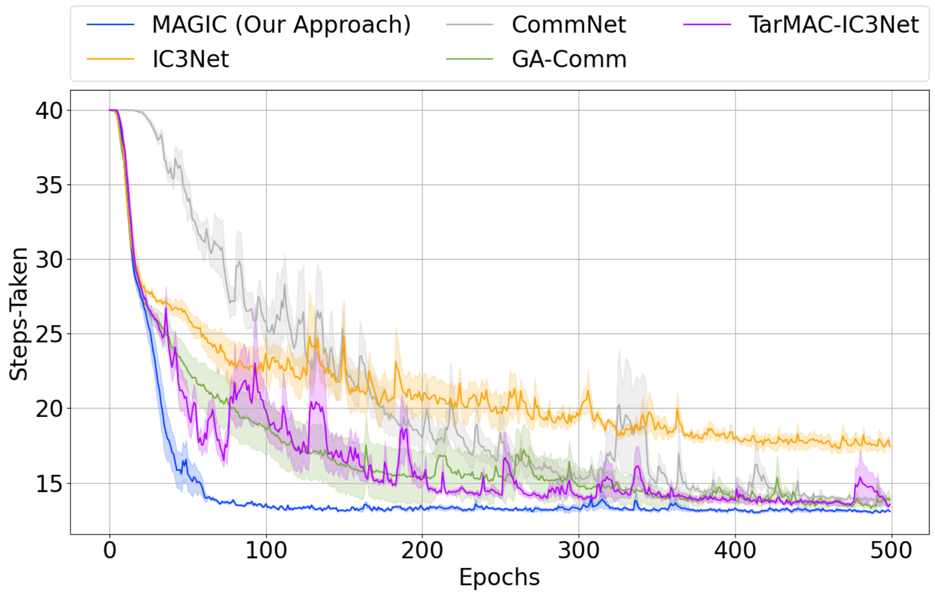
Figure 3.7 depicts the average steps taken for the predators to locate the prey. In both the

Table 3.1: The number of steps taken to complete an episode at convergence in Predator-Prey.

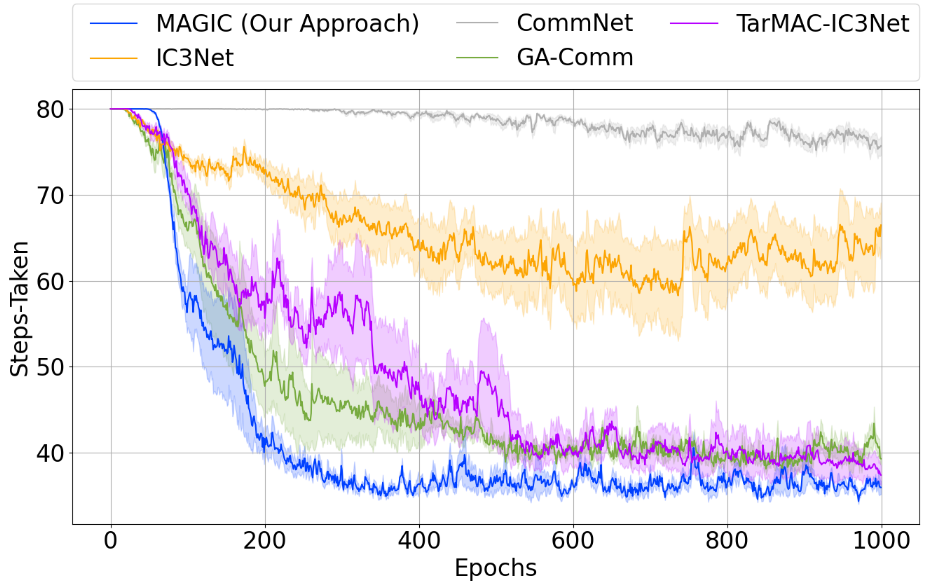
Method	10×10, 5 agents	20×20, 10 agents
MAGIC (Our Approach)	$\mathbf{12.72 \pm 0.03}$	$\mathbf{32.88 \pm 0.14}$
CommNet [23]	13.16 ± 0.04	73.12 ± 0.68
IC3Net [22]	15.60 ± 0.35	55.13 ± 4.80
TarMAC-IC3Net [6]	13.32 ± 0.11	36.16 ± 0.97
GA-Comm [73]	13.06 ± 0.09	35.78 ± 0.37

five- and ten-agent cases, our method converges faster and can achieve better performance than the baselines. Our method converges 52% faster than the next-quickest baseline while still achieving the highest performance. Table 3.1 shows the results of average steps taken to reach the prey at convergence. While approaches such as GA-Comm and TarMAC-IC3Net learn competitive policies for the five agent case, these benchmarks perform much worse than our algorithm in the ten agent case, suggesting a lack of scalability.

Predator-Prey Communication Heatmaps - Figure 3.8 depicts communication heatmaps in Predator-Prey domain with 10 agents in an episode of 31 steps. The color is associated with the probability of communication, with darker colors representing more intensive communication between the two agents. The vertical axis represents message receivers, and the horizontal axis represents message senders. Agent 5 first reaches the prey at step 23, and the other 9 agents quickly reach the prey in the following 7 steps. Figure 3.8(a) displays the communication before the first agent (agent 5) reaches its prey and Figure 3.8(b) displays the communication afterwards. We can see that agents communicate with each

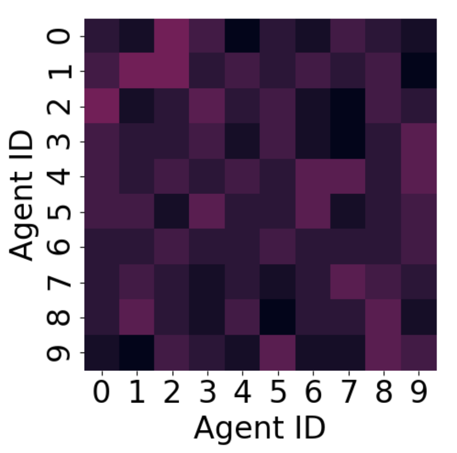


(a) Low difficulty Predator-Prey Environment with size 10×10 and 5 agents

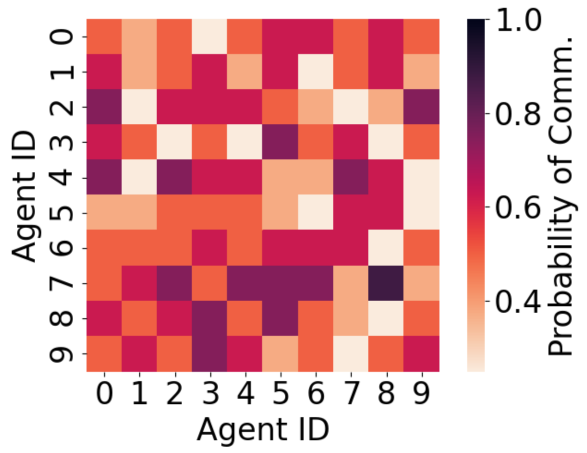


(b) High difficulty Predator-Prey Environment with size 20×20, 10 agents

Figure 3.7: This figure displays the average steps taken to finish an episode as training proceeds in each level of the Predator-Prey environment. The shaded regions represent standard errors. A lower value for steps taken on the vertical axis is better.



(a) Before prey found.



(b) After prey found.

Figure 3.8: Heatmaps of the communication graphs learned by the Scheduler in the Predator-Prey domain.

Table 3.2: The success rate at convergence in Traffic Junction.

Method	7×7, $N_{max} = 5$, $p_{arrive} = 0.3$	14×14, $N_{max} = 10$, $p_{arrive} = 0.2$	18×18, $N_{max} = 20$, $p_{arrive} = 0.05$
MAGIC (Our Approach)	$\mathbf{99.9 \pm 0.1}$ **%**	$\mathbf{99.9 \pm 0.1}$ **%**	$\mathbf{98.0 \pm 0.8}$ **%**
CommNet [23]	$99.3 \pm 0.6\%$	$97.2 \pm 0.3\%$	$66.7 \pm 1.6\%$
IC3Net [22]	$97.8 \pm 1.0\%$	$96.0 \pm 0.7\%$	$85.4 \pm 2.5\%$
TarMAC-IC3Net [6]	$84.8 \pm 4.5\%$	$95.5 \pm 1.3\%$	$88.1 \pm 1.9\%$
GA-Comm [73]	$95.9 \pm 0.1\%$	$97.1 \pm 0.7\%$	$95.8 \pm 1.1\%$

other intensively before finding the prey. After finding the prey, communication becomes unnecessary, and the learned communication graphs should be sparse, as we see in Figure 3.8(b). This inspection supports that MAGIC learns to communicate only when beneficial to team performance.

3.6.2 Traffic Junction

We evaluate our method in the Traffic Junction domain for the cases of a maximum of 5 agents, 10 agents and 20 agents at a junction. Table 3.2 shows the success rate (i.e., no collision in an episode) for each method at convergence in Traffic Junction. Our algorithm achieves near-perfect performance after convergence, widely outperforming all

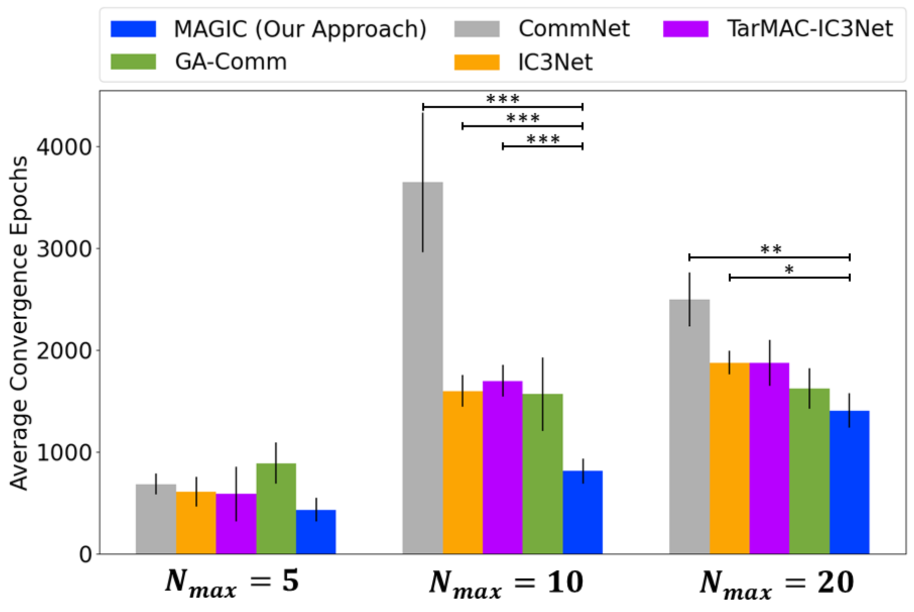


Figure 3.9: The average number of epochs for convergence in Traffic Junction with standard error bars.

benchmarks in its success rate. Figure 3.9 depicts the average number of epochs taken to converges for each method. Our method maintains quick convergence as the number of agents increase. However, several of the benchmarks experience a slowdown in their convergence rate.

