Robust and Scalable Multiagent Reinforcement Learning in Adversarial Scenarios

by

Macheng Shen

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2022

Author .....
Department of Mechanical Engineering
May 13, 2022

Certified by .....
Jonathan P. How
R. C. Maclaurin Professor of Aeronautics and Astronautics
Thesis Supervisor

Certified by .....
John Leonard
Samuel C. Collins Professor of Mechanical and Ocean Engineering
Thesis Supervisor

Certified by .....
George Barbastathis
Singapore Research Professor of Optics; Professor of Mechanical Engineering
Thesis Committee Chair

Accepted by .....
Nicolas Hadjiconstantinou
Graduate Officer, Department of Mechanical Engineering

Robust and Scalable Multiagent Reinforcement Learning in Adversarial Scenarios

by

Macheng Shen

Submitted to the Department of Mechanical Engineering
on May 13, 2022, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Mechanical Engineering

Abstract

Multiagent decision-making is a ubiquitous problem with many real-world applications, such as autonomous driving, multi-player video games, and robot team sports. Key challenges of multiagent learning include the presence of uncertainty in the other agent's behaviors and the curse of dimensionality caused by the high dimensionality of the joint observation, action, and policy space. These challenges are accentuated even further in adversarial scenarios due to the unknown agent intents and unexpected, possibly adversarial behaviors. This thesis presents approaches for robust and scalable multiagent learning with the goal of efficiently building autonomous agents that can operate robustly in adversarial scenarios. The capability of accurately inferring unknown agent intents by observing its behaviors is critical for robust decision-making. A challenge in this case is the high uncertainty in an adversary's actual behavior, including potential deception, which could be significantly different from an a priori behavior model. Capturing the interaction between the ego-agent and the adversaries as well as the reasoning of available information to both agents is critical for modeling this deceptive behavior. This thesis addresses this intent recognition problem using a game-theoretic opponent modeling approach based on a new diversity-driven belief-space ensemble training technique that is used to achieve robustness against deception. To extend the ensemble approach to scenarios with multiple agents, this thesis presents a scalable multiagent learning technique that facilitates near-optimal joint policy learning through a sparse-attention mechanism. This mechanism results in focused parameter update, which significantly improves sample-efficiency. Moreover, this thesis also contributes a novel implicit ensemble training approach that leverages multi-task learning and deep generative policy distribution to achieve better robustness at a much lower computation and memory cost compared with previous ensemble techniques. The combination of robust intent recognition and scalable multiagent learning leads to robust and scalable offline policy learning. However, a fully autonomous agent also needs to be able to continually learn from (and adapt to) new environments and peer agents. Thus this thesis also presents to a safe adaptation approach that enables adaptation to a new opponent while maintaining low exploitabil-

ity for any possible opponent exploitation in adversarial scenarios. The contributions presented in this thesis facilitate building autonomous agents that can make robust decisions under competitive multiagent scenarios with uncertainty and safely adapt to previously unseen peer agents, through computationally efficient learning.

Thesis Supervisor: Jonathan P. How
Title: R. C. Maclaurin Professor of Aeronautics and Astronautics

Thesis Supervisor: John Leonard
Title: Samuel C. Collins Professor of Mechanical and Ocean Engineering

Thesis Committee Chair: George Barbastathis
Title: Singapore Research Professor of Optics; Professor of Mechanical Engineering

Acknowledgments

It has been five years since I started as a graduate student at MIT. During this period, I have been taking courses, learning how to do research, running experiments, writing papers and reports, and getting to know many new friends and nice people. This is definitely an unforgettable but challenging journey in my life. I have experienced joy when my first paper was accepted, but also stressful moment and struggle when I got stuck on my research. Without the support and help from my advisor, mentors, labmates, friends, and family, I would not be able to accomplish this thesis.

First, I would like to thank my research advisor, Professor Jonathan How. Jon is a responsible advisor who devotes a lot of time to research and mentorship to help his students become successful in graduate school. Jon sets a high standard for us and teaches us the methodology of doing research which will have a continual benefit to me in my future career. I am also thankful for Jon's insightful feedback on my research ideas and progress during our weekly meeting, which is incredibly helpful for my research. Thank you, Jon, for your guidance and support during my PhD years.

Thank you to my thesis committee: Professor George Barbastathis and Professor John Leonard, who provided thoughtful advice on my research during the committee meetings and were always responsive to my requests.

I was lucky to join the Aerospace Controls Laboratory (ACL) to meet and become friends with many wonderful people. I would like to say thank you in particular to Dr. Kasra Khosoussi, Dongki Kim, Dr. Michael Everett, who influenced me as my peer mentors and also supported me mentally during the most difficult time of my PhD. I would like to say thank you to Dr. Chuangchuang Sun for being a nice peer to collaborate with and learn from, Dr. Kaveh Fathian for mentorship on writing good technical reports, and to my peer labmates Dongki Kim, Yulun Tian, and Jesus Tordesillas Torres for our friendship during the past five years. Thanks to everyone else in the lab who are always friendly and helpful, making me feel at home.

Thank you to MechE's retired graduate coordinator Leslie Regan, who treats MechE's graduate students as if we were her own children. Thank you to my room-

mate and my dear friends at MIT and Harvard.

Thank you to the Mechanical Engineering Department at MIT, Scientific Systems Company, Inc. (under research agreement # SC-1661-04), and DARPA (ARL DCIST under Cooperative Agreement Number W911NF-17-2-0181, ShELL under grant number HR00112190115) for financially sponsoring my research throughout my PhD.

Most importantly, I would like to say special thanks to my parents. Thank you for your unconditional love and support during my PhD and throughout my whole life!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Multiagent decision-making is a ubiquitous problem with many real-world applications. Despite great breakthroughs in building superhuman AI in recent years such as AlphaZero [1] and Libratus [2], learning to make good decisions in real-world multiagent scenarios, especially adversarial scenarios, remains a challenging problem. This thesis addresses several technical challenges to facilitate efficient learning that enables near-optimal decision-making in multiagent adversarial scenarios.

In particular, this thesis focuses on robust and scalable multiagent reinforcement learning (MARL) in adversarial scenarios. The robustness problem in machine learning arises from the distributional mismatch between training data and testing data [3], which leads to significant performance degradation in the testing phase. In the MARL context, this mismatch corresponds to shifting of agent behaviors from training to testing [4], as well as high uncertainty of agent intents. As a result, understanding agent intents and reasoning about possible agent behaviors are critical capabilities to achieve robustness in multiagent adversarial scenarios, which is the first technical challenge addressed in this thesis.

Besides robustness, scalability is a second issue that prevents the application of current MARL approaches to complicated multiagent scenarios. The training and inference of superhuman AI such as AlphaZero and Libratus require thousands of

CPUs and GPUs [1, 2], which is prohibitively demanding. The ensemble training technique used in AlphaZero is one of the major reasons for this huge computation cost [1], where the latest AlphaZero agent is trained against hundreds of its previous policies. The motivation of using ensemble training is to achieve out-of-distribution robustness against previously unseen opponent policies. To achieve this goal, the agent must be trained against a diverse portfolio of strong opponent behaviors to generalize its training performance to testing. Quite a few empirical studies [5–9] show that this ensemble training process effectively improves robustness against out-of-distribution opponents, which is critical for agents to survive in adversarial scenarios. In general multiagent adversarial scenarios that involve heterogeneous agents, robustifying the ego-agent's policy requires an ensemble of strong opponent policies, while learning strong opponent policies requires an ensemble of strong ego-agent's policies. As a result, each agent has to learn an ensemble of policies, which significantly increases the computation and memory requirements. Besides, extending these techniques to scenarios with more than two agents and possibly many agents is an additional challenge that significantly increases the number of required samples and computation [10]. Therefore, the capability of scaling up multiagent robust policy learning with multiple agents and multiple policies is critical to the development of practical MARL approaches for complicated multiagent scenarios. This thesis presents approaches for addressing these two sub-problems of scalable MARL: a computation-and-memory-efficient approach for ensemble-based policy learning and a sample-efficient joint policy learning approach for multiagent scenarios with many agents. These two approaches enable efficient offline learning of robust policies for multiagent coordination and competition in adversarial scenarios.

Moreover, a fully autonomous agent needs to continually learn from and adapt to new environments and peer agents to remain competitive in adversarial scenarios. While there have been a variety of recent works on fast adaptation via meta-learning [11–13] and multiagent meta-learning [14, 15], maintaining low exploitability and high robustness against a previously unseen and possibly co-adapting adversary during adaptation is an unsolved but important technical challenge in adversarial sce-

narios. This thesis presents a safe adaptation approach that addresses this challenge, which constitutes a complement to meta-learning towards building safe and adaptive autonomous agents.

The following sections introduce the statement of problems addressed by this thesis in further detail (Section 1.2), describe the technical contributions (Section 1.3), and the thesis structure (Section 1.4).

1.2 Problem Statement

This thesis presents solutions to address the problem of how to design robust and scalable RL algorithms for multiagent systems, which can be decomposed into answers to the following sub-problems: 1) How to robustly identify the intent of an opponent agent in adversarial scenarios? 2) How to scale up MARL for learning near-optimal joint policy in scenarios involving more than two, possibly many agents? 3) How to reduce the computation complexity of ensemble training to achieve efficient robust policy learning? 4) How to safely adapt to an opponent while maintaining low exploitability during the adaptation? The following sections elaborate on each sub-problem, while the technical problem formulations are left for the subsequent chapters.

1.2.1 Robust Intent Recognition under Asymmetric-information Adversarial Scenarios

Most existing multiagent learning approaches do not account for the uncertainty of opponent intents [2, 5, 16]. However, many safety-critical multiagent scenarios, e.g. autonomous driving [17], urban security [18] and cyber security [19], require reasoning about opponent intents to make good decisions. Moreover, these scenarios typically involve asymmetric information, where the opponent has an information advantage over the ego-agent, which incentivizes deceptive opponent behaviors. The research problem is to develop a robust intent recognition approach that achieves high accuracy

in asymmetric-information scenarios for making optimal decisions against a previously unseen (possibly deceptive) opponent.

1.2.2 Near-optimal Multiagent Joint Policy Learning with Many Agents

The complexity of a multiagent scenario largely depends on the number of agents involved in the interaction. As a result, learning good joint policy is difficult in scenarios with a large number of agents coordinating or competing with each other. However, many real-world multiagent scenarios involve many agents. To bridge this gap, the research problem is to develop a scalable multiagent learning algorithm that enables learning of near-optimal joint policy in scenarios with a large number of agents.

1.2.3 Efficient Ensemble-based Robust Policy Learning

Policy learned from MARL could be susceptible to overfitting to the opponent policy during the training [20], which leads to performance degradation against a previously unseen opponent when deployed in the real world. Ensemble training [5, 21, 22] is an effective approach to mitigate overfitting and improve robustness. However, the computation overhead of existing ensemble approaches [5, 21] is significant, which prevents efficient robust policy learning in complicated multiagent scenarios. The research problem is to develop a new ensemble training approach with significantly reduced computation and memory overhead, while still achieving high robustness of the learned policy.

1.2.4 Safe Adaptation against Adversarial Exploiter

Adaptation is a critical capability to achieve long-term competitiveness in an ever-changing real-world environment. In multiagent adversarial scenarios, adaptation corresponds to exploiting the sub-optimality of the opponent. However, as the ego-agent exploits its opponent, the exploitability of its own policy also tends to increase, which is known as the trade-off between exploitation and exploitability [23]. The

research problem is to develop a safe adaptation approach for exploiting a sub-optimal opponent while maintaining low exploitability against any other possible opponent exploitation during the adaptation phase.

1.3 Contributions

Contribution 1: Robust Intent Recognition via Game-theoretic Ensemble Opponent Modeling This contribution [18], based on our prior work [24], addresses the problem of robust intent recognition against a previously unseen opponent. The key idea is a game-theoretic opponent modeling approach that captures more sophisticated adversarial behaviors than what a single-agent opponent model can capture. In addition, a diversity-driven ensemble training approach is developed to capture a wide spectrum of possible adversarial behaviors including deception in asymmetric-information scenarios, which effectively improves the accuracy of intent recognition and robustness of the ego-agent's policy against previously unseen adversaries. Specifically, the proposed approach increases the intent recognition precision by about 30% and recall by about 60% against a deceptive adversary compared with existing intent recognition approach based on single opponent modeling without ensemble.

Contribution 2: Near-optimal Multiagent Joint Policy Learning with Sparse Attentional Graph Neural Network This contribution [25] addresses the problem of learning near-optimal joint policy in scenarios with a large number of agents. The key innovation is a novel sparse attention mechanism, which enables selective attention to a small subset of peer agents' information that is critical to the ego-agent's decision-making. As a result, the sample-efficiency is significantly improved so that we can scale up MARL to scenarios with many agents without significant compromise on optimality. We demonstrated learning of joint policy that achieves high performance with this new approach in scenarios with 20-30 agents, where previous approaches perform poorly. Specifically, the proposed approach increases the success

rate by about 70% in cooperative scenarios compared with baselines that do not exploit sparseness. In the competitive scenario, the proposed approach (overall score: 26) wins against the baselines (overall scores: -6 and -20) with significant margin.

Contribution 3: Efficient Ensemble-based Robust Policy Learning through Implicit Ensemble Training This contribution [26] addresses the problem of achieving ensemble-based robust policy learning with significantly reduced computation and memory overhead so as to scale up robust multiagent learning to more complicated scenarios. The key innovation is a novel multi-tasking deep generative model for representing a policy distribution implicitly within a single network architecture. The main benefits are improved sample-efficiency and policy diversity, which leads to significantly improved training efficiency and robustness of the learned policy. With this capability, we are able to learn robust policy within hours which would have taken weeks using previous ensemble training approaches.

Contribution 4: Safe Adaptation against Adversarial Exploiter via Ensemble Regularized Opponent Distribution Modeling This contribution [27] addresses the problem of safe adaptation for exploiting a sub-optimal opponent while maintaining low exploitability during the adaptation phase. The key innovation is a novel Bayesian formulation of MARL where the posterior distribution over the opponent policy not only captures the behavior of the opponent via maximizing the likelihood of the observed data but also is regularized to stay close to a robust policy distribution. As a result, training against this regularized opponent model enables adaptation to the opponent as well as robustness against any possible opponent exploitation. Specifically, the proposed approach increases the overall metric of adaptation and robustness by about 50% and 40% respectively in two multiagent robotic domains compared with two reference approaches that only optimize one of the adaptation and robustness metrics.

1.4 Thesis Structure

The rest of the thesis is structured as follows:

1. Chapter 2 is the preliminaries on the decision-making framework of MARL as the foundation of the approaches developed in this thesis.

2. Chapter 3 presents the approach for robust intent recognition via game-theoretic ensemble opponent modeling (Contribution 1).

   The content of this chapter is based on: Macheng Shen, and Jonathan P. How. "Robust opponent modeling via adversarial ensemble reinforcement learning." Proceedings of the International Conference on Automated Planning and Scheduling. Vol. 31. 2021. URL: `https://ojs.aaai.org/index.php/ICAPS/article/view/16006/15817`.

3. Chapter 4 presents the approach for near-optimal multiagent joint policy learning with a sparse attentional graph neural network (Contribution 2).

   The content of this chapter is based on: Chuangchuang Sun*, Macheng Shen*, and Jonathan P. How. "Scaling up multiagent reinforcement learning for robotic systems: Learn an adaptive sparse communication graph." IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020. URL: `https://ieeexplore.ieee.org/document/9341303`.

4. Chapter 5 presents the implicit ensemble training approach for efficient ensemble-based robust policy learning (Contribution 3).