Impact of the Message Processor - In Traffic Junction, we allow all agents to communicate to accelerate training for all methods, common in highly vision-limited environments [22]. As we are using complete graphs (i.e., Scheduler is not used), our SOTA performance in the Traffic Junction domain displays that the MP considerably contributes to the success of our algorithm.

3.6.3 Google Research Football

Lastly, we evaluate our algorithm and several state-of-the-art communication-based MARL baselines in the GRF environment. The results presented provide some insight into each algorithm's ability to handle stochasticity, sparse rewards, and a high-complexity state-action space. We utilize the scoring success rate as our metric of evaluation.

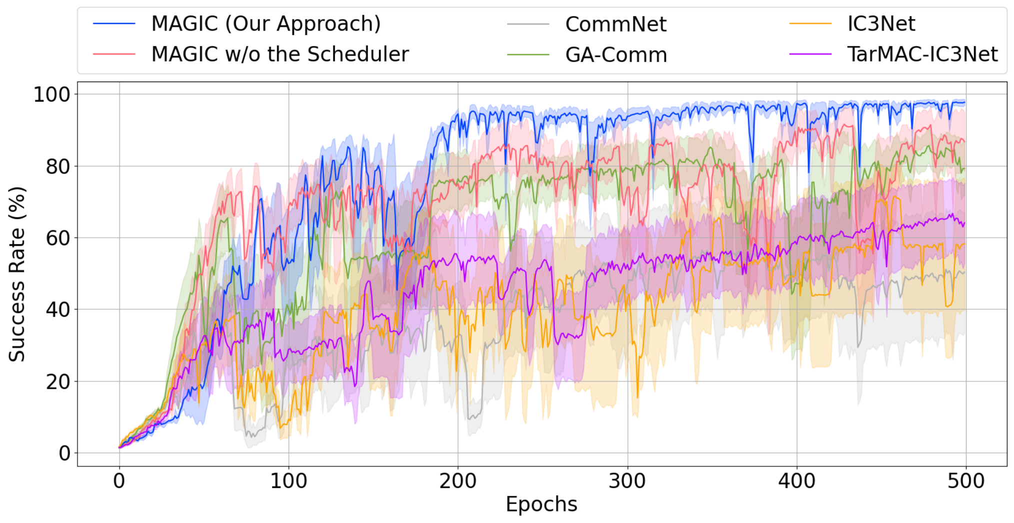


Figure 3.10: The success rate in GRF as training proceeds.

Table 3.3: The success rate and average steps taken to finish an episode in GRF.

Method	Success Rate	Steps Taken
MAGIC (Our Approach)	$\mathbf{98.2 \pm 1.0\%}$	$\mathbf{34.30 \pm 1.34}$
Ours (without the Scheduler)	$91.0 \pm 4.6\%$	36.31 ± 2.59
CommNet [23]	$59.2 \pm 13.7\%$	39.32 ± 2.35
IC3Net [22]	$70.0 \pm 9.8\%$	40.37 ± 1.22
TarMAC-IC3Net [6]	$73.5 \pm 8.3\%$	41.53 ± 2.80
GA-Comm [73]	$88.8 \pm 3.9\%$	39.05 ± 3.05

Impact of the Scheduler - We verify the impact of the Scheduler mechanism in MAGIC by testing our method without the use of the Scheduler. As seen, the addition of a Scheduler provides a performance improvement. This result signifies the importance of determining "when" and "whom" to communicate with.

Figure 3.10 displays the success rate for offending agents to score as the training proceeds. Our method converges to a higher-performing policy than all the baselines. In our settings, although each agent only has local observations, it is able to completely observe the state space. MAGIC without the ability to utilize a scheduler performs poorly. It is interesting to note that even with unlimited vision, utilizing a complete graph for communication performs much worse than utilizing a scheduler that gives precise and targeted communication. Table 3.3 displays the success rate and average steps taken to finish an episode at convergence. Our method has a success rate of 98.5%, which is approximately a 10.5% improvement over the next-best baseline of GA-Comm. Our method achieves the lowest number of average steps, scoring 13.8% more quickly than the closest benchmark of GA-Comm. As we have have outperformed all benchmarks within the previous two domains and in the complex GRF environment, we conclude that MAGIC is high-performing, scales well to the number of agents, robust to stochasticity, and performs well with sparse rewards.

3.6.4 Communication Efficiency

We present an analysis of the communication efficiency of our method in Table 3.4. Communicating efficiently can save resources and allow for messages to be processed more easily. To gauge the efficiency, we utilize the performance improvement due to communication and divide the communication graph density. The graph densities are determined using the sparsity of the adjacency matrix. A fully-connected graph corresponds to a density of 1, and a graph with no connections corresponds to a density of 0. We perform this analysis within a Predator-Prey domain of grid size $5x5$ with 3 predators, where a performance improvement refers to a reduction in steps. To obtain the performance improvement due to communication, we evaluate a communication-blocked variant of each method and compare the performance to the method itself. All methods use the same message size and one-round of communication to maintain a similar network complexity. Utilizing the performance improvement due to communication divided by the graph density as our metric for communication efficiency, we see that MAGIC is the most efficient. MAGIC communicates 27.4% more efficiently on average than baselines while also achieving the highest performance.

Impact of the combined framework of MAGIC - As we compare each method to its non-communicatory variant, in MAGIC, this results in removing the Scheduler and Message Processor. MAGIC achieves the greatest improvement compared to baselines, displaying

Table 3.4: Communication efficiency measured as the performance improvement with communication divided by graph density.

Method	Graph Density	Avg. Steps w/ Comms	Performance Improvement	$\frac{\text{Improvement}}{\text{Density}}$
MAGIC (Our Approach)	**0.644**	**8.504**	**7.562**	**11.743**
CommNet [23]	0.667	9.216	6.455	9.681
IC3Net [22]	0.638	9.208	6.421	10.058
TarMAC-IC3Net[6]	0.856	9.376	5.958	6.956
GA-Comm [73]	0.514	9.334	5.868	11.391

the contribution from the combined effects of the Scheduler and Message Processor.

3.6.5 Discussion

Across multiple test domains, we set a new state-of-the-art in communication-based MARL performance, outperforming baselines, including [23, 22, 73, 6]. Across our domains, we achieve an average 5.8% improvement in steps taken (Predator-Prey), average 1.9% improvement in success rate (Traffic Junction), 10.5% improvement in success rate (GRF), and 13.8% improvement in steps taken compared to the closest benchmark across each domain. The strong performance improvement we achieve in GRF suggests our approach is better able to scale to high-dimension state-action spaces while effectively handling stochasticity and sparse rewards. While our architecture shares some attributes as GA-Comm [73], GA-Comm has several drawbacks including large epoch training times due to the complex hard attention structure, a dot-product soft attention mechanism *without scaling*, and the inability to extend to multi-round communication. Our approach takes significantly less compute by avoiding any recurrent structures in the Scheduler. Specifically, GA-Comm requires 2x long to train with 3 agents, 3x long with 5 agents, and 4x as long with 10 agents. Prior work [99] has also shown that additive attention (used in the GAT layers in our Message Processor) outperforms dot-product attention without scaling when the message size is large. MAGIC's Scheduler can explicitly learn and generate different graphs for different rounds of communication, allowing for higher performance. Additionally, MAGIC achieves the highest communication efficiency, outperforming benchmarks by 27.4% on average.

3.7 Physical Robot Demonstration

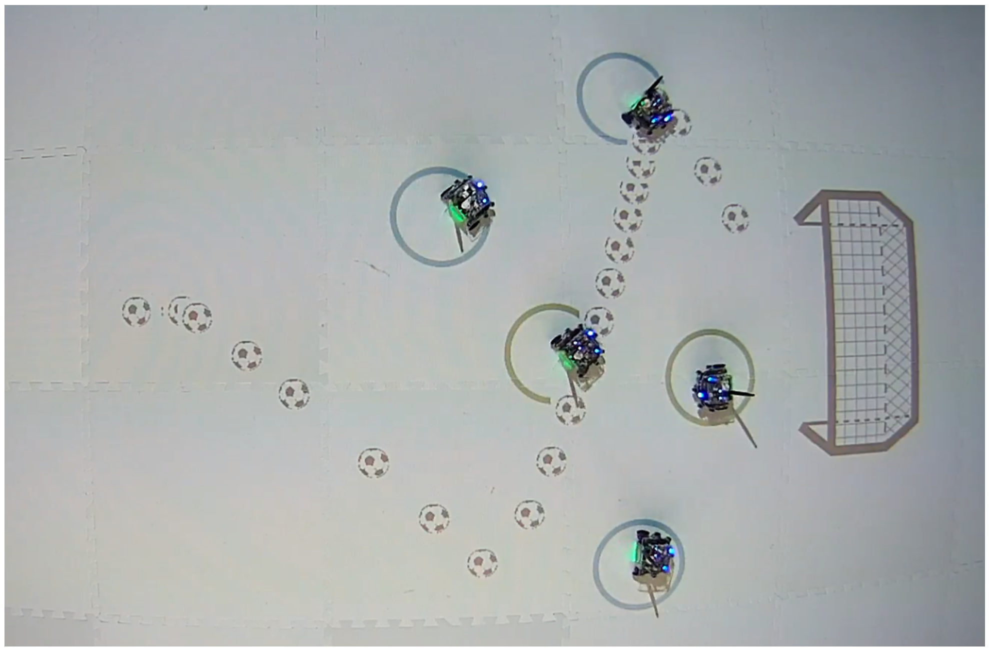


Figure 3.11: This figure displays a demonstration of our algorithm on physical robots on the Robotarium platform. The display shows a 3 vs. 2 soccer scenario, with blue agents as the attackers, and red agents as defenders.

We present a demonstration of our algorithm in a similar 3-vs.-2 soccer scenario on physical robots in the Robotarium, a remotely accessible swarm robotics research platform [100]. This demonstration displays the feasibility of trajectories produced by our MARL algorithm. We present a depiction of MAGIC's deployed trajectory in Figure 3.11. A video is attached in this link.

3.8 Conclusion

In this chapter, we propose a novel, end-to-end-trainable, graph-attention communication protocol, MAGIC, that utilizes a Scheduler to solve the problems of when to communicate and whom to address messages to, and a Message Processor to integrate and process messages. We evaluate our method and baselines in several environments, achieving state-of-the-art performance. In GRF, we achieve a 98.5%, near-perfect success rate, while most baselines struggle to reach 70%. Not only does MAGIC produce SOTA results, MAGIC is able to converge 52% faster than the next-quickest baseline, and communicates 27.4% more efficiently than the average baseline.

CHAPTER 4
LEARNING SCALABLE AND ADAPTABLE MULTI-AGENT COMMUNICATION

4.1 Introduction

In chapter 3, we present a communication protocol for MARL that is designed for training agents to master one task at a time. It can take a long time to train each task from scratch, even if the new task is not changed a lot from the old one. In the real world, people can learn new skills extremely quickly because they never learn from scratch. In multi-agent systems, training on multiple tasks and adapting to new tasks can be challenging, because the agent might need to adjust their inter-agent coordination or communication strategies and the change in the number of agents can make this even more difficult. In this chapter, we first present a multi-agent multi-task reinforcement learning framework, MT-MAGIC, and then present the method of multi-agent meta-reinforcement learning with RNN-based black-box adaptation. These methods are both built upon MAGIC introduced in chapter 3, and they are able to handle tasks with different agent team sizes. Through experiments, we find that MT-MAGIC can learn efficiently in most tested tasks and generalize well on new tasks, and Meta-MAGIC can achieve improved multi-task performance and better unseen-task adaptation compared to MT-MAGIC and the baseline.

We begin this chapter with section 4.2 and introduce the preliminaries for multi-task reinforcement learning and RNN-based meta-reinforcement learning. Then, we present our method in section 4.3. In section 4.4, experiment designs and settings are described. section 4.5 shows the results and discussion of the experiments. Lastly, section 4.6 concludes this chapter.

4.2 Preliminaries

In this chapter, we follow the framework of POMG augmented with communication defined in the previous chapter. In this section, multi-agent multi-task reinforcement learning and black-box adaptation using RNN for single-agent RL will be briefly reviewed.

4.2.1 Multi-Task Reinforcement Learning

The goal of multi-task RL is that an agent needs to optimize its policy θ to solve a sequence of tasks which can be represented as MDPs. A general learning objective of the single-agent multi-task RL for M tasks is displayed in Equation 4.1.

$$J_{\text{MT}}(\theta) = \sum_{m=1}^{M} \mathbb{E}_{s\sim p_m^\pi, a\sim\pi_\theta}\big[\sum_{t=1}^{T}\gamma^{t-1} r_m(s_t, a_t)\big] \tag{4.1}$$

Here, p_m^π is the state distribution regarding policy π_θ and task m, γ is the discount factor, and $r_m(\cdot)$ is the reward function for task m.

4.2.2 Meta-Reinforcement Learning: Black-Box Adaptation with Recurrent Neural Networks

The aim of meta-RL is learning to reinforcement learn. Specifically, an agent should learn a new task more quickly or more proficiently, based on its experience on previous tasks. The general learning objective of meta-RL can be expressed in Equation 4.2 and Equation 4.3.

$$J_{\text{Meta}}(\theta) = \sum_{m=1}^{M} \mathbb{E}_{s\sim p_m^\pi, a\sim\pi_{\phi_m}}\big[\sum_{t=1}^{T}\gamma^{t-1} r_m(s_t, a_t)\big] \tag{4.2}$$

$$\phi_m = f_\theta(\mathcal{M}_m) \tag{4.3}$$

Here, $\mathcal{M}_m$ represents the MDP for task m, $f_\theta(\cdot)$ is an adaption function parameterized by θ that can be applied to different MDPs, and ϕ_m is the policy parameter after adaptation to the MDP for task m. Specifically, depending on the meta-RL method, ϕ_m can represent the hidden state of a RNN within a black-box adaption method, or the entire parameters of the policy network within a optimization-based method like MAML.

In [78, 48], RNN structures are concurrently proposed to be applied in meta-RL for adaptation to new tasks. The dynamics of the recurrent network can represent a procedure of adaptation using the history information from the interaction with a new task. Implementing reinforcement learning and optimizing the network over a trajectory of multiple episodes internally improves the agent's policy on a task while negotiating the exploration-exploitation trade-off. Here, we define the sequential episodes of a task as a meta-episode, where the first few episodes can be used to explore the environment and provide knowledge for the later episodes. Given enough training on multiple tasks, learning and adaptation can occur for each task with the weights of the recurrent network held constant. Specifically, during training, at the start of each meta-episode, a task from a certain distribution is sampled, and the hidden state of the RNN policy is reset. At each time step within a meta-episode, the current state s^t, previous action a^{t-1}, previous reward r^{t-1}, and the flag of time or environment episode termination for the previous time step d^{t-1}, are input to the RNN. Between two episodes within a meta-episode, the RNN hidden state is not reset and preserved for the next episode to provide insights from the previous exploration.

The learning objective maximizes the expected discounted total reward over a meta-episode instead of an environment episode. The meta-RL problem can be cast as a reinforcement learning problem with the objective shown in Equation 4.4.

$$J_{\text{Meta}}(\theta) = \sum_{m=1}^{M} \mathbb{E}_{s\sim p_m^{\pi}, a\sim\pi_\theta}\big[\sum_{t=1}^{T'} \gamma^{t-1} r_m({s'}^t, a_t)\big] \tag{4.4}$$

$${s'}^t = \text{Concat}(s^t, a^{t-1}, r^{t-1}, d^{t-1}) \tag{4.5}$$

Here, T' is the total time steps of a meta-episode, ${s'}^t$ is the meta-state concatenating s^t, a^{t-1}, r^{t-1}, and d^{t-1}, and θ represents the weights of RNN (and other networks if there are any). To this end, θ is the parameters of the RL policy, and also the parameters of the adaptation function of Meta-RL, so it incorporates combined concepts of θ and ϕ_m in Equation 4.2.

If the agent can only get access to the environment with partially observable settings, i.e., the agent interacts with a sequence of POMDPs, the method can be applied without any changes except replacing the input states to the network with observations.

4.3 Method

4.3.1 Multi-Task MAGIC

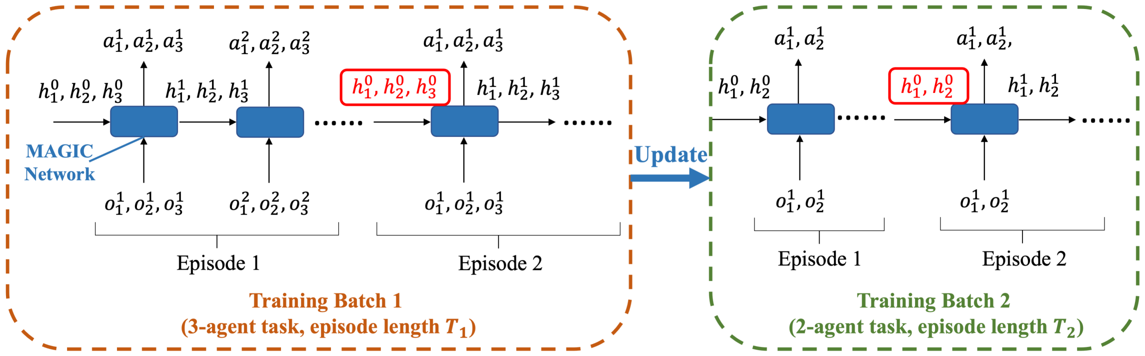


Figure 4.1: This figure displays an example of the training scheme of Multi-Task MAGIC.

We follow a similar training scheme to that described in subsection 3.3.4, with a sequence of different tasks fed in for agents to interact with. We can revise Equation 3.10 to the multi-task form displayed in Equation 4.6.

$$\nabla_{\theta,\phi}\mathcal{L}_{\text{MT}}(\theta,\phi) = \sum_{m=1}^{M}\frac{1}{t_{max}}\sum_{i=1}^{N_m}\sum_{t=1}^{t_{max}}[-\nabla_\theta \log \pi_\theta(a_i^t|o_i^t)(R_{m,i}^t - V_\phi(o_i^t)) + \beta\nabla_\phi(R_{m,i}^t - V_\phi(o_i^t))^2] \tag{4.6}$$

Here, N_m is the number of agents in task m, $R_{m,i}^t$ is the discounted total reward for agent i in task m, starting from time step t in an episode. Note that for each update cycle we collect data and update the model from a mini-batch of one task. We provide an example of training two tasks with 3 agents and 2 agents respectively, in Figure 4.1. The 3-agent task has a episode length T_1 and the 2-agent task's episode length is T_2. At the start of training batch one, the hidden states of the three agents $\{h_1^0, h_2^0, h_3^0\}$ will be initialized and fed into the MAGIC protocol network. At each time step t, the network takes in observations $\{o_1^t, o_2^t, o_3^t\}$, outputs $\{a_1^t, a_2^t, a_3^t\}$, and passes the new hidden states $\{h_1^{t+1}, h_2^{t+1}, h_3^{t+1}\}$ to the next time step. Here, we ignore the LSTM's cell states for brevity. When a new episode starts, the hidden states need to be reset as $\{h_1^0, h_2^0, h_3^0\}$. When episode two is terminated, the model will be updated using the trajectories collected by training batch one for the 3-agent task. Proceeding to training batch two, the hidden states of the two agents $\{h_1^0, h_2^0\}$ will be initialized and fed into the network. At each time step t, the network takes in observations $\{o_1^t, o_2^t\}$, outputs $\{a_1^t, a_2^t\}$, and passes the new hidden states $\{h_1^{t+1}, h_2^{t+1}\}$ to the next time step. As our work follows the framework of POMG augmented with communication, agents act in a decentralized manner, and share the network structures for encoding observations and outputting actions. Accordingly, the observations and actions of different agents are processed in a batch-wise manner, so we do not need to revise the structure of MAGIC network to adapt to agent number changes as tasks are switched.