   The content of this chapter is based on: Macheng Shen and Jonathan P How. "Implicit ensemble training for efficient and robust multiagent reinforcement learning." 2021 International Conference on Machine Learning Workshop on Uncertainty and Robustness in Deep Learning (ICML-UDL), URL: `http://www.gatsby.ucl.ac.uk/~balaji/udl2021/accepted-papers/UDL2021-paper-019.pdf` (Extended version to be submitted to Transactions on Machine Learning Research).

5. Chapter 6 presents the approach for safe adaptation against adversarial exploiter via ensemble regularized opponent distribution modeling (Contribution 4).

   The content of this chapter is based on: Macheng Shen and Jonathan P How. "Safe adaptation in multiagent competition." IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), under review, URL: `https://arxiv.org/pdf/2203.07562.pdf`, 2022.

6. Chapter 7 summarizes the contributions presented in this thesis.

Given the diversity of the topics covered, a literature review on related works is provided for each sub-topic in Chapters 3 to 6.

Chapter 2

Preliminaries

This chapter presents the single-agent and multiagent decision-making frameworks, Markov Decision Process (MDP), Markov Game (MG) and Bayesian Game (BG), as well as the framework of MARL, as the foundation of the approaches presented in the subsequent chapters.

2.1 Markov Decision Process

A Markov Decision Process is represented by a tuple $\langle \mathcal{S}, \mathcal{A}\ \mathcal{P}^T, \mathcal{R} \rangle$ [28], where,

- $\mathcal{S}$ is the set of state,
- $\mathcal{A}$ is the action space,
- $\mathcal{P}^T(s'|s,a)$ is the Markovian state transition,
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function.

The agent in the MDP has a stochastic policy $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0,1]$, which maps each state to a probability distribution over the action space. The objective of this agent is to maximize the expected discounted cumulative reward $R = \sum_{t=0}^{T} \gamma^t r^t$,

$$J = \mathbb{E}_{s\sim p^\pi, a\sim\pi}\left[R(s,a)\right]. \tag{2.1}$$

2.2 Markov Game

A Markov game is a multiagent extension of MDP. A Markov Game for N agents is defined by $G = \langle \mathcal{S}, \{\mathcal{A}_i\}, \{\mathcal{O}_i\}, \mathcal{P}^T, \mathcal{P}^O, \{R_i\}\rangle$ [29], where,

- $\mathcal{S}$ is the set of state,
- $\mathcal{A}_i$ is the action space of each agent, and we use $\boldsymbol{a} = \langle a_1, \ldots, a_n\rangle$ to denote the joint action, with $a_i \in \mathcal{A}_i$,
- $\mathcal{O}_i$ is the observation space for each agent, and we use $\boldsymbol{o} = \langle o_1, \ldots, o_n\rangle$ to denote the joint observation, with $o_i \in \mathcal{O}_i$,
- $\mathcal{P}^T(s'|s, \boldsymbol{a})$ is the Markovian state transition, and $\mathcal{P}^O(\boldsymbol{o}|s, \boldsymbol{a})$ is the observation probability,
- $\mathcal{R}_i : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function of each agent.

Each agent has a stochastic policy $\pi_i : \mathcal{O}_i \times \mathcal{A}_i \mapsto [0, 1]$, and a reward function $r_i : \mathcal{S} \times \mathcal{A}_i \mapsto \mathbb{R}$.

2.3 Bayesian Game

A Bayesian game (BG) [30] is defined by $G = \langle \mathcal{I}, \langle \mathcal{S}, \mathcal{H}\rangle, \{b^0\}, \{\mathcal{A}_i\}, \{\mathcal{O}_i\}, \mathcal{P}^T, \mathcal{P}^O, \{R_i\}\rangle$, where,

- $\mathcal{I}$ is a finite set of agents indexed by $1, \ldots, n$,
- $\Omega = \langle \mathcal{S}, \mathcal{H}\rangle$ is the set of state of nature, which includes the physical states and the agent hidden states,
- $b^0 \in \Delta(\mathcal{S} \times \mathcal{H})$ is the common prior probability distribution over Ω, where Δ is the probabilistic simplex,
- $\mathcal{A}_i$, $\mathcal{O}_i$, $\mathcal{P}^T(s'|s, \boldsymbol{a})$ and $\mathcal{P}^O(\boldsymbol{o}|s, \boldsymbol{a})$ are the same as those defined in Markov Game,
- $\mathcal{R}_i : \Omega \times \mathcal{A} \to \mathbb{R}$ is the reward function of each agent.

Each agent has a belief-space stochastic policy $\Delta\Omega \times \mathcal{A}_i \mapsto [0, 1]$, and a reward function $r_i : \Omega \times \mathcal{A}_i \mapsto \mathbb{R}$.

2.4 Multiagent Reinforcement Learning (MARL)

The objective of each agent is to maximize its own cumulative reward $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$ with discount factor γ and time horizon T [5]. As a result, the learning problem is formulated as finding a joint policy $\boldsymbol{\pi} = \{\pi_i\}^{i=1:N}$, where each policy maximizes its own reward,

$$J_i = \mathbb{E}_{s\sim p^{\boldsymbol{\pi}}, a_i\sim\pi_i, \boldsymbol{a}_{-i}\sim\boldsymbol{\pi}_{-i}} \left[R_i(s, \boldsymbol{a})\right], \tag{2.2}$$

with $p^{\boldsymbol{\pi}}$ being the transition dynamics and the subscript $-i$ denotes the set $\{j|j \neq i, j = 1, 2, \ldots, N\}$.

2.5 MARL with policy distribution

In practice, learning with a single joint policy typically leads to over-fitting between policies and poor generalization to previously unseen policies. A variety of empirical studies [5, 9, 16, 21, 31–33] and a recent theoretical study [8] suggest that it is necessary to maintain a diverse set of policies for each agent to improve the strength of the joint policy via MARL. Therefore, instead of learning a single joint policy, we consider the following objective function which learns a distribution of policies for each agent,

$$J_i = \mathbb{E}_{s\sim p^{\boldsymbol{\pi}}, \boldsymbol{a}\sim\boldsymbol{\pi}, \boldsymbol{\pi}\sim\mathcal{P}^{\boldsymbol{\Pi}}} \left[R_i(s, \boldsymbol{a})\right], \tag{2.3}$$

where $\mathcal{P}^{\boldsymbol{\Pi}}$ is a joint distribution over the joint policy space $\boldsymbol{\Pi} = \Pi_1 \times \Pi_2 \ldots \times \Pi_N$. Each agent is learning its own policy distribution Π_i to optimize its objective J_i subject to the joint distribution $\boldsymbol{\Pi}$.

Note that the feasibility set of Eq. 2.3 contains that of Eq. 2.2, which is analogous to the relationship between a mixed-strategy Nash Equilibrium and a pure-strategy Nash Equilibrium [34]. This relationship also suggests that Eq. 2.3 is a generalized learning objective of Eq. 2.2.

2.6 Summary

This chapter briefly introduces the decision-making frameworks including MDP, MG, BG, and MARL as preliminaries for the subsequent chapters.

Chapter 3

Robust Intent Recognition via Game-theoretic Ensemble Opponent Modeling

3.1 Introduction

Recent advances in deep reinforcement learning (DRL) have achieved breakthroughs in solving challenging decision-making problems in both single-agent environments [35–37] and multiagent games [6, 9, 16, 38, 39]. Among these multiagent games, some have fully observable states, and others includes hidden states that are partially-observable (or unobservable) to some agents. Nevertheless, the agent types (intents) in these games are known to all the agents. For example, Go is a zero-sum game where the two players compete with each other, where player 1 (hereafter we refer to as the ego-agent or the protagonist agent interchangeably) knows that its opponent player 2 is playing an adversarial role that tries to minimize player 1's winning probability. However, there also exist many important multiagent scenarios in which some of the agent types are uncertain or not well known to all. For example, in a cyber-security scenario, the network administrator (ego-agent/protagonist) observes signals that could have been sent by normal users (neutral) or by malicious attacker agents

(adversary), without knowing the type of each individual (hereafter, we refer to the second agent of uncertain types as opponent).

Given this type uncertainty, the network administrator needs to infer the identity of the signal sender before making the decision of blocking the signal or not, which significantly increases the complexity of the network administrator's decision-making process.

An opponent model is typically required, either explicitly [40–43], or implicitly [44–47], to make such inference feasible. An explicit modeling approach tries to model the opponent's policy directly, while an implicit model instead estimates intermediate statistics such as the anticipated value of the ego-agent's policy against the opponent [48]. In the scenarios with uncertain opponent types, the implicit opponent modeling approach could be predicting opponent's hidden type from the observation of opponent's states and actions, while an explicit modeling approach also permit this hidden type inference by using the Bayes' rule. We will show evidence that explicit opponent modeling leads to superior performance compared with implicit opponent modeling.

One simple explicit opponent modelling approach is to treat the opponent as a goal-directed MDP agent by specifying a reward function and then learn/use the optimal goal-achieving policy as the opponent model. This approach has two limitations. First, an adversarial opponent has the incentive of concealing its identity through disguise behavior. For example, an attacker might mimic a normal user's behavior to avoid being detected immediately, while a simple goal-directed reward cannot capture this strategic behavior. We argue that a game-theoretic opponent model which captures the full interaction and strategic reasoning between the ego-agent and the opponent agent is superior than a goal-directed opponent model of single-agent perspective. The second limitation of using the opponent's optimal policy as the opponent model is the high sensitivity with respect to modeling error, which could result in significant performance degradation against a previously unseen opponent. To mitigate sensitivity and improve the robustness of the opponent model, we propose to learn an ensemble of diverse opponent policies through multiagent reinforcement

learning (MARL) and distill these policies to form an 'average' opponent model for inferring the hidden type of an opponent from its observed state-action history.

This work presents an algorithmic framework for learning robust policies in multi-agent scenarios with uncertain opponent types. We focus on the scenarios where the opponent type is unknown to the ego-agent, but the ego-agent's identity is certain to the opponent. We are interested in learning an opponent model for the ego-agent to update its belief on the opponent type and defend robustly against previously unseen opponent. This setting is an abstraction of security-domain scenarios, but has seldom been well-studied in the context of MARL.

3.2 Background

This section reviews the preliminary of the decision-making framework and solution techniques.

3.2.1 Decision-making Framework

The multiagent scenarios with uncertain opponent types as described in the Introduction Section is a special case of Bayesian Games as described in Section 2.3, where $\Omega = \langle \mathcal{S}, \mathcal{H} \rangle$ is the set of state of nature, which includes the physical states and the agent hidden states corresponding to agent types in our problem. Note that the reward function $\mathcal{R}_i : \Omega \times \mathcal{A} \to \mathbb{R}$ depends on both the state, action and the hidden agent types. The same action against different types of opponents could result in rewards of opposite signs (positive v.s. negative). As a result, correctly inferring the opponent's type is crucial for the ego-agent to maximize its reward.

3.2.2 Belief Space Reward

In partially observable domains, the belief-space value function is used instead of the state value function for decision-making, which is defined as the expected cumulative

reward with respect to the state-action distribution under the belief space policy π,

$$V^{\pi}(b_0) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{s^t \sim p(s^t), a^t \sim \pi(b^t)}[R(s^t, a^t)], \tag{3.1}$$

where $p(s^t)$ is the state distribution and b^t is the belief over the state. If the belief is unbiased, then $p(s^t) = b(s^t)$ and Eq. 3.1 reduces to

$$V^{\pi}(b^0) = \sum_{t=0}^{\infty} \gamma^t r(b^t, a^t), \tag{3.2}$$

where $r(b, a) = \mathbb{E}_{s \sim b(s)}[r(s, a)]$ is the belief-space reward. In model-free reinforcement learning (RL), reward is sampled from the environment at each step. The belief-space reward sample $r(b^t, a^t)$ has lower variance than the actual reward sample $r(s^t, a^t)$ because the uncertainty associated with the state distribution has been analytically marginalized out. This low variance is beneficial for RL algorithms. In general, however, the state distribution p^t and the belief b^t could be different, for example, when the environment model $\mathcal{P}$ used for belief update is biased. In this case, the policy maximizing the belief space cumulative reward Eq. 3.2 does not necessarily maximize the actual cumulative reward Eq. 3.1. This mismatch is inevitable in multiagent scenarios with uncertain opponent types, because it is impossible to perfectly model an opponent, which makes these type of problems challenging. Therefore, developing an accurate opponent type inference scheme is crucial for learning a good belief-space policy.

3.2.3 Related Work

Our work is at the intersection of hidden-information/(hidden-role) games, robust MARL, and adversarial attack. The DeepRole algorithm [49] is the first deep MARL approach for hidden role games, to the best of the authors' knowledge. It combines counterfactual regret minimization (CFR) with deep value networks trained through self-play and integrates deductive reasoning into the RL module to reason about joint beliefs and deduce partially observable actions. Our work is similar to [49]

in terms of opponent modeling: both works learn deep policy through self-play and explicitly infer opponent type using the learned policy. However, [49] did not explicitly consider the robustness of the learned policy, while we demonstrated in our example that robustness is a critical issue in hidden role games, and proposed solutions to effectively improve the robustness through ensemble training.

Ensemble training techniques for robust MARL have been developed/applied in [5], [6, 7, 9, 50]. Among these works, [9] is the only one that actively optimizes the ensemble within the population-based-training (PBT) [50] framework. In [9], the main exploiters trained against their main agents and the league exploiters trained against all past players play a similar role as the ensemble evaluation procedure does in our work. That architecture has achieved considerable improvements in the robustness, leading to superhuman level performance in StarCraft II, but the associated significant increase in complexity makes this technique impractical for most implementations. In contrast, our ensemble optimization scheme requires much less computation.

Adversarial attacks against deep neural network and countermeasures have also been widely studied, e.g., in [51–54]. The works on adversarial attacks in RL mostly focus on different types of problems where the adversary can manipulate the reward [55], policy [56], observation [57] or environment [58] of the ego-agent. Our work is different from these bodies of work, in that the adversary is unable to directly manipulate the reward, policy or environment. Our work is based on a similar assumption as in [59]: the adversary cannot directly manipulate the protagonist's observation but can carefully choose an adversarial policy to act in the multi-agent environment to create natural adversarial observations. In [59], the authors showed that in a humanoid robot domain, the adversary can learn policies that reliably win against the ego-agent but generate seemingly random and uncoordinated behavior (feature-level attack), which induce substantially different activations in the ego-agent's policy network than when the ego-agent plays against a normal opponent. Also, these adversarial policies are more successful in high-dimensional environments. In contrast, our work focuses on scenarios where the adversarial attack is on the strategic level (adversary type hidden, carefully chooses action from low-dimensional discrete ac-

tion space that maneuvers the ego-agent's belief). In addition, the main focus of our work is on developing an inference scheme on the hidden type of the opponent and a robust policy learning algorithm in MARL, while [59] focuses on investigating the possibility of learning an adversarial policy against a fixed victim, which is essentially a single-agent problem.

The scope of this work is also closely related with multiagent reasoning and goal recognition, where the standard assumption is the availability of a library of action models (one action model is analogous to one single policy of a certain opponent type in our work), as in [46, 47]. Therefore, our MDP-S baseline corresponds to goal recognition with a library of MDP action models, and the GT-S baseline corresponds to goal recognition with a library of game-theoretic action models, which requires a game-solver for bayesian games. This requirement is non-trivial with planning-based approaches. As a result, planning-based multiagent reasoning was mostly studied in very restricted domains (e.g., matrix games as in [60]; two-stage games as [19]), while our approach is more scalable. Besides, we demonstrated that game-theoretic modeling alone is insufficient for learning robust policy, and our results show that the ensemble training is critical for improving robustness of policy against adversarial exploitation, which has rarely been explored in planning-based multiagent reasoning works.