4.3.2 Meta-MAGIC

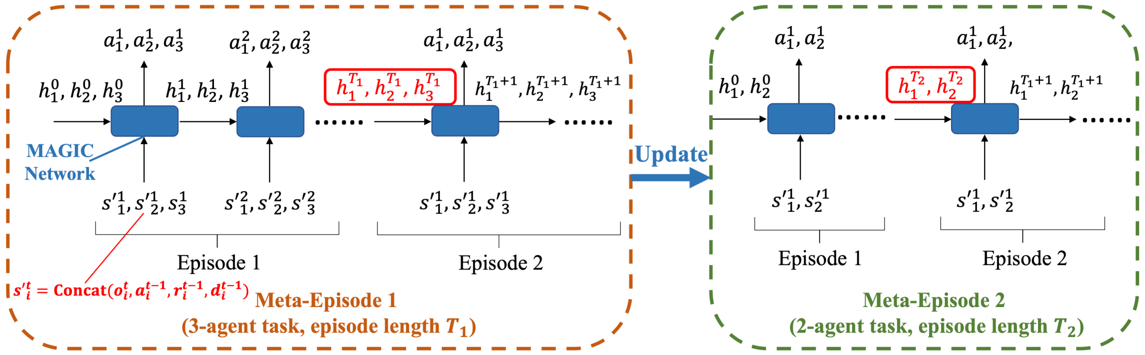


Figure 4.2: This figure displays an example of the training scheme of Meta-MAGIC.

In this subsection, we present Meta-MAGIC, which is a MAGIC-based multi-agent meta-reinforcement learning method that can generalize on tasks with different team sizes. The LSTM structure in MAGIC provides a convenient basis for designing a RNN-based meta-RL method. We employ MAGIC's inherent LSTM structure to perform adaptation to tasks as described in subsection 4.2.2. Therefore, we can cast the multi-agent meta-RL problem as a common multi-agent multi-task reinforcement learning problem, with the update rule displayed in Equation 4.7.

$$\nabla_{\theta,\phi}\mathcal{L}_{\text{Meta}}(\theta,\phi) = \sum_{m=1}^{M}\frac{1}{t'_{max}}\sum_{i=1}^{N_m}\sum_{t=1}^{t'_{max}}[-\nabla_\theta \log \pi_\theta(a_i^t|{s'}_i^t)(R_{m,i}^t - V_\phi({s'}_i^t)) + \beta\nabla_\phi(R_{m,i}^t - V_\phi({s'}_i^t))^2] \tag{4.7}$$

$${s'}_i^t = \text{Concat}(o_i^t, a_i^{t-1}, r_i^{t-1}, d_i^{t-1}) \tag{4.8}$$

Here, t'_{max} is the number of steps taken within a meta-episode which is also a training batch. The hidden states of RNN is not reset within the same meta-episode. ${s'}_i^t$ represents the meta-state of agent i at time step t, concatenating the agent specific observation o_i^t, action a_i^{t-1}, reward r_i^{t-1}, and optional extra information d_i^{t-1} (e.g., a flag of time or environment episode termination). θ represents the communication-based LSTM policy, and ϕ represents the corresponding value function parameters.

The training scheme is displayed in Figure 4.2 with an example of a 3-agent (episode length T_1) and 2-agent (episode length T_2) task. Similar to Multi-Task MAGIC, at the start of training batch one, the hidden states of the three agents $\{h_1^0, h_2^0, h_3^0\}$ will be initialized and fed into the MAGIC protocol network. At each time step t, the network takes in the meta-states $\{{s'}_1^t, {s'}_2^t, {s'}_3^t\}$, outputs $\{a_1^t, a_2^t, a_3^t\}$, and passes the new hidden states $\{h_1^{t+1}, h_2^{t+1}, h_3^{t+1}\}$ to the next time step. When a new episode starts, the hidden states are not reset but instead preserved as $\{h_1^{T_1}, h_2^{T_1}, h_3^{T_1}\}$. In this way, the exploration information of each agent from episode one can be delivered to episode two for better decision-makings to the 3-agent task. When episode two is terminated, the model will be updated using the trajectories collected by meta-episode one for the 3-agent task. Here, the update does not only optimize for a better LSTM policy, but also optimize for a better adaption to the task.

What is more, optimizing over the entire meta-episode with a RNN policy automatically help agents learn to explore. Proceeding to training batch two, the hidden states from previous meta-episode are not preserved anymore, and the hidden states of the two agents $\{h_1^0, h_2^0\}$ are initialized and fed into the network. At each time step t, the network takes in $\{{s'}_1^t, {s'}_2^t\}$, outputs $\{a_1^t, a_2^t\}$, and passes the new hidden states $\{h_1^{T_2}, h_2^{T_2}\}$ to the next time step.

4.4 Experiment Settings

We test Multi-Task MAGIC and Meta-MAGIC on GRF and Predator-Prey environments. In this section, we introduce the chosen baseline, objectives and designs, tested scenarios and state representations, and the training details of our experiments.

4.4.1 Baseline

We choose TarMAC-IC3Net as our baseline for experiments of multi-agent multi-task RL and meta-RL. GA-Comm is not employed because it does not support training on tasks with different team sizes. TarMAC-IC3Net has comparable performance to GA-Comm while requires much less computational time. We follow the steps in subsection 4.3.1 to enable TarMAC-IC3Net to work with multi-task settings, which we refer to as MT-TarMAC-IC3Net. For brevity, we also refer to Multi-Task MAGIC as MT-MAGIC.

4.4.2 Objectives and Designs

Through the experiments, we would like to answer the following two questions:

1. Can MT-MAGIC learn efficiently in multi-task settings and generalize well on new tasks?

2. Can Meta-MAGIC provide improved multi-task performance and better unseen-task adaptation compared to MT-MAGIC and MT-TarMAC-IC3Net?

Accordingly, we design experiments and evaluate methods in three aspects:

1. Multi-Task Training: First, we train and evaluate the three methods in multi-task settings. Multi-task training is also the upstream task for meta-learning and transfer learning (including zero-shot and fine-tuning), so models will be saved from multi-task training and used for later experiments.

2. Zero-Shot/Few-Shot Testing on new tasks: Here, we test each method's performance on new tasks without any model updates using the original models saved from multi-task training. The goal is to test the zero-shot transferability of MT-MAGIC and MT-TarMAC-IC3Net, and meta-test few-shot adaptation performance on new tasks for Meta-MAGIC. Here, k-shot for Meta-MAGIC means that the agents will continuously interact with the same new environment for k episodes. We set k as 10 in our testing.

3. Fine-Tuning on New Tasks: Here, we will fine-tune the models of MT-MAGIC and compare the performance of models trained from scratch on new tasks. The goal is to evaluate the transferability of MT-MAGIC.

4.4.3 Tested Scenarios and State Representations

In GRF, for multi-task training, our tasks include 2 vs. 2, 3 vs. 2, and 3 vs. 3. For zero-shot/few-shot testing or fine-tuning, the tasks include 3 vs. 2, 3 vs. 3* (multi-task training on 2 vs. 2 and 3 vs. 3), and 2 vs.3, 4 vs. 3 (multi-task training on 2 vs. 2, 3 vs. 2, and 3 vs. 3). Tasks are different from each other in agent numbers and initialized positions. We keep other settings the same as the one described in section 3.4. As task changes during multi-task training, the dimensions of single agent's observation and action are required to be constant. We employ zero padding to unify the observation spaces of different tasks in GRF due to the difference of player numbers. Following section 3.6, we use the average success rate in an epoch to evaluate methods in GRF.

In Predator-Prey, for multi-task training, our tasks include 4-agents, 5-agents, 6-agent cases in 10×10 grid, and 8-agent, 10-agent, 12-agent cases in 20×20 grid. For zero-shot/few-shot testing or fine-tuning, the tasks include 3-agent, 8-agent cases in 10×10 grid (multi-task training on 4-agent, 5-agent, 6-agent 10×10 grids), 7-agent, 9-agent, 11-agent cases in 20×20 grid (multi-task training on 8-agents, 10-agent 20×20 grids, or 8-agents, 10-agent, 12-agent 20×20 grids). The settings for all 10×10-grid tasks follow the 5-agent 10×10-grid task in section 3.4. The settings for all 20×20-grid tasks follow the 10-agent 20×20-grid task in section 3.4. Because the observation is grid-based in Predator-Prey, changing the number of agents while keeping the grid size fixed will not affect the dimension of the observation space. Following section 3.6, we use the average steps taken to finish an episode in an epoch to evaluate methods in Predator-Prey.

4.4.4 Training Details

During the multi-task training phase, the size of training batch/meta-episode is 1000 to stabilize training, compared to the batch size of 500 employed in single-task cases described in section 3.5. Each epoch includes $5 * M$ training batch/meta-episode, where M represents the total number of tasks. In our training, the numbers of tasks are not large, so instead of randomly sampling tasks from the task set, the agents are set to interact with each task in turn to enable each task to be experienced uniformly. Therefore, each task is trained on for five times in each epoch. On top of the multi-task settings, we follow the same training parameters and settings in section 3.5.

4.5 Results and Discussion

In this section, we present and discuss the results of the experiments in three subsections: Multi-Task Training, Zero-Shot/Few-Shot Adaptation to Unseen Tasks, and Fine-Tuning. The results in all GRF tasks are averaged over 4 seeds, and the results in all Predator-Prey tasks are averaged over 3 seeds.

4.5.1 Multi-Task Training

Here, we present the results of multi-task training in GRF (Table 4.1) and Predator-Prey (Table 4.2).

Table 4.1: In this table, we display the results of multi-task training in GRF. We use the average success rate in an epoch to evaluate methods in GRF. The performance (mean±standard error) of each task and the average performance are provided.