3.3 Approach

We first give an overview of our approach. We use MARL to derive a game-theoretic model of the opponent, where the ego-agent and the opponent agent are trained against each other. Note that this learned opponent model is an 'imagined opponent', which is different than the unseen opponent during the testing. This opponent model is only used to update the ego-agent's belief about the type of the opponent during the testing. We use neural network to represent a belief space policy for the ego-agent. The belief state is updated via Bayes' rule using the learned opponent model. The opponent model learning process consists of an ensemble policy training step

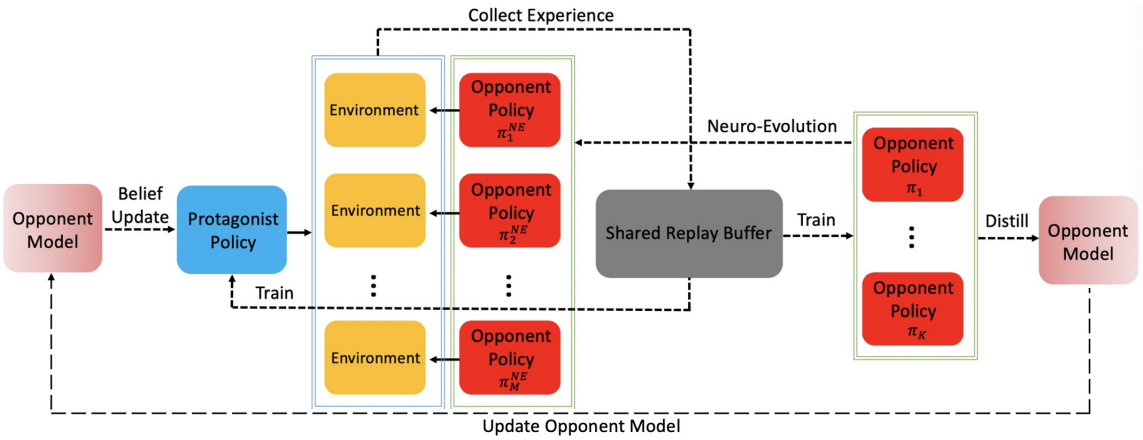


Figure 3-1: Illustration of the workflow: we train one ego-agent/protagonist policy that uses an internal opponent model for belief update. The opponent model is learned by distilling an ensemble of opponent policies trained against the ego-agent/protagonist policy. Both the ego-agent and the opponent improve their policies concurrently in the training environment

and a policy distillation step. We apply a neuro-evolutionary method to improve the diversity of the ensemble population to avoid over-fitting to a single opponent model, which improves the robustness of the opponent modeling. The above steps are illustrated in Fig. 3-1. We present the detail of each step in the following sections.

3.3.1 MARL with Ensemble Training

To improve the policy robustness of the ego-agent, we formulate its RL objective as the average cumulative reward against an ensemble of opponent policies of size K, as in [5],

$$J(\pi_i) = \mathbb{E}_{\substack{k\sim\text{unif}(1,K),\\ a_i\sim\pi_i,\\ a_{-i}\sim\pi_{-i}^{(k)}}}\left[\sum_{t=0}^{\infty}\gamma^t r_i(s,\boldsymbol{a})\right], \tag{3.3}$$

where $\text{unif}(1,K)$ denotes the uniform distribution. The policy ensemble $\{\pi_{-i}^{(k)}, k = 1,2,\ldots,K\}$ is also learned from training the opponent agent against the ego-agent's policy. Via this concurrent learning, both the ego-agent and its opponent improve their policies. Nonetheless, there is no explicit mechanism to enforce distinction among the policies within the ensemble. As a result, there could be policies that are very similar to the others. To address this redundancy issue, we apply the cooperative

evolutionary reinforcement learning (CERL) approach [61]. The key idea is to use different hyper-parameter settings for each opponent policy, use an off-policy learning algorithm and a shared experience replay buffer to improve the sample efficiency for efficient training.

3.3.2 Belief Space Policy and Belief Update

We use the belief space approach for agent policy learning. Agents explicitly maintain a belief over the hidden states (including the uncertain opponent types), and learns a belief-space policy that maps belief to action. We parameterize this mapping using a multi-layer perceptron (MLP). Instead of Eq. 3.3, the learning objective becomes,

$$J(\pi_i) = \mathbb{E}_{\substack{k\sim\text{unif}(1,K),\\ a_i\sim\pi_i(b_i),\\ a_{-i}\sim\pi_{-i}^{(k)}}} \left[\sum_{t=0}^{\infty} \gamma^t r_i(b_i, \boldsymbol{a}) \right]. \tag{3.4}$$

A belief update mechanism is required to fully specify the agent policy. The belief is the posterior distribution over the hidden states given action and observation history, $b_i^t = p(s^t, h^t | o_i^{0:t})$. The intuition behind this hidden state inference is: the observation is affected by the joint action, which depends on the joint policy as well as the hidden type; the joint policy also depends on some hidden state such as agent type. Therefore, reasoning about the hidden agent type via modeling the agent policy is possible.

We present the belief update rule for hidden type inference, starting from introducing our key assumptions.

Assumption 1 (Objective observation)**.** *Agents' observations are conditionally independent of their internal type states, given the physical state and joint action.*

Assumption 2 (Independent decision-maker)**.** *Each agent i makes its own decision conditioned on its own type variable h_i^t, and its immediate observation o_i^t.*

Assumption 3 (Time-invariant agent type)**.** *Within each episode, the agent types are sampled at the beginning of this episode and do not change over time.*

Based on these assumptions, we derive the belief update scheme beginning from the Bayes' rule,

$$b_i^t \propto p(o_i^t|s^t, h^t, o_i^{0:t-1})p(s^t, h^t|o_i^{0:t-1}) \tag{3.5}$$

the first term of which can be written as

$$\begin{aligned} p(o_i^t|s^t, h^t, o_i^{0:t-1}) &= p(o_i^t|s^t, h^t) \\ &= \int p(o_i^t|\boldsymbol{a}^t, s^t, h^t)p(\boldsymbol{a}^t|s^t, h^t)d\boldsymbol{a}^t \end{aligned} \tag{3.6}$$

where the first term, $p(o_i^t|\boldsymbol{a}^t, s^t, h^t)$, is the observation probability. It is reasonable to assume $p(o_i^t|\boldsymbol{a}^t, s^t, h^t) = p(o_i^t|\boldsymbol{a}^t, s^t) = \mathcal{P}^O(o_i^t|\boldsymbol{a}^t, s^t)$, i.e., agents' observations are independent from their internal type states (see Assumption 1). The second term in Eq. (3.6) $p(\boldsymbol{a}^t|s^t, h^t)$ is the key connection between opponent type inference and opponent policy modeling. Intuitively, this term is closely related to agent policy, we introduce the joint observation immediately before all the agents taking actions, denoted as $\boldsymbol{o}^{t^-}$, and rewrite $p(\boldsymbol{a}^t|s^t, h^t)$ as follows,

$$p(\boldsymbol{a}^t|s^t, h^t) = \int p(\boldsymbol{a}^t|\boldsymbol{o}^{t^-}, s^t, h^t)p(\boldsymbol{o}^{t^-}|s^t, h^t)d\boldsymbol{o}^{t^-}. \tag{3.7}$$

The second term $p(\boldsymbol{o}^{t^-}|s^t, h^t)$ is the observation probability $\mathcal{P}^O(\boldsymbol{o}^{t^-}|s^t)$. This probability is not conditioned on the immediate joint actions, because the joint actions have not been taken yet. The first term $p(\boldsymbol{a}^t|\boldsymbol{o}^{t^-}, s^t, h^t)$ is related to the joint policies. In order to reveal this connection, we invoke Assumption 2. Based on this assumption, we have the following factorization,

$$p(\boldsymbol{a}^t|\boldsymbol{o}^{t^-}, s^t, h^t) = p(\boldsymbol{a}^t|\boldsymbol{o}^{t^-}, h^t) \approx \prod_j^N \pi_j(o_j^t|h_j). \tag{3.8}$$

To summarize, Eq. (3.6) can be represented as:

$$p(o_i^t|s^t, h^t, o_i^{0:t-1}) = \mathbb{E}_{\boldsymbol{a}^t \sim \boldsymbol{\pi}(\bar{\boldsymbol{o}}|\boldsymbol{h})}[\mathcal{P}^O(o_i^t|\boldsymbol{a}^t, s^t)], \tag{3.9}$$

where $\bar{\boldsymbol{o}} = \int \mathcal{P}^O(\boldsymbol{o}^{t^-}|s^t)d\boldsymbol{o}^{t^-}$. The interpretation of Eq. (3.9) is that the probability of receiving an observation o_i^t is the expected observation by marginalizing out all the possible joint actions over the observation probability $\mathcal{P}^O(o_i^t|\boldsymbol{a}^t, s^t)$, where the probability of the joint actions $\boldsymbol{\pi}(\bar{\boldsymbol{o}}|\boldsymbol{h})$ is obtained from the joint policies using the expected joint observation of all the agents.

The second term in Eq. (3.5), $p(s^t, h^t|o_i^{0:t-1})$ can be expressed as

$$\int p(s^t, h^t|s^{t-1}, h^{t-1})p(s^{t-1}, h^{t-1}|o_i^{0:t-1})ds^{t-1}dh^{t-1} \tag{3.10}$$

$$= \int p(s^t, h^t|s^{t-1}, h^{t-1})b_i^{t-1}ds^{t-1}dh^{t-1}. \tag{3.11}$$

To further simplify this expression, we invoke Assumption 3 so that

$$p(s^t, h^t|s^{t-1}, h^{t-1}) = p(s^t, h^t|s^{t-1}, h^{t-1})\delta(h^t|h^{t-1}),$$

where $\delta(h^t|h^{t-1})$ denotes the Dirac-delta measure. With this assumption, Eq. (3.11) simplifies to

$$p(s^t, h^t|o_i^{0:t-1}) = \int p(s^t|s^{t-1}, h^t)b_i^{t-1}ds^{t-1}. \tag{3.12}$$

Combining Eqs. (3.5), (3.9), and (3.12) yields the recursive belief update rule,

$$b_i^t \propto \mathbb{E}_{\boldsymbol{a}^t\sim\boldsymbol{\pi}(\bar{\boldsymbol{o}}|\boldsymbol{h})}[\mathcal{P}^O(o_i^t|\boldsymbol{a}^t, s^t)] \int p(s^t|s^{t-1}, h^t)b_i^{t-1}ds^{t-1}, \tag{3.13}$$

which has the following interpretation: To infer the state of current step, we can predict it based on the posterior belief of the last step, by propagating the physical state distribution and correcting the belief over the hidden type variable via comparing the actual observation with the anticipated observation according to agent policy modeling.

Remark 1. *In the belief update rule, the inference over the hidden type variable is implicit inside the expectation term. The observation probability is crucial to the discriminative power of this inference. To illustrate this point, consider an extreme case where the observation contains no information about agents' actions, i.e.,* $\mathcal{P}^O(o_i^t|\boldsymbol{a}^t, s^t)$ *is*

not a function of $\boldsymbol{a}^t$. In this case, this expectation term will be independent of the joint policy (will be a constant due to normalization condition of expectation). As a result, no information about the hidden type variable can be extracted from this term. This makes sense, because if the observation tells us nothing about the actions taken by the other agents (dictated by their policies and hidden types), then it is impossible to update our belief over their hidden types. Conversely, if the observation contains full information about the joint action (e.g., the ego-directly observes the joint action), this expectation term would be highly dependent on the joint policies (therefore, on the hidden type variable), and the discriminative power of this inference scheme is maximized.

The observation probability $\mathcal{P}^O(o_i^t|\boldsymbol{a}^t, s^t)$, the agent policies $\boldsymbol{\pi}$, and the state transition probability $p(s^t|s^{t-1}, h^{t-1})$ are required to implement the belief update, which is anticipated. This work focuses on a special case in which the physical states are fully observable to all the agents, so agents do not need to maintain a belief over s^t. This assumption simplifies the computational aspect of the problem, but it does not diminish the central difficulty of the problem, i.e., inferring the hidden type of opponent.

To approximate the policies of agent j of each possible type $\{h_j^{(m)}\}_{m=1}^M$, recall that, in the ensemble training step, we create K different policies $\{\pi_{j,m}^{(k)}\}_{k=1}^K$ for each agent of each type. Here we use shorthand $\pi_{j,m}$ to denote agent j with type $h_j^{(m)}$. Each policy within one ensemble can be interpreted as one of the likely strategies that could be adopted by agent j with type $h_j^{(m)}$. However, in the belief update equation, we need only one single policy for agent j with type $h_j^{(m)}$. As a result, we must synthesize the policy ensemble into one representative policy that best represents the average behavior of the policy ensemble. We learn this representative policy by minimizing the information theoretic distance (Kullback−Leibler (KL) divergence) between this policy and the policy ensemble. The resulting optimization problem to learn the

representative policy $\pi^0_{j,m}$ is then

$$J(\pi^0_{j,m}) = \sum_{k=1}^{K} \mathbb{KL}(\pi^{(k)}_{j,m}, \pi^0_{j,m}), \tag{3.14}$$

which is policy distillation [62]. The solution to this minimization is,

$$\pi^0_{j,m} = \frac{1}{K} \sum_{k=1}^{K} \pi^{(k)}_{j,m}, \tag{3.15}$$

which happens to be the average over the policies within one ensemble. As a result, we can approximate $\pi^0_{j,m}$ by training a policy distillation network to minimize the residue of Eq. (3.15) using data sample, which is computationally much more efficient than calculating all the K policies during testing.

3.3.3 Policy Ensemble Optimization

The ensemble training step typically improves the robustness of the ego-agent's policy. However, two problems need to be addressed to make this approach more effective and efficient. First, we want a metric for measuring policy robustness and we want to explicitly optimize this robustness metrics. Second, we want to reduce the additional computation overhead introduced by ensemble training.

We address these two problems by training a hold-out adversary agent to exploit the ego-agent's policy, and use the resulted ego-agent's reward as a robustness score. Instead of using a fixed-size ensemble, we dynamically resize the ensemble through three operations: **pop**, **append**, and **exchange**. **pop** randomly removes one policy from the ensemble and pushes it into a deactivation-set. **append** randomly selects one policy from the deactivation-set and adds it to the ensemble. **exchange** randomly selects one policy from both the ensemble and the deactivation-set and exchanges them.

The objective of modifying the ensemble is to obtain a good trade-off between robustness and computational complexity, which is dominated by the ensemble size.

Procedure 1 Ensemble evaluation

1: Fix the ego-agent's policy
2: Train a single opponent policy against the fixed ego-agent's policy
3: Obtain the average ego-agent reward r^p and opponent reward r^o after training

Procedure 2 Ensemble Optimization

1: Randomly select an operation ξ from {**pop**, **append**, **exchange**} to apply on the policy ensemble
2: Obtain a new metrics ρ_{new} via Procedure 1
3: Accept the operation ξ with probability p, where
4: $p = \exp(\min\{0, \rho_{\textbf{old}} - \rho_{\textbf{new}}\}/T)$

We propose to measure the robustness via Procedure 1, and we define the following metric (with weights λ_i)

$$\rho = -r^p + \lambda_1 r^o + \lambda_2 K, \tag{3.16}$$

where K is the varying size of the policy ensemble. The combined reward term $-r^p + \lambda_1 r^o$ is a measure of the robustness of the ego-agent's policy, which is noisy due to the intrinsic stochasticity of RL, while K is a surrogate measure for computation complexity. Minimizing ρ leads to a trade-off between policy robustness and computation complexity. We interpret this minimization problem as a stochastic optimization over the powerset of the initial policy ensemble. We solve this stochastic optimization via simulated annealing (Procedure 2).

Table 3.1: Hyper-parameter of ensemble optimization

Hyper-parameter	Value
Opponent loss weight λ_1	0.1
Ensemble size weight λ_2	1.0
Initial temperature T_0	30.0
Minimum temperature T_{min}	0.2
Temperature decay rate	0.975

3.4 Experiments

This section addresses the following questions:

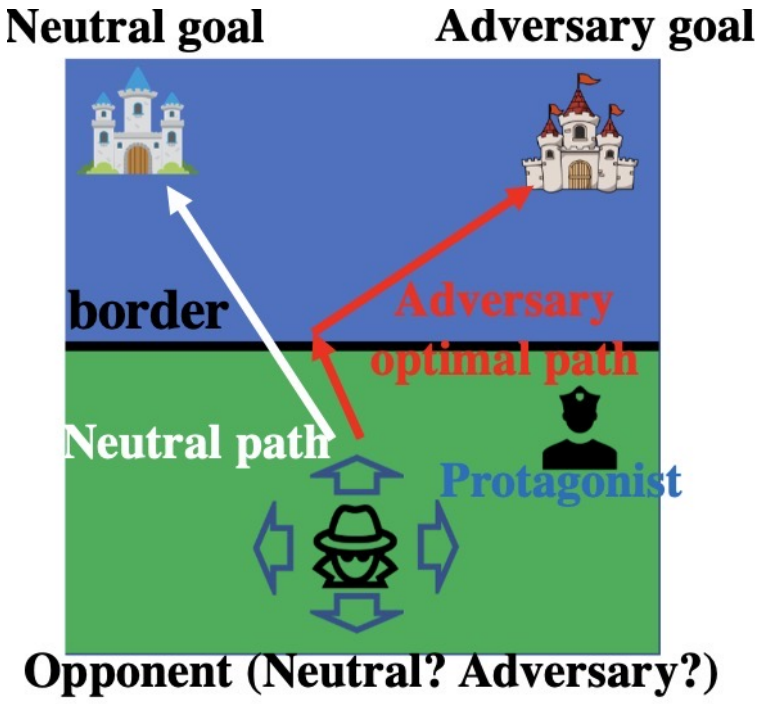


Figure 3-2: Sketch of the scenario, the opponent (bottom middle) could be an adversary or a neutral agent. The ego-agent/protagonist agent (upper right corner in the green region) must infer the identity of the opponent and tag the adversary and let the neural agent pass.

1. Is it necessary to use ensemble training, considering its additional computation overhead?
2. Is it beneficial to explicitly model opponent policy and maintain a belief?
3. How much improvement do we get from ensemble training?

3.4.1 Scenario: Urban-security Game

We design a two-player urban-security game with uncertain opponent types to evaluate our algorithm, as illustrated in Fig. 3-2. There are two agents: the ego-agent/protagonist agent (the officer) and the opponent agent with two possible types (either a neutral agent or an adversary). The opponent's objective is to reach its home base (a neutral opponent goes to the ally's base, the blue castle, while an adversarial opponent goes to the enemy's base, the red castle). The protagonist's objective is to identify the type of its opponent and obtain reward by tagging the adversarial oppo-

nent and let pass the neutral opponent. Mistakenly tagging a neutral would incur a large penalty to the protagonist. The protagonist cannot enter the blue region of the map, so once the opponent has passed the green region and entered the blue region, the protagonist cannot tag it anymore. If the opponent is an adversary, it receives a large penalty if tagged. The opponent always receives penalty if it has not reached its base, and the penalty increases with its distance from its base. Based on the game rules, an adversarial opponent could try multiple strategies. For example, one strategy is to rush towards its home base to minimize the distance penalty. However, if it takes this greedy strategy, the protagonist can quickly identify this adversary and try to tag it (large penalty for the adversary). Another strategy is to initially head towards the ally base, to trick the protagonist agent into believing that the opponent is a neutral opponent. Once the adversary is close enough to the blue region, it can safely head to its base. This strategy incurs a larger distance penalty but might get a higher reward by avoiding being tagged.

State and Action Space

The state of each agent is its 2-d position, i.e., $\mathcal{S}_i = [0, 8] \times [0, 8]$. The protagonist agent has a discrete action space

$\mathcal{A}^p = [\textbf{move left}, \textbf{move right}, \textbf{move up}, \textbf{move down}, \textbf{tag}, \textbf{probe}]$.

The opponent agent's action space is

$\mathcal{A}^o = [\textbf{move left}, \textbf{move right}, \textbf{move up}, \textbf{move down}]$.

Each of the 'move' action changes the agent position by one unit distance. The tag action succeeds if and only if the distance between the two agents is less than 2.5. The probe action is equivalent to query a noisy measurement of the opponent's true type, where there is 0.8 probability getting the correct type and 0.2 probability getting the wrong type. The protagonist agent could take this probe action to help with its inference besides simply observing the opponent. Each probe action incurs cost, so the protagonist agent has to wisely decide when and how many times to probe.

Reward

The reward of the opponent agent consists of two parts: (1) $r_d = -0.25d^{2/5}$, where d is its distance from its home base; (2) $r_{\text{tagged}} = -10$ if being tagged.

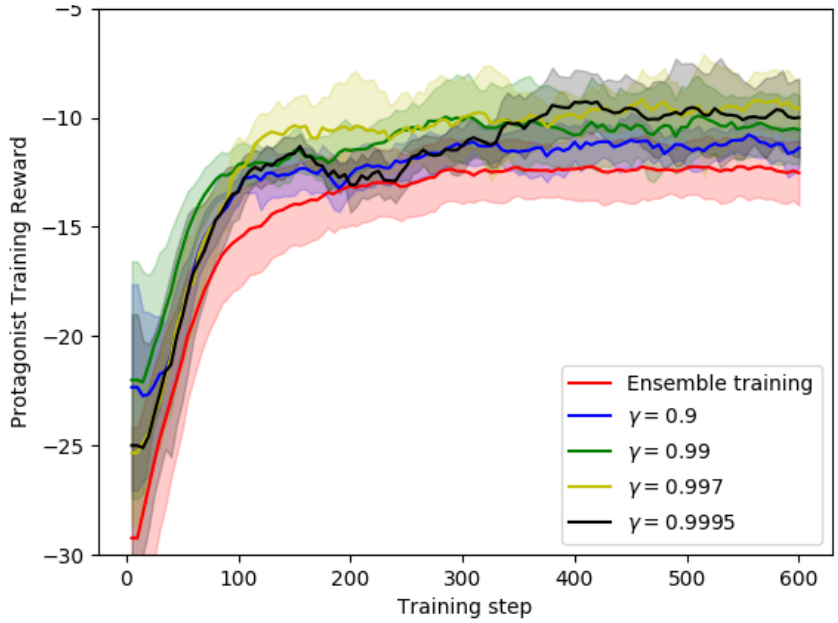
The state-action reward of the protagonist agent consists of several parts: (1) $r_{\text{tag adversary}} = 10$ if tagging an adversarial agent; (2) $r_{\text{tag neutral}} = -20$ if tagging a neutral agent; (3) $r_{\text{d2o}} = -0.25d_{\text{o}}^{2/5}$, where d_{o} is the distance between the protagonist agent and the opponent. This is a heuristic reward to help the protagonist agent learning sensible behaviors; (4) Tag cost $r_{\text{tag cost}} = -0.2$, no matter tagging is successful or not; (5) Probe cost $r_{\text{probe cost}} = -0.25C$, where C is the cumulative counts of the probing action so far, i.e., the probe cost per time increases as the total number of probing increases. This effectively prevents the agent from abusing the probe action.

3.4.2 Ensemble Training vs. Single Model

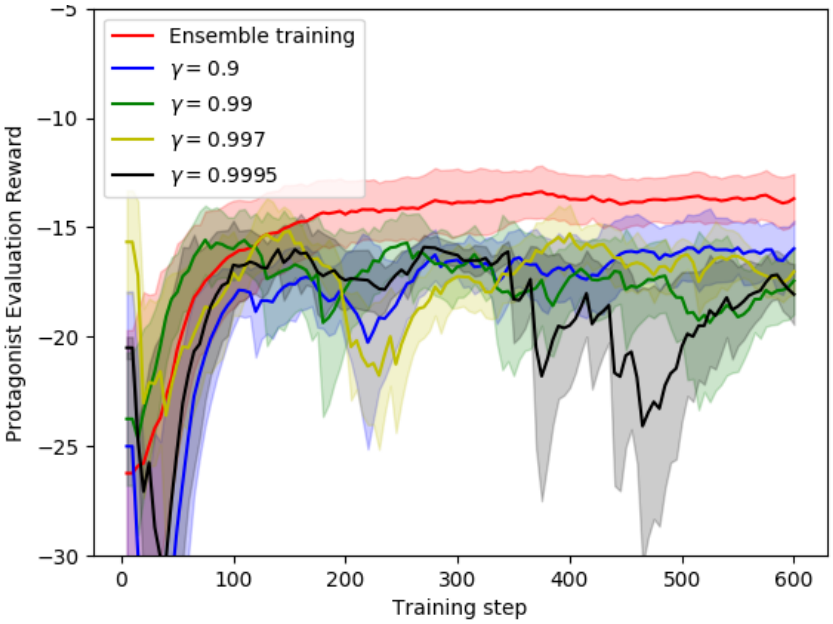
To answer the first question, we compared the protagonist policy learned from training against an ensemble of opponent policies and that from training against a single opponent policy. We used a similar ensemble as used in [61], which consists of four opponent policies, each policy is learned from training against the protagonist policy. We used four different discount factors for the opponent learning objectives: $\gamma_1 = 0.9, \gamma_2 = 0.99, \gamma_3 = 0.997, \gamma_4 = 0.9995$. An interpretation of this setting is a variety of opponent playing styles ranging from myopic to far-sighted strategies.

For comparison, we also trained the protagonist policy individually against each opponent model, so we obtained five protagonist policies in total. For evaluation, we trained five separate opponent evaluation policies, each corresponding to one of the protagonist policies. The evaluation policies all used the same discount factor $\gamma = 0.99$.

Fig. 3-3a and Fig. 3-3b (each curve is averaged over three runs) show the training and evaluation rewards. During training, the single model policies generally lead to higher protagonist reward, while the ensemble training results in the lowest protag-



(a) Protagonist training reward



(b) Protagonist evaluation reward

Figure 3-3: Training and evaluation rewards of the protagonist agent: Single opponent models ($\gamma = 0.9$, $\gamma = 0.99$, $\gamma = 0.997$, $\gamma = 0.9995$) performs better than ensemble training in the training phase due to overfitting to simple opponent models, while ensemble training outperforms single opponent models in evaluation

Table 3.2: Mean evaluation reward: vs. LSTM

Algorithm*	Protagonist	Adversary
belief space, with EO & CE	**-14.4±1.49**	**-83.0±17.0**
LSTM, with EO & CE	-17.7±1.9	-66.2±13.8
belief space, w/o EO & CE	-16.5±1.1	-58.6±24.9
LSTM, w/o EO & CE	-16.8±3.1	-49.4±6.6

* EO: ensemble optimization, CE: cooperative evolution.

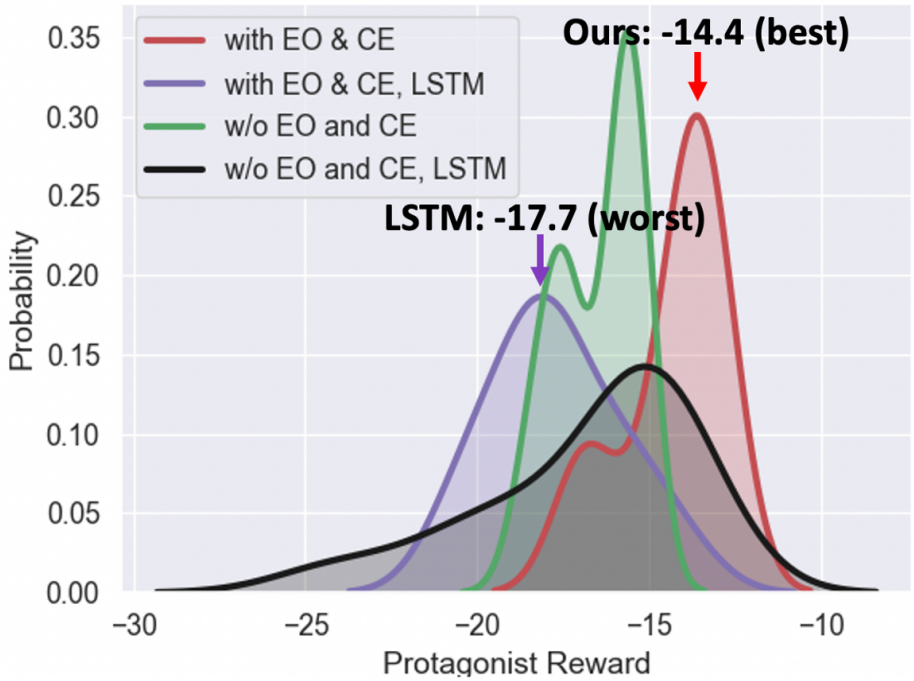
onist reward. This suggests that the protagonist policy overfits to one of the single opponent models, thus achieving high training reward but low evaluation reward. In contrast, the protagonist policy trained against the ensemble achieves the best evaluation reward. It is worth pointing out that, in the second single model setting, although the hyper-parameter $\gamma_2 = 0.99$ is the same as that of the evaluation opponent, the evaluation reward is still significantly worse than the training reward. This is not surprising, as the agent could learn different policies even with the same hyper-parameter setting. Therefore, overfitting is almost inevitable when training against a single model.

3.4.3 Belief Space Policy vs. Implicit Approach via RNN

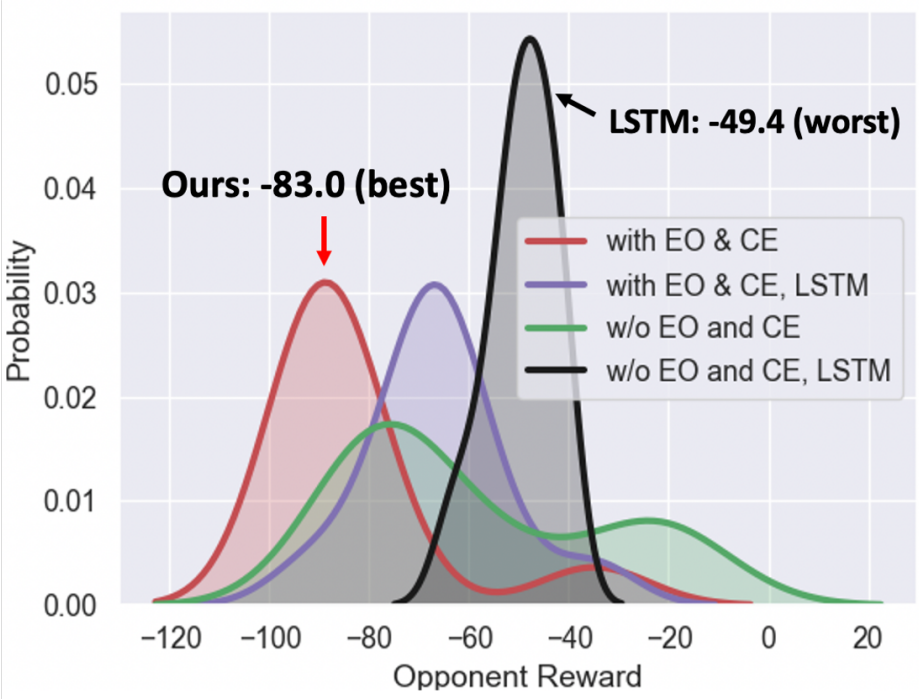
To answer the second question, we replaced belief space policy with a recurrent policy parameterized by a Long short term memory network (LSTM). Fig. 3-4 (histogram of rewards from 10 runs) and Table 3.2 (mean rewards) show the comparison between these two settings, where the belief space policy consistently outperforms the recurrent policy. This result agrees with our conjecture that learning a recurrent policy is difficult due to the lack of prior knowledge on the information structure and the high-variance of the state-space reward.