(a) Task Set: 2 vs. 2, 3 vs. 3

Method	2 vs. 2	3 vs. 3	Avg.
MT-TarMAC-IC3Net	$49.01 \pm 0.92\%$	$77.40 \pm 10.30\%$	$63.21 \pm 5.04\%$
MT-MAGIC	$42.94 \pm 2.54\%$	$88.00 \pm 6.67\%$	$65.47 \pm 4.26\%$
Meta-MAGIC	**49.67 ± 10.30%**	**92.13 ± 4.73%**	**70.90 ± 3.56%**

(b) Task Set: 2 vs. 2, 3 vs. 2, 3 vs. 3

Method	2 vs. 2	3 vs. 2	3 vs. 3	Avg.
MT-TarMAC-IC3Net	$55.48 \pm 6.91\%$	$91.45 \pm 3.24\%$	$53.76 \pm 11.40\%$	$66.90 \pm 5.04\%$
MT-MAGIC	**69.78 ± 7.40%**	$89.66 \pm 4.14\%$	$68.71 \pm 14.00\%$	$76.05 \pm 5.17\%$
Meta-MAGIC	$63.87 \pm 6.11\%$	**92.50 ± 1.77%**	**78.57 ± 2.69%**	**78.31 ± 1.47%**

Google Research Football

In GRF, we run separate multi-task training on two sets of tasks, and present the performance of each task and the average performance for each set. Meta-MAGIC leads in success rate in most cases, and MT-MAGIC can outperform MT-TarMAC-IC3Net in most tasks. Both Meta-MAGIC and MT-MAGIC achieve over 60% of success rate in all scenarios except 2 vs. 2. We also find that training on more tasks might help improve the average performance and sometimes help the models escape from sub-optimum.

Predator-Prey

In Predator-Prey, we run experiments on three sets of tasks, with one on the 10×10 grid, and two on the 20×20 grid. From the results, we can see that MT-TarMAC-IC3Net has already successfully learned good policies for agents to find the prey quickly in multi-task

Table 4.2: In this table, we display the results of multi-task training in Predator-Prey. We use the average steps taken to finish an episode in an epoch to evaluate methods in Predator-Prey. The performance (mean±standard error) of each task and the average performance are provided.

(a) Task Set: 10 × 10: 4, 5, 6 agents

Method	4 agents	5 agents	6 agents	Avg.
MT-TarMAC-IC3Net	13.80 ± 0.24	14.04 ± 0.18	13.25 ± 0.15	13.70 ± 0.19
MT-MAGIC	12.67 ± 0.12	12.94 ± 0.01	$\mathbf{12.30 \pm 0.04}$	12.64 ± 0.04
Meta-MAGIC	$\mathbf{12.55 \pm 0.03}$	$\mathbf{12.93 \pm 0.04}$	12.40 ± 0.09	$\mathbf{12.63 \pm 0.06}$

(b) Task Set: 20 × 20: 8, 10 agents

Method	8 agents	10 agents	Avg.
MT-TarMAC-IC3Net	33.39 ± 0.44	34.89 ± 0.17	34.14 ± 0.30
MT-MAGIC	32.91 ± 0.86	34.30 ± 0.73	33.69 ± 0.86
Meta-MAGIC	$\mathbf{31.77 \pm 0.21}$	$\mathbf{33.63 \pm 0.02}$	$\mathbf{32.70 \pm 0.10}$

(c) Task Set: 20 × 20: 8, 10, 12 agents

Method	8 agents	10 agents	12 agents	Avg.
MT-TarMAC-IC3Net	32.49 ± 0.08	33.42 ± 0.36	30.06 ± 0.58	31.99 ± 0.25
MT-MAGIC	31.99 ± 0.56	33.16 ± 0.36	30.38 ± 0.16	31.84 ± 0.36
Meta-MAGIC	$\mathbf{31.34 \pm 0.24}$	$\mathbf{32.42 \pm 0.28}$	$\mathbf{29.26 \pm 0.40}$	$\mathbf{31.01 \pm 0.29}$

settings. Meta-MAGIC and MT-MAGIC can still beat MT-TarMAC-IC3Net by a considerable margin, and Meta-MAGIC can reach the best performance in most tasks. Comparing the two sets of tasks in 20×20 grid, training on the extra task with 12 agents helps improve the average performance of all tasks and individual performance of the tasks with 8 and 10 agents. This indicates that the designed multi-task training method for multiple agents efficiently exploits the shared structure of different tasks, and the trained agents can communicate and make decisions effectively even when the team size changes.

4.5.2 Zero-Shot/Few-Shot Adaptation to Unseen Tasks

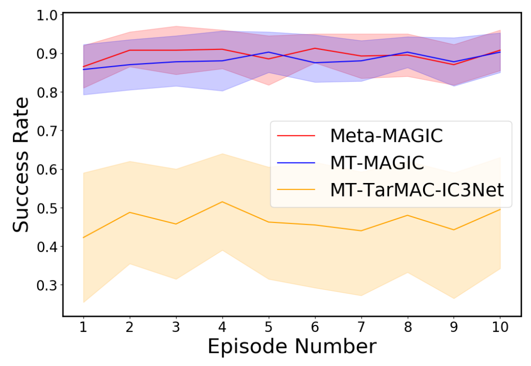
Here, we present the results of zero-shot (MT-TarMAC-IC3Net, MT-MAGIC) or few-shot (Meta-MAGIC) adaptation to unseen tasks. As mentioned in subsection 4.4.2, no method will update the model at this stage. Each method is tested on interactions with the new environment of 100 batches, with 10 sequential episodes in each batch. Meta-MAGIC's hidden states are allowed to be passed between two sequential episodes. The performance of each method is averaged over these 1000 episodes and presented in Table 4.3 and Table 4.4. Each episode's performance in a batch is also averaged and displayed in Figure 4.3 and Figure 4.4. In all tables and figures, we annotate the unseen task and the corresponding source tasks from multi-task training.

Table 4.3: In this table, we display the results of zero-shot/few-shot testing in GRF. The adaptation performance (mean±standard error) on each unseen task is provided.

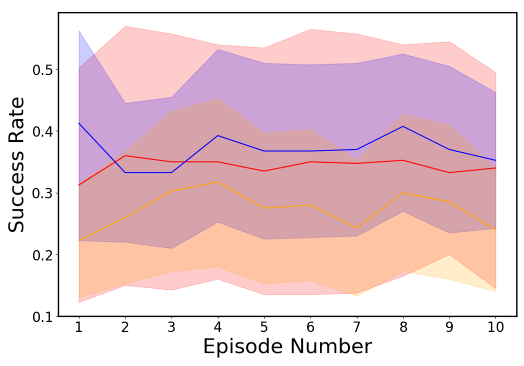
Method	2 vs. 2, 3 vs. 3		2 vs. 2, 3 vs. 2, 3 vs. 3	
	3 vs. 2	3 vs. 3*	2 vs. 3	4 vs. 3
MT-TarMAC-IC3Net	$70.18 \pm 14.0\%$	$27.25 \pm 11.4\%$	$14.58 \pm 2.92\%$	$18.85 \pm 9.02\%$
MT-MAGIC	$88.10 \pm 5.07\%$	$\mathbf{37.05 \pm 13.1}\%$	$\mathbf{34.33 \pm 13.7}\%$	$17.50 \pm 7.94\%$
Meta-MAGIC	$\mathbf{89.53 \pm 4.85}\%$	$34.43 \pm 16.9\%$	$25.05 \pm 6.38\%$	$\mathbf{23.65 \pm 2.87\%}$

Google Research Football

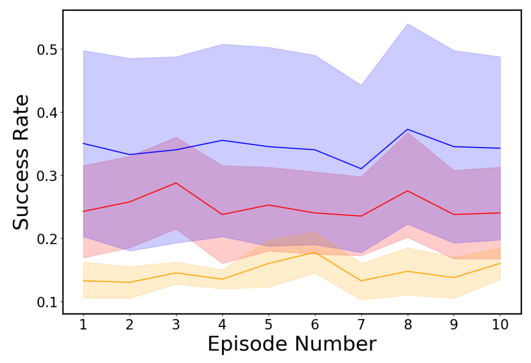
From the task set of 2 vs. 2 and 3 vs. 3, we test/meta-test each method on 3 vs.2 and 3 vs. 3*, and from the task set of 2 vs. 2, 3 vs.2, and 3 vs. 3, we test/meta-test each method on 2 vs.3 and 4 vs. 3. Meta-MAGIC leads in 3 vs. 2 and achieves a success rate close to 90%. However, in all other tasks, direct adaptation without model updates achieve no more than 40% success rate, and MT-MAGIC beats Meta-MAGIC in 3 vs. 3* and 2 vs. 3. We run experiments on task sets with relatively small sizes, and unlike Predator-Prey, GRF is more challenging and the agents' initialized position for each task is fixed, so even a slightly-revised new task can become out of distribution from the small task set and lead to failure in zero-shot/few-shot adaptation. On the other hand, from Figure 4.3, we find that Meta-MAGIC constantly has a performance improvement from episode one to episode two, while other methods do not. This can indicate that Meta-MAGIC performs fast adaptation within the first two episodes in the new environment. Furthermore, from the



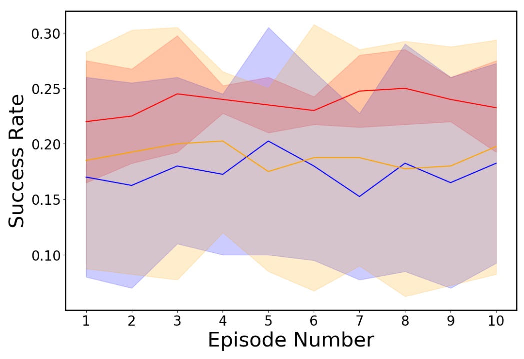
(a) 3 vs. 2 from 2 vs. 2, 3 vs. 3



(b) 3 vs. 3* from 2 vs. 2, 3 vs. 3



(c) 2 vs. 3 from 2 vs. 2, 3 vs. 2, 3 vs. 3



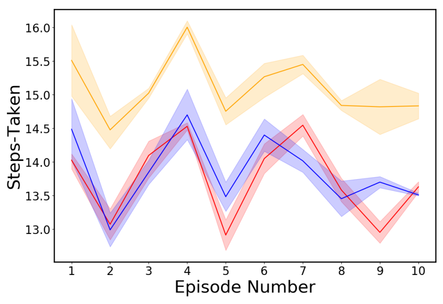
(d) 4 vs. 3 from 2 vs. 2, 3 vs. 2, 3 vs. 3

Figure 4.3: This figure displays the zero-shot/few-shot adaptation performance of ten episodes for unseen tasks within GRF. The performance is averaged over 100 batches. The shaded regions represent standard errors. Starting higher means the model adapts better within a few steps in the first episode based on the multi-task training. A positive slope means the model keeps improving (i.e., adapting) as the agents interact within the simulator. We find MT-MAGIC and Meta-MAGIC can start higher than MT-TarMAC-IC3Net, and Meta-MAGIC constantly improves, intelligently adapting within two or three episodes, while other methods do not. Unrolling for longer does not improve performance. This aligns with the results of another black-box adaptation method for single-agent meta-RL [49].