3.4.4 Ablation Study

To answer the third question, we compared our algorithm with its ablated versions: (I) without neuro-evolution, (II) without both neuro-evolution and ensemble optimization. For the ablated version II, we randomly sampled subsets of the ensemble

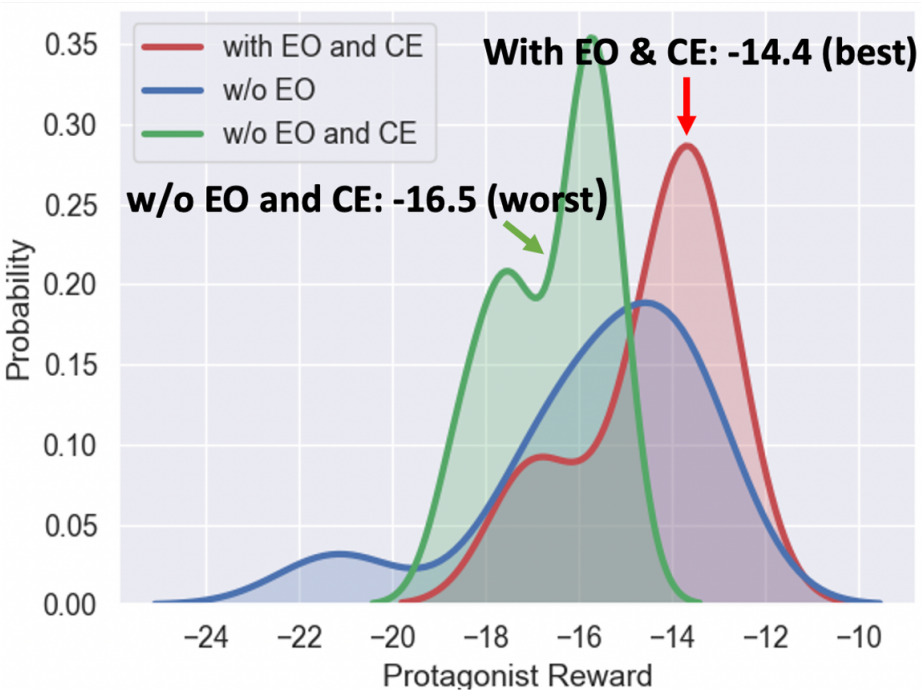


(a) Protagonist reward: higher is better

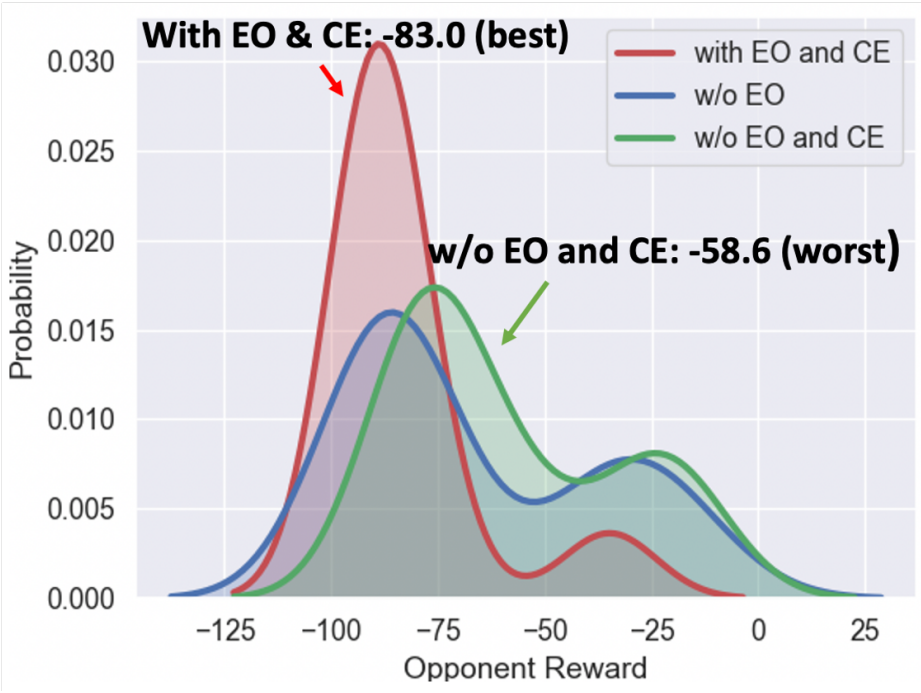


(b) Opponent (adversary) reward: lower is better

Figure 3-4: Evaluation rewards distribution of the protagonist agent (top) and the opponent agent (bottom): (1) belief space policy, with ensemble optimization (EO) and cooperative-evolution (CE); (2) LSTM, with EO and CE; (3) belief space policy, without EO and CE (single opponent model); (4) LSTM, without EO and CE (single opponent model); Belief space policy (1) and (3) outperforms LSTM (2) and (4)



(a) Protagonist reward: higher is better



(b) Opponent (adversary) reward: lower is better

Figure 3-5: Evaluation rewards of the protagonist agent (top) and the opponent agent (bottom): (1) with both ensemble optimization (EO) and cooperative-evolution (CE); (2) with CE but without EO; (3) without EO and CE (single opponent model); EO + CE outperforms CE only, which outperforms single opponent model

Table 3.3: Mean evaluation reward: ablation study

Ablated version*	Protagonist	Adversary
with EO & CE	**-14.4±1.49**	**-83.0±17.0**
w/o EO	-15.5±2.2	-65.8±28.5
w/o EO & CE	-16.5±1.1	-58.6±24.9

* EO: ensemble optimization, CE: cooperative evolution.

from its powerset and used the fixed subset for training. We ran 10 independent simulations for each of the ablated version. Fig. 3-5 (histogram of rewards from 10 runs) and Table 3.3 (mean rewards) shows the evaluation rewards of the full and ablated versions of our algorithm. This result suggests that both neuro-evolution and ensemble optimization have important contributions to the performance improvement.

3.4.5 Accuracy of Hidden Type Inference

To gain insight into the reason behind the results shown above, we present the accuracy of the opponent's hidden type inference under the different opponent modeling settings:

1. MDP-S: The baseline approach, where the opponent agent is modeled as a goal-directed MDP agent, and the optimal goal-achieving policy is used by the protagonist agent for opponent type inference.

2. GT-S: A game theoretic opponent modeling approach with a single opponent policy where the protagonist agent and the opponent concurrently update their policies during the training, and the protagonist agent uses this learned opponent model for type inference during the testing.

3. GT-E: A game theoretic opponent modeling approach with an ensemble of opponent policies, and the protagonist agent uses the distilled policy from this ensemble for type inference during the testing.

We randomly sampled the opponent type with an equal probability (50%) of neutral and adversary. Table 3.4 shows the average true positive rate (TPR, successfully iden-

Table 3.4: Opponent type inference accuracy / Rewards

Metrics*/Approach	MDP-S	GT-S	GT-E
TPR10	0.12	0.34	**0.68**
TPR20	0.06	0.26	**0.74**
TNR10	0.91	0.73	0.85
TNR20	0.96	0.82	0.91
Precision10	0.57	0.56	**0.82**
Precision20	0.60	0.59	**0.89**
Recall10	0.12	0.34	**0.68**
Recall20	0.06	0.26	**0.74**
Protagonist reward	-19.44	-17.55	**-14.4**
Adversary reward	-50.48	-58.19	**-83.0**

* TRPn: TPR after n time steps, similarly for the other metrics.

tify an adversary), true negative rate (TNR, successfully identify a neutral), precision, recall, and mean rewards for both agents.

The TPR corresponding to MDP-S is quite low, indicating that the protagonist agent completely mis-classifies the adversary agent as a neutral agent. As the time step increases (comparison between TPR10 and TPR20), this mis-classification becomes worse (from 0.12 to 0.06). This result indicates that there is a significant mismatch between the protagonist's model of the adversary policy and the actual adversary policy during the testing. Besides, the true negative rate is quite high. This result suggests that the adversary indeed learned a policy that confuses the protagonist agent by mimicking the behavior of a neutral agent, and therefore, the protagonist agent always classifies the opponent as a neutral agent. Similarly, the TPR corresponds to GT-S is also significantly lower than the prior probability (50%), which indicates a mismatch between the opponent model and the actual opponent, while GT-S outperforms MDP-S by a large margin since the game-theoretic aspect accounts for the strategic reasoning between the adversary and the protagonist agent which is missing in the MDP-S approach. In contrast, the TPR corresponding to GT-E significantly outperforms the other two approaches. Moreover, as the time step increases, the TPR also increases from 0.68 to 0.74, which indicates that this opponent model successfully captures an adversary's general behavior pattern and is able to generalize

to previously unseen adversaries.

The TNR statistics of these three modeling approaches are all significantly higher than 50%, which indicates a good success rate of identifying a neutral agent. This result is also anticipated due to the fact that the neutral agent's policy is simple (heading towards its goal), so it can be accurately learned by the protagonist agent. Although the TNR of MDP-S is highest among these three modeling approaches, this high TNR does not necessarily indicate MDP-S is good at identifying the neutral agent, but rather always 'guessing' the opponent as being neutral. As a result, the recall scores of the MDP-S and the GT-S approaches are much lower than that of the GT-E approach, and consequently lead to a poorer protagonist agent performance as indicated by the rewards.

3.5 Summary

We summarize the key findings of this work as follows:

- We present algorithms based on MARL and ensemble training for robust opponent modeling and posterior inference over the opponent type from the observed action.
- We demonstrate that the explicit opponent modeling outperforms a black-box RNN approach, and the ensemble training approach outperforms a single agent model. We analyze the reason for this observation by inspecting the agent type inference, and show that the performance of the ego-agent policy (protagonist) is highly correlated to the quality of the type inference accuracy.

Chapter 4

Scalable Multiagent Joint Policy Learning with Sparse-attentional Graph Neural Network

4.1 Introduction

Reinforcement Learning (RL) has achieved enormous successes in robotics [63] and gaming [64] in both single and multiagent settings. For example, deep reinforcement learning (DRL) achieved super-human performance in the two-player game Go, which has a very high-dimensional state-action space [65, 66]. However, in multiagent scenarios, the sizes of the state space, joint action space, and joint observation space grow exponentially with the number of agents. As a result of this high dimensionality, existing multiagent reinforcement learning (MARL) algorithms require significant computational resources to learn an optimal policy, which impedes the application of MARL to systems such as swarm robotics [67]. Thus, improving the scalability of MARL is a necessary step towards building large-scale multiagent learning systems for real-world applications.

In MARL, the increase of complexity of finding an optimal joint policy, with respect to the number of agents, is a result of coupled interactions between agents

[68]. However, in many multiagent scenarios, the interactions between agents are quite sparse. For example, in a soccer game, an agent typically only needs to pay attention to other nearby agents when dribbling because agents far away are not able to intercept. The existence of such sparsity structures of the state transition dynamics (or the state-action-reward relationships) suggests that an agent may only need to attend to information from a small subset of the agents for near-optimal decision-making. Note that the other players that require attention might not be nearby, such as the receiver of a long pass in soccer. In such cases, the agent only needs to selectively attend to agents that "matter the most". As a result, the agent can spatially and temporally reduce the scale of the planning problem.

In large-scale MARL, sample complexity is a bottleneck of scalability [69]. To reduce the sample complexity, another feature we can exploit is the interchangeability of homogeneous agents: switching two agents' state/action will not make any difference to the environment. This interchangeability implies permutation-invariance of the multiagent state-action value function (a.k.a. the centralized Q-function) as well as interchangeability of agent policies. However, many MARL algorithms such as MADDPG [70], VDN [71], QMIX [72] do not exploit this symmetry and thus have to learn this interchangeability from experience, which increases the sample complexity unnecessarily.

Graph neural network (GNN) is a specific neural network architecture in which permutation-invariance features can be embedded via graph pooling operations, so this approach has been applied in MARL [73–75] to exploit the interchangeability. As MARL is a non-structural scenario where the links/connections between the nodes/agents are ambiguous to decide, a graph has to be created in advance to apply GNN for MARL. Refs. [73–75], apply ad-hoc methods, such as k-nearest neighbors, hard threshold, and random dropout to obtain a graph structure. However, these methods require handcrafted metrics to measure the closeness between agents, which are scenario-specific and thus not general/principled. Inappropriately selecting neighbors based on a poorly designed closeness metric could lead to the failure of learning a useful policy.

While attention mechanisms [76] could be applied to learn the strength of the connections between a pair of agents (i.e., closeness metric) in a general and principled way, such strengths are often dense, leading to a nearly-complete computation graph that does not benefit scalability. The dense attention mechanism results from that the softmax activation function operated on the raw attention logits generates a probability distribution with full support. One solution to enforce a sparse graph is top k thresholding [77], which keeps the k-largest attention scores and truncates the rest to zero. However, this truncation is a non-differentiable operation that may cause problems for gradient-based optimization algorithms, such as those used in end-to-end training. Therefore, a sparse attention mechanism that preserves the gradient flow necessary for gradient-based training is required.

To address the non-differentiability issue in sparse attention mechanisms, we generalize sparsemax [78] and obtain a sparsity mechanism whose pattern is adaptive to the environment states. This sparsity mechanism can reduce the complexity of both the forward pass and the back-propagation of the policy and value networks, as well as preserving the end-to-end trainability in contrast to hard thresholding. With the introduction of GNN and generalized sparsemax, which can preserve permutation invariance and promote sparsity respectively, the scalability of MARL is improved.

The discussion so far was restricted to homogeneous agents and thus permutation-invariance is desirable. However, in heterogeneous multiagent systems or competitive environments, permutation invariance and interchangeability are no longer valid. For example, in soccer, switching positions of two players from different sides can make a difference to the game. To address this heterogeneity, GNN-based MARL must distinguish the different semantic meanings of the connections between different agent pairs (e.g. friend/friend relationship versus friend/foe relationship). We address this requirement by multi-relational graph convolution network [79] to pass messages using different graph convolution layers on graph edge connections with different semantic meanings.

To summarize, we propose to learn an adaptive sparse communication graph within the GNN-based framework to improve the scalability of MARL, which applies

to both homogeneous and heterogeneous multiagent systems in mixed cooperative-competitive scenarios.

4.2 Background

4.2.1 Multi-head attention

The scaled dot-product attention mechanism was first proposed in [76] for natural language processing. An attention function maps the query and a set of key-value pairs to the output, which is the weighted sum of the values. The weight assigned to the each value calculated via a compatibility function of the query and the corresponding key. In the context of MARL, let $h_i, i \in N$ be the representation of the agents. Key, query and value of agent i is defined as $K_i^l = W_K h_i^l \in \mathbb{R}^{d_K}$, $Q_i^l = W_Q h_i^l$ and $V_i^l = W_V h_i^l$, respectively with W_K, W_Q and W_V are parameter matrices. The output for agent i is then

$$\mathrm{Att}_i(h) = \sum_j w_{ij} V_j, \tag{4.1}$$

where $w_{i\bullet} \in \mathbb{R}^n$, the i-th row of the weight matrix w, is defined as

$$w_{i\bullet} = \sigma_a\Big(\frac{(K_i)^T Q}{\sqrt{d_K}}\Big) \tag{4.2}$$

with σ_a being the softmax function in previous works of GNN-based MARL. The weight $w_{i\bullet}$ is dense as $\mathrm{softmax}_i(z) \neq 0$ for any vector z and i.

To increase the expressiveness, multi-head attention is applied here via simply concatenating the outputs from a single attention function [76].

4.2.2 Relational GNN

In heterogeneous multiagent systems, different agent pair can have different relations, such as friend or foe in a two-party zero-sum game. As a result, information aggregation from agents with different relations should have different parameters. Work in [79] proposed relational graph convolutional network to model multi-relational data.

The forward-pass update of agent i in a multi-relational graph is as follows

$$h_i^{(l+1)} = \sigma\Big(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}W_r^{(l)}h_j^{(l)} + W_0^{(l)}h_i^{(l)}\Big), \tag{4.3}$$

where $\mathcal{N}_i^r$ denotes the set of neighbor indices of agent i under relation $r \in \mathcal{R}$ and $c_{i,r}$ is a normalization constant. To distinguish the heterogeneity in MARL, similar to this convolution-based multi-relational GNN, we apply different attention heads on agent pairs with different relations.

4.3 Related Work

One of the existing works exploiting the structure in MARL is the mean-field reinforcement learning (MFRL) [80] algorithm, which takes as input the observation and the mean action of neighboring agents to make the decision, and neglects the actions of all the other agents. This simplification leads to good scalability. However, the mean action cannot distinguish the difference among neighboring agents and the locality approximations fail to capture information from a far but important agent for optimal decision-making, which leads to sub-optimal policies. Multi-Actor-Attention-Critic (MAAC) is proposed in [81] to aggregate information using attention mechanism from all the other agents. Similarly, [73, 75, 82] also employ the attention mechanism to learn a representation for the action-value function. However, the communication graphs used there are either dense or ad-hoc (k nearest neighbors), which makes the learning difficult.

Sparse attention mechanisms were first studied by the natural language processing community in [78], where sparsemax was proposed as a sparse alternative to the activation function softmax. The basic idea is to project the attention logits onto the probability simplex, which can generate zero entries once the projection hits the boundary of the simplex. While generalized sparse attention mechanisms were further studied in [83–85], they are not adaptive to the state in the context of MARL, in terms of the sparsity pattern.

Given this state of the art, the contributions of this work are twofold. First, we propose a new adaptive sparse attention mechanism in MARL to learn a sparse communication graph, which improves the scalability of MARL by lowering the sample complexity. Second, we extend our GNN-based MARL to heterogeneous systems in mixed cooperative-competitive settings using multi-relational GNN. The evaluations show that our algorithm significantly outperforms previous approaches on applications involving a large number of agents. This technique can be applied to empower large-scale autonomous systems such as swarm robotics.

4.4 Approach

In this section, we present our approach to exploit the sparsity in MARL by generalizing the dense soft-max attention to adaptive sparse attention. Moreover, our approach to apply multi-relational attention mechanism for heterogeneous games involving competitive agents is also introduced.

4.4.1 Learning a communication graph via adaptive sparse attention

The scaled dot-product attention is applied to learn the communication graph in MARL. If an attention weight between a pair of agents is zero, then there is no communication/message passing between them. Thus, the normalization function $\sigma_a(\bullet)$ in (4.2) is critical to learn a communication graph. As usually used in the attention mechanism [76] or classifications, $\sigma_a(\bullet)$ is usually set to be softmax, which cannot induce sparsity. We propose an adaptive sparse activation function as an alternative to softmax.

Let $x \in \mathbb{R}^d$ be the raw attention logits and y be normalized attention strength in the $(d-1)$-dimensional probability simplex defined as $\Delta^d := \{y \in \mathbb{R}^d | y \geq 0, \mathbf{1}^T y = 1\}$. We are interested in the mapping from $x \in \mathbb{R}^d$ to $y \in \Delta^d$. In other words, such a mapping can transform real weights to a probability distribution, i.e., the normalized

attention strength between a pair of agents. The classical softmax, used in most attention mechanisms, is defined component-wisely as

$$y_i = \underset{i}{\text{softmax}}(x) = \frac{e^{x_i}}{\sum_{i=1}^{d} e^{x_i}}. \tag{4.4}$$

A limitation of the softmax transformation is that the resulting probability distribution always has full support, which makes the communication graph dense, resulting in high complexity. In order to reduce the complexity, our idea is to replace the softmax activation function with a generalized activation function, which could adaptively be dense or sparse based on the state. To investigate alternative activation functions to softmax, consider the max operator defined as

$$\max(x) := \max_{i \in [d]}(x_i) = \sup_{y \in \Delta^d} y^T x, \tag{4.5}$$

where $[d] = \{1, \ldots, d\}$. The second equality comes from that the supremum of the linear form over a simplex is always achieved at a vertex, i.e., one of the standard basis vector $\{e_i\}_{i \in [d]}$. As a result, the max operator puts all the probability mass onto a single element, or in other words, only one entry of y is nonzero corresponding to the largest entry of x. For example, with $x = [0, t] \in \mathbb{R}^2$, the probability distribution w.r.t. the logit t, i.e., $(\arg\sup_{y \in \Delta^d} y^T x)_2$, is a step function, as $(\arg\sup_{y \in \Delta^d} y^T x)_2$ equals 1 if $t > 0$ and 0 otherwise. This discontinuity at $t = 0$ of the step function is not amenable to gradient-based optimization algorithms for training deep neural networks. One solution to the discontinuity issue encountered in (4.6) is to add a regularized $\Omega(y)$ in the max operator as

$$\Pi_\Omega(x) = \underset{y \in \Delta^d}{\arg\max}\, y^T x + \gamma \Omega(y) \tag{4.6}$$

Different regularizers $\Omega(y)$ produce different mappings with distinct properties (see summary in Table 4.1). Note that with $\Omega(y)$ as the Shannon entropy, $\Pi_\Omega(x)$ recovers softmax. With the states/observations evolving, the ideal profile of $\Pi_\Omega(x)$ should be able to adapt the sparsity extent (controlled via γ) and the pattern (controlled via

Table 4.1: List of different regularizers and their corresponding mappings $y = \Pi_\Omega(x)$, where x is the raw attention logits and y is the probability distribution in Δ^d.

Entropy	$\Omega(y)$	$\Pi_\Omega(x)$	Ref.
Shannon	$\sum_i y_i \log(y_i)$	$\text{softmax}_i(x) = \frac{e^{x_i}}{\sum_{i=1}^d e^{x_i}}$	[84]
l_2 norm	$-\frac{1}{2}\sum_i y_i^2$	$\arg\min_{y\in\Delta^d} \|y - x\|^2$	[78]
Tsallis	$\begin{cases} \frac{\sum_i (y_i - y_i^\alpha)}{\alpha(\alpha-1)}, & \alpha \neq 1 \\ \sum_i y_i \log(y_i), & \alpha = 1 \end{cases}$	No closed-form	[86]
Generalized	$\frac{1}{q}\sum_i (y_i - \frac{e^{qy_i} - 1}{e^q - 1})$	No closed-form	[87]

Table 4.2: List of different $G(x)$ and their resulting mappings $\Pi_\Omega(x)$

$\gamma G_i(x)$	$\frac{e^{x_i}}{\sum_i e^{x_i}}$	$\frac{x_i^2}{\sum_i x_i^2}$	x_i
$\Pi_\Omega(x)$	softmax	softmax	sparsemax
Property	Translation invariance $\Pi_\Omega(x) = \Pi_\Omega(x + c\mathbf{1})$	Scaling invariance $\Pi_\Omega(x) = \Pi_\Omega(cx)$	Translation invariance $\Pi_\Omega(x) = \Pi_\Omega(x + c\mathbf{1})$
Example	$\Pi_\Omega([100, 101]) = \Pi_\Omega([0, 1])$	$\Pi_\Omega([1, 2]) = \Pi_\Omega([1, 2] \times 10^{-3})$	$\Pi_\Omega([100, 101]) = \Pi_\Omega([0, 1])$

the selection of $\Omega(y)$) accordingly.

Note that the Tsallis entropy and the generalized entropy in Table 4.1 do not have closed-form solutions [84], which will increase the computational burden since iterative numerical algorithms will have to be employed. Sparsemax has a closed-form solution and can induce sparsity, but sparsemax is not adaptive and lacks flexibility as it is unable to switch from one sparsity pattern to another when necessary. We aim to combine the advantages and avoid the disadvantages using this new formulation

$$\Pi_\Omega(x) = \arg\min_{y\in\Delta^d} ||y - \gamma G(x)||^2, \tag{4.7}$$

with $G(x) : \mathbb{R}^d \to \mathbb{R}^d$ and γ being a learnable neural network and a scalar, respectively. By choosing different $G(x)$, $\Pi_\Omega(x)$ can exhibit different sparsity patterns including softmax and sparsemax. With $G(x)$ fixed, the parameter γ can control how sparse the output could be, similar to the temperature parameter in softmax. The summary in Table 4.2 shows that (4.7) will lead to a general mapping and can combine properties

such as translation and scaling invariance adaptively. Work in [85] proposed sparse-hourglass that can adjust the trade-off between translation and scaling invariance via tunable parameters. However, it is unclear under which circumstances one property is more desirable than the other, so there is little to no prior knowledge on how to tune such parameters. In contrast, our formulation in (4.7) can balance such trade-off via learning $G(x)$ and γ while work in [85] is based on a fixed form of $G(x)$ with tunable parameters.

While we can let the neural network learn $G(x) : \mathbb{R}^d \to \mathbb{R}^d$ without any restrictions, there is indeed prior knowledge that we can apply, e.g., monotonicity. It is desired to keep the monotonicity of $\Pi_\Omega(x)$, i.e., $\forall x_i > x_j, (\Pi_\Omega(x))_i > (\Pi_\Omega(x)_j$, as larger attention logit should be mapped into larger attention strength. As sparsemax is monotonic, this requires that $\forall x_i > x_j, G_i(x) > G_j(x)$, or in other words, the order of the input of $G(x)$ coincides with that of the output. To keep this property, $G(x)$ is designed component-wisely as $G_i(x) = \psi(\phi_1(x_i), \sum_i \phi_2(x_i))$, with $\psi : \mathbb{R}^2 \to \mathbb{R}^1, \phi_1, \phi_2 : \mathbb{R}^1 \to \mathbb{R}^1$ are neural networks with hidden layers. Note that $G_i(x)$ should be coupled with all of the entries of x instead of be a univariate function only depending on x_i, as demonstrated in Table II. As the second argument of ψ (i.e., $\sum_i \phi_2(x_i)$) is invariant to $G_i(x), \forall i \in [d]$, the order preserving of $G(x) : \mathbb{R}^d \to \mathbb{R}^d$ is equivalent to the monotonicity of $\psi(\bullet)$ and $\phi_1(\bullet)$. In order to keep this monotonicity, we enforce all the weights of the networks ψ and ϕ_1 to be positive [88], by applying an absolute value function on the weights. This architecture can accelerate the learning process with extra prior knowledge, as it is monotonic by design.

4.4.2 Message passing in MARL via GNN

We will present how the information is aggregated to learn a representation for per-agent value/policy network using a graph neural network. The scaled dot-product attention mechanism (Section 4.2.1) with our generalized sparsemax as the activation function, denoted as *sparse-Att*, is applied to learn a communication graph and pass messages through the connections in the graph.

We start with homogeneous multiagent system, where the relation between any

agent pair is identical. A graph is defined as $\mathcal{G} := (\mathcal{V}, \mathcal{E})$, where $v_i \in \mathcal{V}$ represent an agent and the cardinality of $\mathcal{V}$ is $|\mathcal{V}|$. Moreover, $e_{ij} \in \mathcal{E}$ is 1 if agent i and j can communicate directly (or agent j is observable to agent i), and 0 otherwise. This is a restriction on the communication graph and $\mathcal{E}$ is the set of all possible edges. Then sparse-Att aims to learn a subset of $\mathcal{E}$ via induced sparsity without compromising much optimality. For agent i, let $U_i = f_a(X_i)$ and E_i be its observation and entity encoding respectively, where $X_i, i \in \mathcal{V}$ is the local state and f_a is a learnable agent encoder network. Then the initial observation embedding of agent i, denoted as $h_i^{(1)}$, is

$$h_i^{(1)} = f_{mp}(U_i \| E_i), \tag{4.8}$$

where f_{mp} is another learnable network and the operator $\|$ denotes concatenation. Then at hop l (l-th round of message passing), agent i aggregates information from its possible neighbors belonging to the set $\mathcal{N} = \{j \in \mathcal{V} | e_{ij} = 1\}$ as follows

$$h_i^{(l+1)} = f_{mp}\Big(h_i^{(l)} \| \text{sparse-Att}_i^{\mathcal{N}}(h^{(l)})\Big). \tag{4.9}$$

With $l \geq 2$, the multi-hop message passing can enable the agent to obtain information from beyond its immediate neighbors. In the message aggregation from all of the agents $\mathcal{N}$, identical parameters are used in sparse-Att$_i^{\mathcal{N}}$, which enforces the permutation-invariance. This property is desirable because homogeneous agents are interchangeable.

However, interchangeability is no longer applicable to heterogeneous systems or mixed cooperative-competitive environment. For example, with $\mathcal{V}_1, \mathcal{V}_2 \subseteq \mathcal{V}$ being a two-team partition of $\mathcal{V}$, agents cooperate with other agents from the same team but compete against agents from the other team. For agent $i \in \mathcal{V}_1$, its teammate neighborhood and enemy neighborhood are $\mathcal{N}_+ = \{j \in \mathcal{V}_1 | e_{ij} = 1\}$ and $\mathcal{N}_- = \{j \in \mathcal{V}_2 | e_{ij} = 1\}$, respectively. The edges connecting teammates and enemies are called positive and negative edges. Then based on multi-relational GNN, agent i aggregates

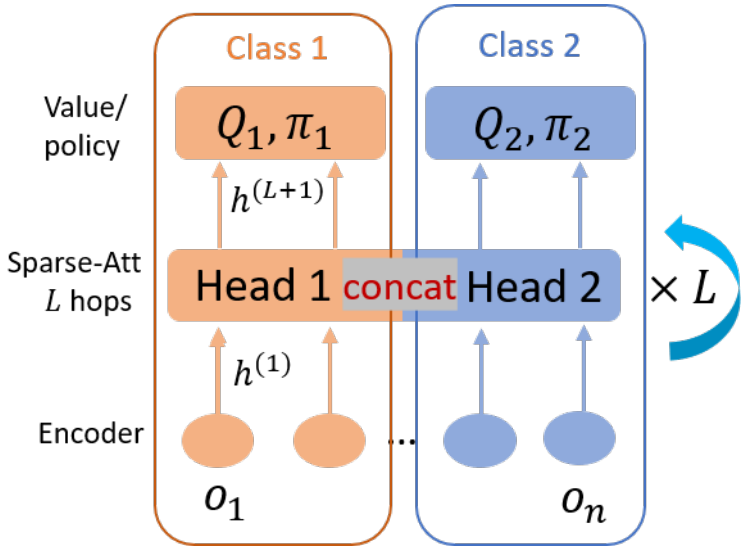


Figure 4-1: Our sparse-Att framework consists of three modules: encoder, multi-relational sparse attention mechanism, and value/policy network, with homogeneous agents sharing all parameters. Agents employ different attention heads to aggregate information alongside connections with different semantic meanings, followed by a concatenation. L is the number of the message-passing rounds; see (4.9). "concat" denotes the concatenation operation. Here only two classes (shown in red and blue) of heterogeneous agents are shown for simplicity.

information at hop l in the following way

$$h_i^{(l+1)} = f_{mp}\Big(h_i^{(l)} \| \text{sparse-Att}_i^{\mathcal{N}_+}(h^{(l)}) \| \text{sparse-Att}_i^{\mathcal{N}_-}(h^{(l)})\Big),$$

where $\text{sparse-Att}_i^{\mathcal{N}_+}$ and $\text{sparse-Att}_i^{\mathcal{N}_-}$ are different attention heads. Additionally, balance theory [89] suggests that "the teammate of my teammate is my teammate" and "the enemy of my enemy is my teammate." In a two-team competitive game, any walk (a sequence of nodes and edges of a graph) between an agent pair in the communication graph, comprising of both positive and negative edges, will lead to the same relation between the agent pair [90]. This property eliminates the ambiguity that the information aggregated from the same agent (but different walk) might have a different teammate/enemy property.