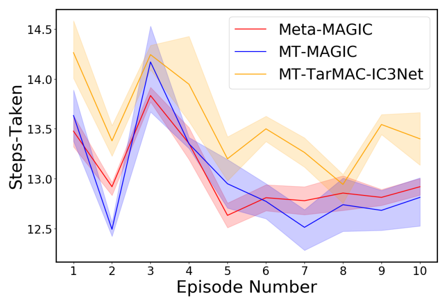
following results in Predator-Prey, we find a similar phenomenon regarding Meta-MAGIC, which indicates the significance of the adaptation process of our method.

Predator-Prey

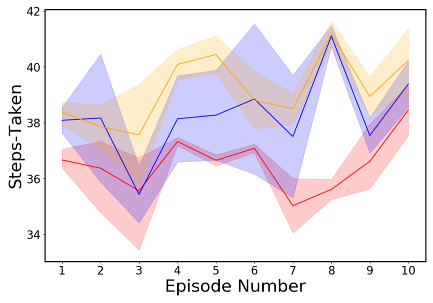
Table 4.4 shows that Meta-MAGIC outperforms other methods when adapting to most new scenarios in Predator-Prey, and MT-MAGIC has better zero-shot performance than MT-TarMAC-IC3Net. From Figure 4.4, we can see that the steps taken achieved by Meta-MAGIC decrease from episode one to episode two in almost all cases, indicating that Meta-



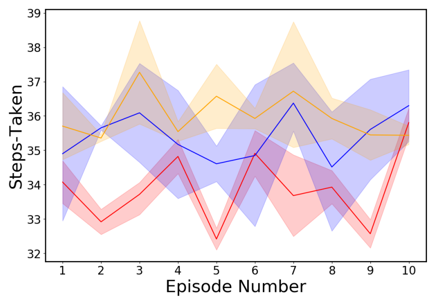
(a) 10 × 10: 3 agents from 4, 5, 6 agents



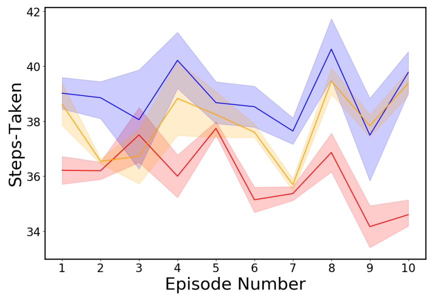
(b) 10 × 10: 8 agents from 4, 5, 6 agents



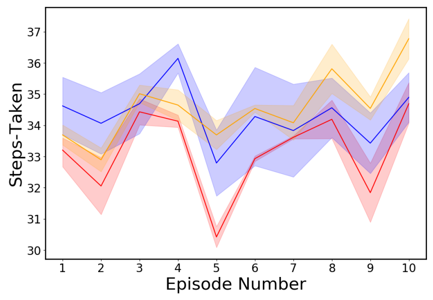
(c) 20 × 20: 7 agents from 8, 10 agents



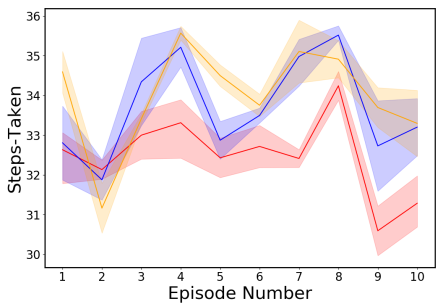
(d) 20 × 20: 9 agents from 8, 10 agents



(e) 20×20: 7 agents from 8, 10, 12 agents



(f) 20×20: 9 agents from 8, 10, 12 agents



(g) 20×20: 11 agents from 8, 10, 12 agents

Figure 4.4: This figure displays the zero-shot/few-shot adaptation performance of ten episodes in Predator-Prey unseen tasks. The performance is averaged over 100 batches. The shaded regions represent standard errors. A lower value for steps taken on the vertical axis is better. Starting lower means the model adapts better within a few steps in the first episode based on the multi-task training. A negative slope means the model keeps improving adaptation as the agents interact with more episodes. We see Meta-MAGIC can adapt to a task within two episodes, and is more likely to keep improving in its adaptation afterwards compared to other methods. We note that a meta-RL method based on black-box adaptation usually does not improve the performance monotonically through interactions [49, 48].

Table 4.4: In this table, we display the results of zero-shot/few-shot testing in Predator-Prey. The adaptation performance (mean±standard error) on each unseen task is provided.

(a) Source Task Set: 10×10: 4, 5, 6 agents

Method	3 agents	8 agents
MT-TarMAC-IC3Net	15.10 ± 0.18	13.57 ± 0.14
MT-MAGIC	13.86 ± 0.15	**13.01 ± 0.15**
Meta-MAGIC	**13.74 ± 0.04**	13.04 ± 0.07

(b) Source Task Set: 20×20: 8, 10 agents

Method	7 agents	9 agents
MT-TarMAC-IC3Net	39.19 ± 0.34	35.99 ± 0.39
MT-MAGIC	38.25 ± 1.37	35.41 ± 1.01
Meta-MAGIC	**36.54 ± 0.42**	**33.88 ± 0.09**

(c) Source Task Set: 20×20: 8, 10, 12 agents

Method	7 agents	9 agents	11 agents
MT-TarMAC-IC3Net	37.90 ± 0.31	34.57 ± 0.21	34.00 ± 0.22
MT-MAGIC	38.89 ± 0.71	34.33 ± 0.95	33.70 ± 0.26
Meta-MAGIC	**35.98 ± 0.36**	**33.15 ± 0.34**	**32.47 ± 0.14**

MAGIC can make fast adaptation within one or two episodes. Interestingly, we observe a similar phenomenon in most curves of MT-MAGIC and MT-TarMAC-IC3Net. Although MT-MAGIC and MT-TarMAC-IC3Net do not take in reward and action information and do not preserve hidden states between episodes, to some extent, their RNN structures can still take advantage of the insights obtained from the history information in the same episode. This aligns with the results of the single-agent 3D visual navigation environment reported in [48] and [49]. The agent can do fast adaptation and take fewer steps to reach the goal in a maze in the second episode after the exploration of the first episode. Then the performance stops increasing in the following episodes.

In longer-term, Meta-MAGIC keeps a lower bound of all the methods through ten episodes, and more curves of Meta-MAGIC (Figure 4.4(a), 4.4(b), 4.4(e), 4.4(g)) present decaying tendency in steps taken than those of MT-MAGIC or MT-TarMAC-IC3Net. What is more, Meta-MAGIC achieves lower steps after the first two episodes in all the curves, while the other two methods do not. This shows that our designed method adapts better to unseen environments in both short and long terms.

4.5.3 Fine-Tuning

Table 4.5: In this table, we display the results of fine-tuning in GRF and Predator-Prey unseen tasks. We compare the fine-tuning performance (mean±standard error) of MT-MAGIC to the training-from-scratch performance of MAGIC on selected tasks. We also enclose the average epochs taken to achieve the reported performance in the parenthesis.

(a) Fine-Tuning Results in GRF

Method	2 vs. 2, 3 vs. 3	2 vs. 2, 3 vs. 2, 3 vs. 3
	3 vs. 2	4 vs. 3
MT-MAGIC Fine-Tuning	**97.11** ± **1.19**% (**35**)	89.01 ± 2.14% (**207**)
MAGIC from Scratch	92.52 ± 4.98% (415)	**97.21** ± **0.82**% (429)

(b) Fine-Tuning Results in Predator-Prey, Source Task Set: 10 × 10: 4, 5, 6 agents

Method	3 agents	8 agents
MT-MAGIC Fine-Tuning	**13.15** ± **0.03**% (**129**)	**12.70** ± **0.01**% (**25**)
MAGIC from Scratch	13.41 ± 0.02% (257)	13.09 ± 0.15% (294)

(c) Fine-Tuning Results in Predator-Prey, Source Task Set: 20 × 20: 8, 10, 12 agents

Method	7 agents	9 agents	11 agents
MT-MAGIC Fine-Tuning	**34.38** ± **0.74**% (**33**)	**33.43** ± **0.47**% (**16**)	**32.32** ± **0.18**% (**18**)
MAGIC from Scratch	35.14 ± 0.08% (881)	33.99 ± 0.39% (911)	33.26 ± 0.50% (917)

Here, we present the fine-tuning results on selected new tasks compared to training from scratch. We follow the same settings of multi-task training to train a single task from scratch. The performance of MAGIC from scratch is chosen from the epoch with the highest success rate or the lowest steps taken during training. Note that we use a larger batch size but update the model fewer times within an epoch, compared to the experiments in chapter 3, so it takes more epochs to converge to good performance.

From Table 4.5, we can know that the pre-trained models of MT-MAGIC can be transferred well to new environments. MT-MAGIC Fine-Tuning achieves better performance than training from scratch while taking only 11.28% of its training epochs. The only outlier is in 4 vs. 3 of GRF, which is out of the distribution of the task set 2 vs. 2, 3 vs.2, and 3 vs. 3. The results show that training on multiple tasks by MT-MAGIC improves the model's generalization ability, and fine-tuning from this pre-trained model can quickly reach a better upper bound of the performance in unseen tasks.