The proposed algorithmic framework is illustrated in Fig. 4-1. After L rounds of message passing, each agent has an updated encoding $h_i^{(L+1)}$. This encoding is then fed into the value network and the policy network, which estimate the state value and a probability distribution over all possible actions, respectively. As homo-

geneous agents are interchangeable, they share all of the parameters, including entity encoding, policy, value and message passing. Proximal policy gradient (PPO, [91]) is employed to train the model in an end-to-end manner. As only local information is required, the proposed approach is decentralized. Moreover, our approach maintains the transferability of GNN-based approaches as all the network dimensions are invariant to agent/entity number in the system.

4.5 Experiments

4.5.1 Task description

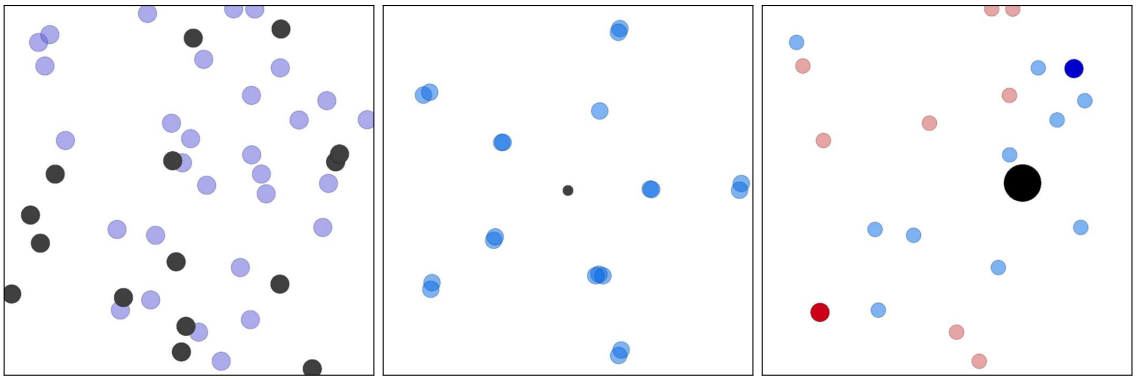
The proposed algorithm is evaluated in three swarm robotics tasks: Coverage, Formation, and ParticleSoccer [92], first two of which are cooperative and the third is competitive. The tasks are simulated in the Multiagent Particle Environment[1](MAPE [70]). The agents in MAPE can move in a 2-dimensional space following a double integrator dynamic model. The action space of the agents is discretized, with each agent can accelerate/decelerate in both X and Y direction. The three tasks are briefly introduced as follows.

Coverage: There are n_A agents (light purple) and n_L landmarks (black) in the environment (see illustration in Fig. 4-2a). The objective for the agents is to cover the landmarks with the smallest possible number of timesteps. Agents are not assigned to reach a certain landmark, but instead, have to figure out the assignment via communication such that the task can be finished optimally.

Formation: There are n_A agents (blue) and 1 landmarks (black) in the environment (see illustration in Fig. 4-2b), with n_A being an even natural number. The agents need to split into two sub-teams of equal size, with each of them building a formation of a regular pentagon. The two regular pentagons with different sizes are both centered at the landmark.

ParticleSoccer: There are n_A agents and 3 landmarks in the environment (see

[1]https://github.com/openai/multiagent-particle-envs



(a) Coverage (b) Formation (c) ParticleSoccer

Figure 4-2: Three different simulation tasks used in this work.

illustration in Fig. 4-2c), with the bigger landmark as a movable ball and the two smaller ones as a fixed landmark. A team wins the game via pushing the black ball to the opponent team's goal. The goal color of the light blue (red, resp.) team is blue (red, resp.).

4.5.2 Implementation specifications

The agent encoder $f_a(\bullet)$ and the entity encoder take input the 4-dimensional agent states and 2-dimensional entity states, respectively. The queries, keys, and values in all of the sparse attention mechanism are 128-dimensional. The communication hop is $L = 2$. All neural networks are fully connected with the ReLU activation function. In the sparsity-promoting function (4.7), ϕ_1, ϕ_2 and ψ all have one hidden layer with dimensions being 16, 16 and 64, respectively. The absolute value function is used to keep the weights of the monotonicity-preserving neural network positive.

Evaluation is performed every 320 episodes and PPO update is executed for 4 epochs after collecting experience of 4096 timesteps.

4.5.3 Results

In the cooperative scenarios i.e., Coverage and Formation, two metrics are used to evaluate the algorithms. The first is the average reward per step and the second is the task success rate. Higher means better performance for both metrics.

We compare our algorithms with two baselines: GNN-based MARL with dense attention mechanism [73] and MAAC [81]. These two algorithms are considered to be strong baselines as they reported advantageous results against algorithms including MADDPG [70], COMA [93], VDN [94] and QMIX [72]. Public repositories[2][3] are used for comparison. As both repositories also apply their algorithms on MAPE, the default hyperparameters are used for comparison.

In simulation, we set $n_A = 30$ and $n_A = 20$ for Coverage and Formation, respectively. Fig. 4-3 and Fig. 4-4 demonstrated that our algorithm can achieve higher rewards than the two baselines with fewer episodes. This validates that sparse-Att can accelerate the learning process via aggregating information from agents that matter the most. Moreover, in terms of the second metric, i.e., success rate, our algorithm consistently outperforms the two baselines by a significant margin (with a much smaller variance), as shown in Fig. 4-5. The evaluations of both metrics for two scenarios provide strong support for the advantages of our algorithm.

For the competitive ParticleSoccer task, we set $n_A = 20$ with both red team and blue team of size $\frac{n_A}{2} = 10$. As this task is competitive, the above two metrics are no longer applicable. Instead, we let the red (blue, resp.) play against a blue (red, resp.) team from another algorithm. Table 4.3 presents the results of the inter-algorithm competition. The overall score of each algorithm equals the sum of the winning evaluation episodes of its red team and blue team playing against blue and red team respectively from other algorithms. The overall scores in Table 4.3 show that our algorithm can learn strong policies.

4.5.4 Interpretability of the sparse communication graph

Let us proceed by considering the inherent sparity in Formation and ParticleSoccer. As mentioned in the description of the Formation scenario, the formation of each pentagon is related to half of the agents, while the sub-team assignments need to be learned. In the implementation, the reward is set to require that the first $\frac{n_A}{2}$ agents

[2]https://github.com/sumitsk/matrl.git
[3]https://github.com/shariqiqbal2810/MAAC

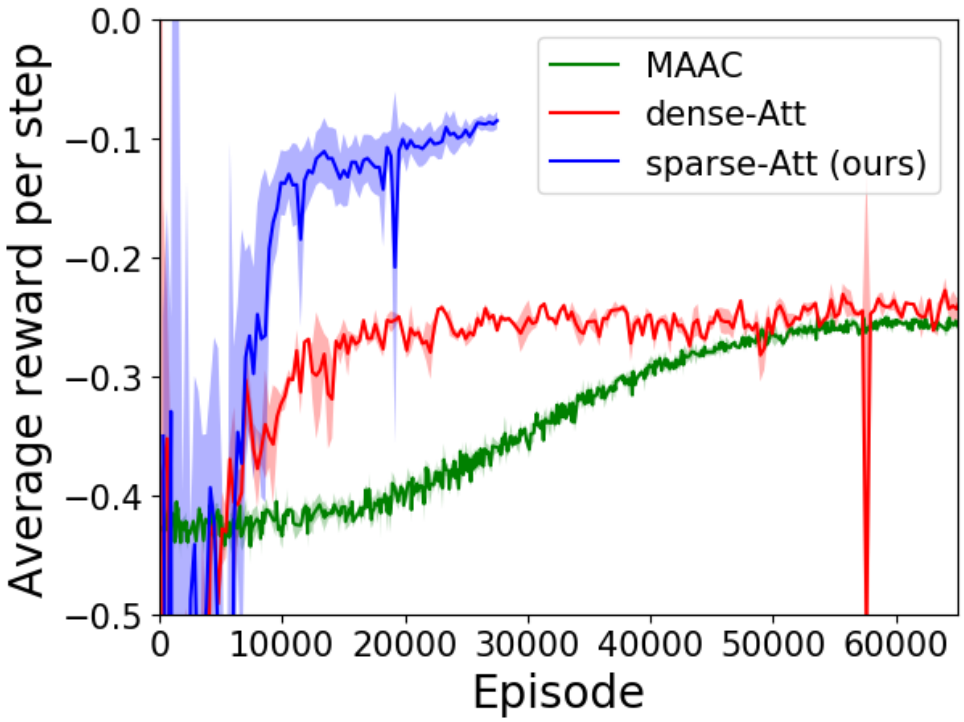


Figure 4-3: Reward comparison of our algorithm against two baselines for the Coverage task.

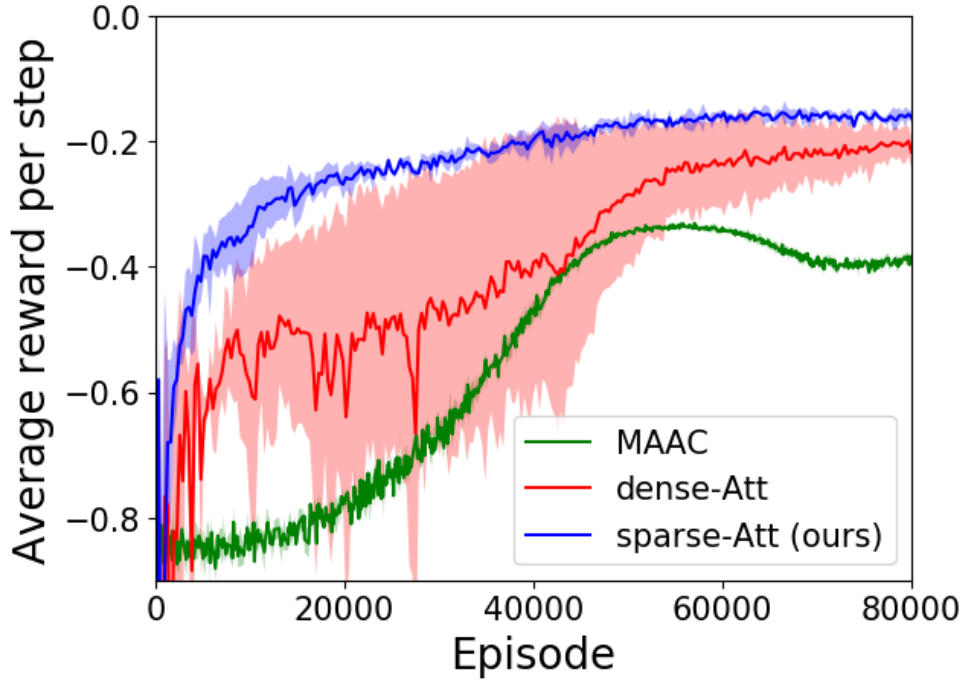
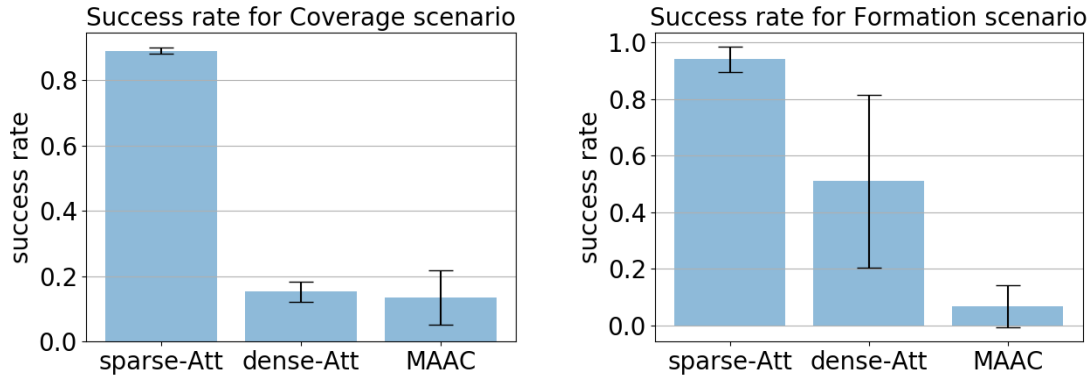


Figure 4-4: Reward comparison of our algorithm against two baselines for the Formation task.



(a) Coverage

(b) Formation

Figure 4-5: Performance comparison of three algorithm on two scenarios. Multiple policies learned from each algorithm are evaluated and the mean/standard deviation are plotted.

closest to the landmark build the formations of the inner pentagon and the remaining $\frac{n_A}{2}$ agents to build the formations of the outer pentagon. With the convergence of the learning algorithm, once a sub-team partition is learned to complete the two sub-tasks, the learned agent indexing of each team should not vary due to the distance sorting and the two pentagons are relatively far away. As a result, the reward to complete each sub-task is only related to the corresponding sub-team and hence the two sub-teams are decoupled from each other. The adjacency matrix of the learned communication graph shown in Fig. 4-6a validates that the inter-team communication is very sparse. This adjacency matrix is up to row/column permutation as indexing of each sub-team is learned without being known as a prior. Moreover, in a sub-team, the algorithm learns a communication graph similar to a star-graph. It can be understood that each sub-team selects a leader. As a star-graph is a connected graph with possibly minimum edges, this communication protocol is both effective and efficient. Also, the length of the path between any agent pair in a star graph is no greater than 2, which echos the two-hop communication ($L = 2$) we used in the simulation. That is because due to the two-hop message-passing, the agents can eventually communicate with agents as far as two edges away, which includes all of the agents in a star graph. Note that the sparsity on the diagonal entries of the

Table 4.3: Evaluation of three algorithms in the competitive ParticleSoccer task. Each pair is evaluated for 50 episodes and the $(\bullet, \bullet, \bullet)$ in each cell denotes the number of red team winning episodes, blue team wining episodes and the draw episodes. A draw means that neither team scores within a given episode length. win_{red} and win_{blue} are the winning episodes of the red and blue team, respectively when competing against blue and red team from other algorithms.