4.6 Conclusion and Discussion

In this chapter, we explore a multi-task extension, MT-MAGIC, and a meta-learning extension to MAGIC, Meta-MAGIC. Both methods can perform multi-task learning across teams with different numbers of agents, and can also be generalized to unseen tasks of various sizes. To the best of our knowledge, Meta-MAGIC is the first RNN-based black-box adaptation framework for multi-agent meta-RL. We evaluate our methods and a baseline across two domains, GRF and Predator-Prey. Meta-MAGIC and MT-MAGIC outperform the MT-TarMAC-IC3Net by a considerable margin in multi-task training and generalize well to unseen tasks in most cases. Meta-MAGIC can do fast adaptation to new environments and keeps an upper bound on the performance of all methods in Predator-Prey. Fine-tuning from pre-trained models by MT-MAGIC can quickly achieve better performance on new tasks than training from scratch, with only 11.28% of training epochs.

Meanwhile, we would like to note that our methods and the baseline sometimes tend to be overfitting in GRF, and thus do not generalize well to unseen environments. There are three main reasons. Firstly, the sizes of our training task sets are relatively small, which limits the task diversity in GRF. Secondly, within GRF, the initialized positions of all agents in GRF are fixed, while agents in Predator-Prey are randomly spawned in the grid at the start of each episode, which provides more sufficient representations of the task distribution. Thirdly, GRF is more challenging, so a slightly-changed new scenario can be easily out of distribution of the task set and leads to failure.

CHAPTER 5
CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this thesis, I present related work, methodologies, experiments, and discussions from my research that push the frontier of reinforcement learning adaptable and scalable multi-agent communication.

In chapter 3 I propose a novel, fully differentiable communication protocol for MARL, MAGIC, that utilizes a Scheduler to solve the problems of when to communicate and whom to address messages to, and a Message Processor based on designed Graph Attention Networks to integrate and process messages. We evaluate our method and baselines in several environments including the unsolved and challenging GRF. Our method achieves state-of-the-art performance while communicating more efficiently than the baselines.

In chapter 4, I describe a multi-agent multi-task RL training scheme and a multi-agent meta-RL framework that are based on the communication protocol presented in chapter 3, named MT-MAGIC and Meta-MAGIC, respectively. Both methods utilize the advantages of the decentralized execution structure, and thus manage to generalize and adapt to unseen tasks with different numbers of agents. Meta-MAGIC initiatively explores using the RNN architecture as the adaptation function in multi-agent meta-RL. Through experiments, we find that MT-MAGIC enables the agents to learn efficiently in most tested tasks and generalize well on new tasks, and Meta-MAGIC can achieve improved multi-task performance and better adaptation to new tasks compared to MT-MAGIC and the baseline.

5.2 Future Work

Communication with noise, limited bandwidth, and delay. In the real world, communication is usually hindered by noise, limited bandwidth, and delay, and these problems might be more serious as the number of agents increases. Currently, our communication protocol proposed in chapter 3 has not considered such conditions. In the future, I seek to develop a multi-agent communication protocol that accommodates occasionally receiving noises and sending continuous or binary messages with a limited size. Meanwhile, to consider delays, we seed to define problems within the framework of Delay-Aware Markov Game proposed in [101].

Decentralized communication I also seek to enable the agents to conduct decentralized communication that does not require a centralized scheduler (even if light-weighted) to decide the dynamic communication patterns among agents during execution. The agents are expected to decide on targeted message receivers on their own from local observations. In this case, it would be difficult to learn a globally efficient communication strategy. A potential path to the solution is to use a centralized scheduler only in the training phase, and train a proxy scheduler for an individual agent to simulate the global scheduler, similar to the idea of the proxy encoder in [38].

Finding shared structures in multi-agent tasks. The multi-task design introduced in chapter 4 is mainly focused on accommodating tasks with different team sizes, and a specialized knowledge sharing mechanism has not been considered. In the future, I would seek to explore representation learning and task clustering techniques for multi-agent tasks. What is more, we can also gain insights from hierarchical MARL where multi-agent tasks can be structured by hierarchically learning a strategic latent variable [102, 39].

REFERENCES

[1] A. Oroojlooyjadid and D. Hajinezhad, “A review of cooperative multi-agent deep reinforcement learning,” *ArXiv*, vol. abs/1908.03963, 2019.

[2] E. Salas, T. Dickinson, S. A. Converse, and S. Tannenbaum, “Toward an understanding of team performance and training.,” 1992.

[3] N. Cooke, J. Gorman, C. W. Myers, and J. Duran, “Interactive team cognition,” *Cognitive science*, vol. 37 2, pp. 255–85, 2013.

[4] G. Tokadli and M. C. Dorneich, “Interaction paradigms: From human-human teaming to human-autonomy teaming,” *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, pp. 1–8, 2019.

[5] E. Salas, N. Cooke, and M. Rosen, “On teams, teamwork, and team performance: Discoveries and developments,” *Human Factors: The Journal of Human Factors and Ergonomic Society*, vol. 50, pp. 540–547, 2008.

[6] A. Das *et al.*, “Tarmac: Targeted multi-agent communication,” in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, PMLR, 2019, pp. 1538–1546.

[7] M. Dohler, A. Gkelias, and H. Aghvami, “A resource allocation strategy for distributed mimo multi-hop communication systems,” *IEEE Communications Letters*, vol. 8, no. 2, pp. 99–101, 2004.

[8] G. Korkmaz, E. Ekici, F. Özgüner, and Ü. Özgüner, “Urban multi-hop broadcast protocol for inter-vehicle communication systems,” in *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*, 2004, pp. 76–85.

[9] V. Mnih *et al.*, “Playing atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.

[10] D. Silver *et al.*, “Mastering the game of go with deep neural networks and tree search,” *nature*, vol. 529, no. 7587, p. 484, 2016.

[11] T. Haarnoja *et al.*, “Soft actor-critic algorithms and applications,” *arXiv preprint arXiv:1812.05905*, 2018.

[12] D. Kalashnikov *et al.*, “Scalable deep reinforcement learning for vision-based robotic manipulation,” in *Conference on Robot Learning*, PMLR, 2018, pp. 651–673.

[13] C. Wu, A. R. Kreidieh, K. Parvate, E. Vinitsky, and A. M. Bayen, "Flow: A modular learning framework for mixed autonomy traffic," *IEEE Transactions on Robotics*, 2021.

[14] R. Paleja, Y. Niu, A. Silva, C. Ritchie, S. Choi, and M. Gombolay, "Learning interpretable, high-performing policies for continuous control problems," *arXiv preprint arXiv:2202.02352*, 2022.

[15] C. Yu, J. Liu, S. Nemati, and G. Yin, "Reinforcement learning in healthcare: A survey," *ACM Computing Surveys (CSUR)*, vol. 55, no. 1, pp. 1–36, 2021.

[16] W. Xiong, T. Hoang, and W. Y. Wang, "Deeppath: A reinforcement learning method for knowledge graph reasoning," *arXiv preprint arXiv:1707.06690*, 2017.

[17] X. Wang, W. Chen, J. Wu, Y.-F. Wang, and W. Y. Wang, "Video captioning via hierarchical reinforcement learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4213–4222.

[18] P. Qin, W. Xu, and W. Y. Wang, "Robust distant supervision relation extraction via deep reinforcement learning," *arXiv preprint arXiv:1805.09927*, 2018.

[19] O. Vinyals *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[20] C. Berner *et al.*, "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680*, 2019.

[21] I.-C. Baek and K.-J. Kim, "Efficient multi-agent reinforcement learning using clustering for many agents," 2019.

[22] A. Singh, T. Jain, and S. Sukhbaatar, "Learning when to communicate at scale in multiagent cooperative and competitive tasks," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019.

[23] S. Sukhbaatar, R. Fergus, *et al.*, "Learning multiagent communication with backpropagation," *Advances in neural information processing systems*, vol. 29, 2016.

[24] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, Eds., 2016, pp. 2137–2145.

[25] P. Peng *et al.*, "Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games," *arXiv preprint arXiv:1703.10069*, 2017.

[26] R. Wang, X. He, R. Yu, W. Qiu, B. An, and Z. Rabinovich, "Learning efficient multi-agent communication: An information bottleneck approach," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, ser. Proceedings of Machine Learning Research, vol. 119, PMLR, 2020, pp. 9908–9918.

[27] D. Kim *et al.*, "Learning to schedule communication in multi-agent reinforcement learning," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019.

[28] T. L. Griffiths, F. Callaway, M. B. Chang, E. Grant, P. M. Krueger, and F. Lieder, "Doing more with less: Meta-reasoning and meta-learning in humans and machines," *Current Opinion in Behavioral Sciences*, vol. 29, pp. 24–30, 2019.

[29] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[30] E. Parisotto, J. L. Ba, and R. Salakhutdinov, "Actor-mimic: Deep multitask and transfer reinforcement learning," *arXiv preprint arXiv:1511.06342*, 2015.

[31] X. Yang, S. Kim, and E. Xing, "Heterogeneous multitask learning with joint sparsity constraints," *Advances in neural information processing systems*, vol. 22, 2009.

[32] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Boosting for transfer learning," in *Proceedings of the 24th International Conference on Machine Learning*, ser. ICML '07, Corvalis, Oregon, USA: Association for Computing Machinery, 2007, pp. 193–200, ISBN: 9781595937933.

[33] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *International conference on machine learning*, PMLR, 2015, pp. 1180–1189.

[34] J. Schmidhuber, "Evolutionary principles in self-referential learning, or on learning how to learn: The meta-meta-... hook," Ph.D. dissertation, Technische Universität München, 1987.

[35] S. Bengio, Y. Bengio, J. Cloutier, and J. Gescei, "On the optimization of a synaptic learning rule," in *Optimality in Biological and Artificial Networks?* Routledge, 2013, pp. 281–303.

[36] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in *International conference on machine learning*, PMLR, 2017, pp. 1126–1135.

[37] T. Zhang *et al.*, “Multi-agent collaboration via reward attribution decomposition,” *arXiv preprint arXiv:2010.08531*, 2020.

[38] P. Gu, M. Zhao, J. Hao, and B. An, “Online ad hoc teamwork under partial observability,” in *International Conference on Learning Representations*, 2021.

[39] B. Liu, Q. Liu, P. Stone, A. Garg, Y. Zhu, and A. Anandkumar, “Coach-player multi-agent reinforcement learning for dynamic team composition,” in *International Conference on Machine Learning*, PMLR, 2021, pp. 6860–6870.

[40] T. Wang, R. Liao, J. Ba, and S. Fidler, “Nervenet: Learning structured policy with graph neural networks,” in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, OpenReview.net, 2018.

[41] C. Blake, V. Kurin, M. Igl, and S. Whiteson, “Snowflake: Scaling gnns to high-dimensional continuous control via parameter freezing,” *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[42] W. Huang, I. Mordatch, and D. Pathak, “One policy to control them all: Shared modular policies for agent-agnostic control,” in *International Conference on Machine Learning*, PMLR, 2020, pp. 4455–4464.

[43] H. Jia, B. Ding, H. Wang, X. Gong, and X. Zhou, “Fast adaptation via meta learning in multi-agent cooperative tasks,” in *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, IEEE, 2019, pp. 707–714.

[44] M. Andrychowicz *et al.*, “Learning to learn by gradient descent by gradient descent,” *Advances in neural information processing systems*, vol. 29, 2016.

[45] Y. Hu, M. Chen, W. Saad, H. V. Poor, and S. Cui, “Distributed multi-agent meta learning for trajectory design in wireless drone networks,” *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 10, pp. 3177–3192, 2021.

[46] M. S. Munir, N. H. Tran, W. Saad, and C. S. Hong, “Multi-agent meta-reinforcement learning for self-powered and sustainable edge computing systems,” *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3353–3374, 2021.

[47] Q. Zhang and D. Chen, "A meta-gradient approach to learning cooperative multi-agent communication topology," in *Fifth Workshop on Meta-Learning at the Conference on Neural Information Processing Systems*, 2021.

[48] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "Rl^2: Fast reinforcement learning via slow reinforcement learning," *arXiv preprint arXiv:1611.02779*, 2016.

[49] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," *arXiv preprint arXiv:1707.03141*, 2017.

[50] M. Gombolay, A. Bair, C. Huang, and J. Shah, "Computational design of mixed-initiative human–robot teaming that considers human factors: Situational awareness, workload, and workflow preferences," *The International Journal of Robotics Research*, vol. 36, pp. 597–617, 2017.

[51] M. Gombolay, R. Wilcox, and J. Shah, "Fast scheduling of robot teams performing tasks with temporospatial constraints," *IEEE Transactions on Robotics*, vol. 34, pp. 220–239, 2018.

[52] M. Gombolay *et al.*, "Human-machine collaborative optimization via apprenticeship scheduling," *ArXiv*, vol. abs/1805.04220, 2018.

[53] D.-K. Kim *et al.*, "A policy gradient algorithm for learning to learn in multiagent reinforcement learning," *ArXiv*, vol. abs/2011.00382, 2020.

[54] E. Seraj and M. Gombolay, "Coordinated control of uavs for human-centered active sensing of wildfires," *2020 American Control Conference (ACC)*, pp. 1845–1852, 2020.

[55] L. G. Strickland, C. E. Pippin, and M. Gombolay, "Learning to steer swarm-vs.-swarm engagements," in *AIAA Scitech 2021 Forum*. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2021-0165.

[56] R. Paleja, M. Ghuy, N. Ranawaka Arachchige, R. Jensen, and M. Gombolay, "The utility of explainable ai in ad hoc human-machine teaming," 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. 610–623.

[57] E. Seraj *et al.*, "Learning efficient diverse communication for cooperative heterogeneous teaming," in *AAMAS*, 2021.

[58] L. Busoniu, R. Babuka, and B. D. Schutter, “A comprehensive survey of multiagent reinforcement learning,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, pp. 156–172, 2008.

[59] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, “Multi-agent actor-critic for mixed cooperative-competitive environments,” in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon *et al.*, Eds., 2017, pp. 6379–6390.

[60] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, “Counterfactual multi-agent policy gradients,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[61] S. Iqbal and F. Sha, “Actor-attention-critic for multi-agent reinforcement learning,” in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, PMLR, 2019, pp. 2961–2970.

[62] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, J. G. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, PMLR, 2018, pp. 1856–1865.

[63] J. Jiang and Z. Lu, “Learning attentional communication for multi-agent cooperation,” in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 7265–7275.

[64] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model,” *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.

[65] Z. Wang and M. Gombolay, “Heterogeneous graph attention networks for scalable multi-robot scheduling with temporospatial constraints,” in *RSS 2020*, 2020.

[66] Z. Wang and M. Gombolay, “Learning scheduling policies for multi-robot coordination with graph attention networks,” *IEEE Robotics and Automation Letters*, vol. 5, pp. 4509–4516, 2020.

[67] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. Yu, “A comprehensive survey on graph neural networks,” *IEEE transactions on neural networks and learning systems*, 2020.

[68] J. Sheng *et al.*, “Learning structured communication for multi-agent reinforcement learning,” *arXiv preprint arXiv:2002.04235*, 2020.

[69] J. Jiang, C. Dun, and Z. Lu, “Graph convolutional reinforcement learning for multi-agent cooperation,” *arXiv preprint arXiv:1810.09202*, vol. 2, no. 3, 2018.

[70] A. Malysheva, T. T. Sung, C.-B. Sohn, D. Kudenko, and A. Shpilman, “Deep multi-agent reinforcement learning with relevance graphs,” *arXiv preprint arXiv:1811.12557*, 2018.

[71] S. Li, J. Gupta, P. Morales, R. Allen, and M. Kochenderfer, “Deep implicit coordination graphs for multi-agent reinforcement learning,” *ArXiv*, vol. abs/2006.11438, 2020.

[72] H. Ryu, H. Shin, and J. Park, “Multi-agent actor-critic with hierarchical graph attention network,” *arXiv preprint arXiv:1909.12557*, 2019.

[73] Y. Liu, W. Wang, Y. Hu, J. Hao, X. Chen, and Y. Gao, “Multi-agent game abstraction via graph attention neural network,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 7211–7218.

[74] S. Omidshafiei, J. Pazis, 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*, PMLR, 2017, pp. 2681–2690.

[75] D. Schwab, Y. Zhu, and M. Veloso, “Tensor action spaces for multi-agent robot transfer learning,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 5380–5386.

[76] J. Cui, W. Macke, H. Yedidsion, A. Goyal, D. Urielli, and P. Stone, “Scalable multi-agent driving policies for reducing traffic congestion,” *arXiv preprint arXiv:2103.00058*, 2021.

[77] P. Stone, G. A. Kaminka, S. Kraus, and J. S. Rosenschein, “Ad hoc autonomous agent teams: Collaboration without pre-coordination,” in *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[78] J. X. Wang *et al.*, “Learning to reinforcement learn,” *arXiv preprint arXiv:1611.05763*, 2016.

[79] J. Foerster, G. Farquhar, M. Al-Shedivat, T. Rocktäschel, E. Xing, and S. Whiteson, "Dice: The infinitely differentiable monte carlo estimator," in *International Conference on Machine Learning*, PMLR, 2018, pp. 1529–1538.

[80] J. Rothfuss, D. Lee, I. Clavera, T. Asfour, and P. Abbeel, "Promp: Proximal meta-policy search," *arXiv preprint arXiv:1810.06784*, 2018.

[81] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen, "Efficient off-policy meta-reinforcement learning via probabilistic context variables," in *International conference on machine learning*, PMLR, 2019, pp. 5331–5340.

[82] L. Zintgraf, M. Igl, K. Shiarlis, A. Mahajan, K. Hofmann, and S. Whiteson, "Variational task embeddings for fast adapta-tion in deep reinforcement learning," in *International Conference on Learning Representations Workshop (ICLRW)*, 2019.

[83] J. Humplik, A. Galashov, L. Hasenclever, P. A. Ortega, Y. W. Teh, and N. Heess, "Meta reinforcement learning as task inference," *arXiv preprint arXiv:1905.06424*, 2019.

[84] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv preprint arXiv:1803.02999*, 2018.

[85] M. Kayaalp, S. Vlaski, and A. H. Sayed, "Dif-maml: Decentralized multi-agent meta-learning," *arXiv preprint arXiv:2010.02870*, 2020.

[86] J. Huang, W. Huang, D. Wu, and L. Lan, "Meta actor-critic framework for multi-agent reinforcement learning," in *2021 4th International Conference on Artificial Intelligence and Pattern Recognition*, 2021, pp. 636–643.

[87] J. Ackermann, V. Gabler, T. Osa, and M. Sugiyama, "Reducing overestimation bias in multi-agent domains using double centralized critics," *arXiv preprint arXiv:1910.01465*, 2019.

[88] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[89] M. Al-Shedivat, T. Bansal, Y. Burda, I. Sutskever, I. Mordatch, and P. Abbeel, "Continuous adaptation via meta-learning in nonstationary and competitive environments," *arXiv preprint arXiv:1710.03641*, 2017.

[90] D. K. Kim *et al.*, "A policy gradient algorithm for learning to learn in multiagent reinforcement learning," in *International Conference on Machine Learning*, PMLR, 2021, pp. 5541–5550.

[91] W.-C. Tseng, W. Wei, D.-C. Juan, and M. Sun, "Meta-cpr: Generalize to unseen large number of agents with communication pattern recognition module," *arXiv preprint arXiv:2112.07222*, 2021.

[92] Y. Niu, R. Paleja, and M. Gombolay, "Multi-agent graph-attention communication and teaming," in *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, 2021, pp. 964–973.

[93] D. Adjodah *et al.*, "Communication topologies between learning agents in deep reinforcement learning," *arXiv preprint arXiv:1902.06740*, 2019.

[94] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine learning proceedings 1994*, Elsevier, 1994, pp. 157–163.

[95] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[96] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, OpenReview.net, 2018.

[97] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *ArXiv*, vol. abs/1811.03378, 2018.

[98] K. Kurach *et al.*, "Google research football: A novel reinforcement learning environment," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 4501–4510.

[99] D. Britz, A. Goldie, M.-T. Luong, and Q. Le, "Massive exploration of neural machine translation architectures," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 1442–1451.

[100] D. Pickem, E. Squires, and M. Egerstedt, "The robotarium : An open , remote-access , multi-robot laboratory," 2016.

[101] B. Chen, M. Xu, Z. Liu, L. Li, and D. Zhao, "Delay-aware multi-agent reinforcement learning for cooperative and competitive environments," *arXiv preprint arXiv:2005.05441*, 2020.

[102] J. Yang, I. Borovikov, and H. Zha, "Hierarchical cooperative multi-agent reinforcement learning with skill discovery," *arXiv preprint arXiv:1912.03558*, 2019.