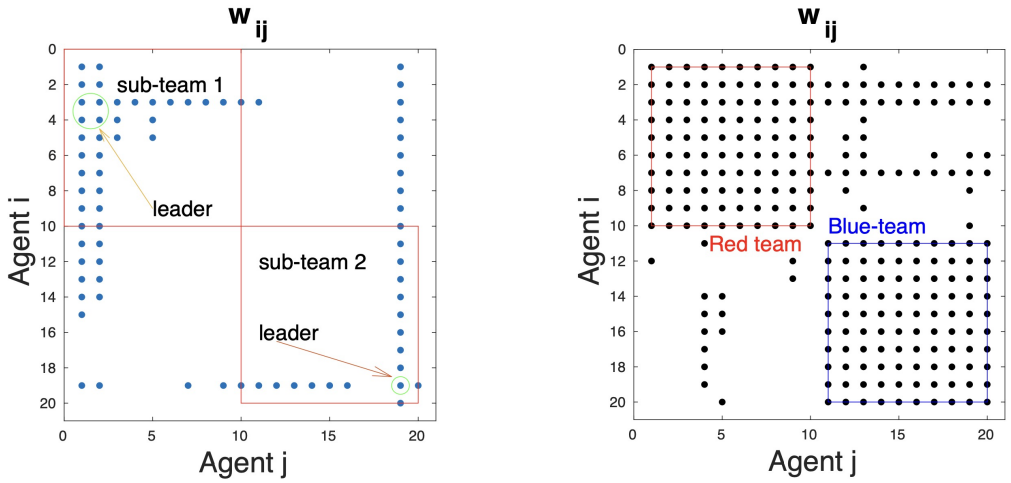
Red \ Blue	sparse-Att **(ours)**	dense-Att	MAAC	win_{red}
sparse-Att **(ours)**	$(48, 0, 2)$	$(15, 0, 35)$	$(26, 0, 24)$	41
dense-Att	$(9, 1, 40)$	$(5, 0, 45)$	$(3, 0, 47)$	11
MAAC	(7,0,43)	(2,0,48)	$(3, 0, 47)$	9
win_{blue}	-15	-17	-29	N/A
	sparse-Att **(ours)**	dense-Att	MAAC	
overall scores: $\text{win}_{\text{red}} + \text{win}_{\text{blue}}$	**26**	-6	-20	

communication graph does not mean that the agent's own information is neglected, as it is separately concatenated; see (4.9).

Also, in the ParticleSoccer scenario, from each team's perspective, agents need to coordinate tightly within the team to greedily push the ball to the other team's goal while only attending to a small number of agents from the other team. This leads to dense intra-team communication but relatively sparse inter-team communication. This is validated by the approximately block-diagonal adjacency matrix of the learned communication graph in Fig. 4-6b.

4.6 Summary

This work exploits sparsity to scale up MARL, which is motivated by the fact that interactions are often sparse in multiagent systems. We propose a new general and adaptive sparsity-inducing activation function to empower an attention mechanism, which can learn a sparse communication graph among agents. The sparse commu-



(a) Formation (b) ParticleSoccer

Figure 4-6: Sparse communication graph for two scenarios. For the Formation, our sparse-Att learns to split into two sub-team as desired and the learned sparse star-like communication graph makes communication both effective and efficient. In the ParticleSoccer, sparse-Att learn to pay more attention to teammates and a necessary subset of enemies.

nication graph can make the message-passing both effective and efficient such that the scalability of MARL is improved without compromising optimality. Our algorithm outperforms two baselines by a significant margin on three tasks. Moreover, for scenarios with inherent sparsity, it is shown that the sparsity of the learned communication graph is interpretable.

Chapter 5

Efficient Robust Policy Learning through Implicit Ensemble Training

5.1 Introduction

In competitive multiagent scenarios, the learned policy of each agent depends on the joint policy of all the other learning agents. However, since all agents are learning concurrently, this leads to a non-stationary environment [5]. One challenge resulting from such multiagent learning is that the policy learned from training might not perform well in the testing environment where the opponents' policies could be significantly different from those of the training opponents (distribution shift between training and testing). Even worse, the testing opponents could be trained to exploit the weakness of the policy learned from the training [20].

One effective approach to mitigate the performance degradation from training to testing is ensemble training, which has been applied in many previous works [5, 9, 16, 21, 31]. In ensemble training, each agent has multiple policies (as in [5, 9, 31]) or keeps multiple copies of previous policies (as in [16, 21]), from which one policy for each agent is sampled to play against each other. As a result, each policy is optimized against a distribution of the other agents' policies, which effectively reduces the distribution shift between training and testing (because a single policy can be regarded as a single Dirac-delta distribution centered at one point within the policy

space). However, one drawback of applying this ensemble training technique is the significant increase in computation and memory consumption due to the learning and storage of multiple policies. Besides, the number of policies required for guaranteed policy strength improvement is a function of the non-transitivity dimension [8] of the multiagent scenario, which could be on the order of tens for simple games such as Tic-Tac-Toe and more than thousands for complicated real-world games [8]. As a result, this ensemble training approach either becomes intractable due to constraints on computational resources or ends up with learning weak policies due to insufficient policy diversity.

In this work, we propose a novel implicit ensemble training (IET) approach by formulating ensemble training as a multitask learning problem. Instead of maintaining multiple policies explicitly with independent neural networks, our IET approach uses a unified hierarchical modular network architecture [95] with a learnable conditional latent variable distribution. The unified network architecture increases the knowledge sharing within the modules for improved learning efficiency and the conditional latent variable captures the diversity of the policy, which is necessary for robustness.

Our contributions are: 1) we identify the cause of inefficiency in standard ensemble training and suggest a multitask solution to improve the efficiency of ensemble training; 2) we propose a novel algorithm that extends ensemble training with latent variables and a multitask network; and 3) we show that the new algorithm improves both learning efficiency (with less computation) and robustness.

5.2 Background

5.2.1 MARL with policy distribution

In practice, learning with a single joint policy typically leads to over-fitting between policies and poor generalization to previously unseen policies. The distribution based learning objective, Eq. 2.3,

$$J_i = \mathbb{E}_{s\sim p^{\boldsymbol{\pi}}, \boldsymbol{a}\sim\boldsymbol{\pi}, \boldsymbol{\pi}\sim p^{\boldsymbol{\Pi}}}\left[R_i(s, \boldsymbol{a})\right], \tag{5.1}$$

is a more appropriate learning objective than the single policy learning objective Eq. 2.2.

One simple parametrization of $\mathcal{P}_j(\Pi_j)$ is a uniform distribution over a policy ensemble of fixed size (a set of independent policies, each parametrized by a neural network), such as in [5]. Policy space response oracle (PSRO) [32] is another parametrization, with an expanding set of policies. At each iteration, a new policy produced by an oracle, via solving for the (approximate) best response against a mixture of other agents' policies, is added to the policy ensemble. PSRO leads to better convergence towards Nash Equilibrium in poker games than self-play (single policy) and fictitious self-play (uniform distributed over previous policies). Despite the better convergence property, these ensemble approaches are not scalable as the computation increases linearly with respect to the ensemble size, while the number of policies to guarantee policy improvement may increases exponentially with respect to the complexity of the multiagent interactions [8]. This limitation motivates developing a new parametrization of policy distribution with better parameter-efficiency.

5.2.2 Related Work

This work is at the intersection of robust MARL and efficient MARL. Ensemble training for robust MARL has been studied in many previous works such as [5, 9, 21, 31, 50]. These approaches focus on improving the robustness of the learned policies, regardless of the associated increase of computational complexity. In contrast, our work focuses on improving the efficiency of ensemble training without sacrificing robustness. Another approach for robust MARL is minimax policy optimization, in which each agent optimizes its policy assuming all the other agents and the environment dynamics are adversarial (see [96, 97]). One difficulty with this approach is the requirement of solving the nested minimax optimization, which is typically approximated by optimizing the inner minimization for one step only [97]. In addition, the minimax formulation tends to result in overly-conservative policies because of the pessimistic assumption about the other agents.

Our work is also closely related to multi-task reinforcement learning (MTRL).

MTRL aims to improve the learning efficiency by knowledge sharing across tasks [98]. Recent works [95, 99, 100] found that the modular policy network is an effective architecture for improving parameter-efficiency via sharing of learned modules. However, that MTRL work focuses on solving the single-agent multi-task problem. In contrast, our work leverages the modular network architecture combined with latent conditional policy learning [101, 102] to improve the efficiency of ensemble-based robust MARL. The innovation of our approach is re-formulating ensemble training as multi-task learning while using a latent variable to preserve policy diversity which is essential for robust MARL.

5.3 Approach

In this section, we present a parameter-efficient parametrization of the policy distribution, which we call an implicit ensemble. We start from revealing the relationship between a uniform ensemble with a latent-conditioned policy, and then show how our implicit ensemble approach extends the uniform ensemble for higher parameter-efficiency and policy diversity.

5.3.1 Generalization of ensemble training as latent-conditioned policy

Ensemble training with a uniform distribution over a fixed-sized set of policies is a standard approach for improving the robustness of policies in MARL. In ensemble training, each agent's policy π_i is an ensemble of K sub-policies, with each denoted as $\pi_{\theta_i^{(k)}}$ or $\pi_i^{(k)}$ and parameterized by separate NN parameters $\theta_i^{(k)}$. The generative process for the ensemble training policy π_i is expressed as

$$\begin{aligned} k &\sim \text{unif(1, K)} = \frac{1}{K}\sum_{j=1}^{K}\delta(k-j), \\ \theta_i|k &\sim \delta(\theta_i - \theta_i^{(k)}), \\ a_i|\theta_i &\sim f_{\theta_i}(o_i), \end{aligned} \tag{5.2}$$

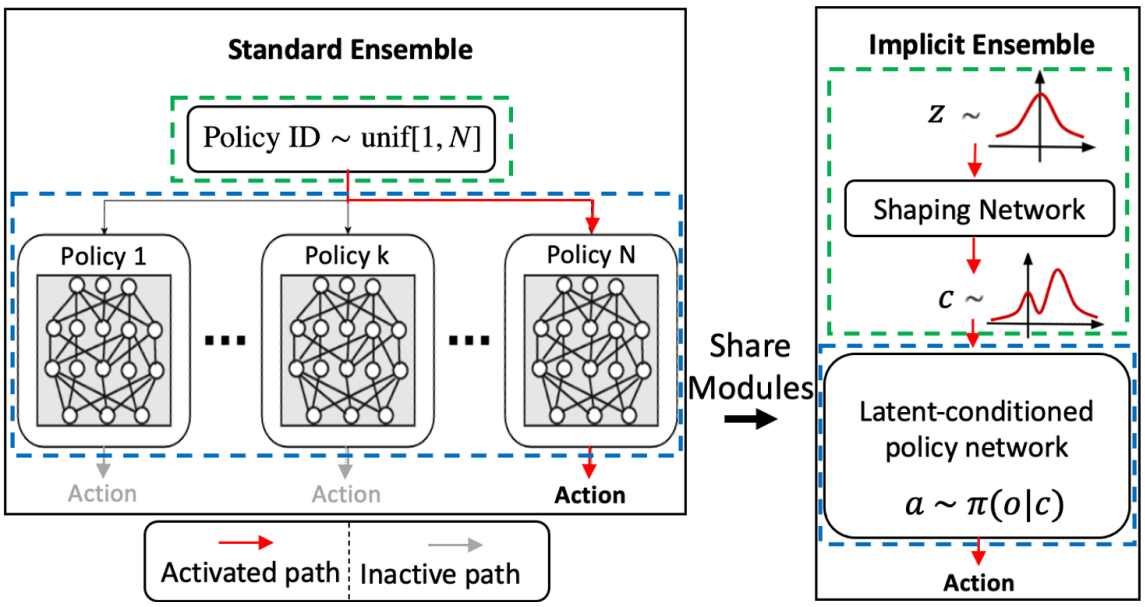


Figure 5-1: Connections between a standard ensemble and our proposed implicit ensemble: In the standard ensemble, a random index is sampled and the corresponding network is selected to output an action; while in the implicit ensemble, a random Gaussian variable is sampled and encoded into a latent vector by a shaping network, which is then passed to a conditional policy network to output an action.

where δ is the Dirac-delta distribution. One interpretation of Eq. 5.2 is that π_i is a conditional policy on a uniform discrete latent random variable k,

$$a_i|k \sim \bar{f}(o_i, \theta_i^{(1)}, \theta_i^{(2)}, \ldots, \theta_i^{(K)}; k) = f_{\theta_i^{(k)}}(o_i). \tag{5.3}$$

Eq. 5.3 suggests that ensemble training can be interpreted as a latent-conditioned policy [101–103] conditioned on the discrete random variable k coming from a uniform distribution. However, this parametrization is inefficient because only one out of the K sets of sub-policy parameters $[\theta_i^{(1)}, \theta_i^{(2)}, \ldots, \theta_i^{(K)}]$ is activated per sampling of a_i. As a result, the rollout from executing one set of sub-policy parameter cannot be used to optimize the rest $K-1$ sets of sub-policy parameters, which reduces the sample efficiency by a factor of K.

5.3.2 Implicit ensemble training

Our IET generalizes ensemble training via two steps: 1) relax the discrete random variable $k \in \{1, 2, \ldots, K\}$ into a continuous latent variable $c \in \mathbb{R}^L$ with a learned distribution that adaptively captures the diversity of the policy ensemble; 2) replace the K independent NNs with a unified modular network architecture with parameter ϕ that improves parameter-efficiency by knowledge sharing between sub-modules within the network. The continuous relaxation also makes the policy differentiable with respect to the latent variable, thus making it possible to synthesize new policies from the learned skills represented by the sub-modules within the modular network by perturbing the latent variable distribution.

The generative process of this implicit ensemble is

$$\begin{aligned} z &\sim \mathcal{N}(\mathbf{0}, \mathbb{I}_{L\times L}), \\ c|z &= g_\psi(z), \\ a_i|c &\sim h_\phi(o_i; c). \end{aligned} \tag{5.4}$$

In Eq. 5.4, a random Gaussian noise vector z is sampled from the standard multivariate Gaussian distribution (sampled once at the beginning of each episode and remains fixed during the episode), then passed through a shaping network parameterized by ψ to output a latent condition variable c. Finally, the action is sampled from a policy which is parameterized by ϕ and conditioned on the latent condition variable c. Both ψ and ϕ are learnable parameters that are optimized end-to-end with respect to the reinforcement learning objective.

The implicit ensemble Eq. 5.4 is a more flexible parameterization than the ensemble training Eq. 5.2. In fact, the latter is a special case of the former, where the distribution of the latent variable c collapses into a discrete distribution (corresponding to the uniform discrete distribution of k), and the shared parameter ϕ is divided into K disjoint sets of parameters each of which is activated in the policy network when only one of the discrete values of c is drawn.

In contrast to the ensemble training formulation Eq.5.2, where the ensemble size

K is a hyperparameter to control the diversity of the (ensemble) policy, Eq. 5.4 does not contain this explicit hyperparameter. Instead, the diversity of the policy is captured adaptively by the learned latent distribution parameterized by ψ: a complicated multi-modal distribution corresponds to high diversity, while a simple uni-modal distribution corresponds to low diversity.

5.3.3 Model architecture

Fig. 5-1 shows the model architecture for implementing the IET. There are two major components within this architecture: 1) a shaping network, corresponding to the mapping $g_\psi(\cdot)$ in Eq. 5.4 that transforms the Gaussian noise vector z into the latent condition variable c; and 2) a conditional policy network, corresponding to the parametrization h_ϕ.

The shaping network is responsible for adding diversity to the conditional policy for improving the robustness of the learned policy, and the conditional policy network improves parameter-efficiency via knowledge sharing. The detailed design of these two networks is presented as follows.

Shaping network

The shaping network takes a Gaussian noise vector $z \in \mathbb{R}^L$ as input and transforms z into a latent condition variable c, which modifies the standard multi-variate Gaussian distribution into a learned (complicated) distribution. We use a feed-forward network with 2 fully-connected layers followed by a Softmax layer to parameterize this shaping network.

Multi-tasking network

Recent studies [95, 99, 100] found that modular architecture is a parameter-efficient way of learning multi-tasking policies. In ensemble training, the multi-tasking requirement naturally arises from the fact that each policy is optimized against the ensemble of policies of the other agents.

To improve the parameter-efficiency of ensemble training, we use the modular architecture proposed in [95] as our conditional policy network for multi-tasking. The modular network is composed of a base network and a routing network. The base network is a modular network that has n layers, and each layer has m modules. Each module is a feedforward network, and the input and the output are d-dimension vectors. The routing network takes the observation and the latent condition variable as inputs, and it then outputs the n normalized weight vectors, one for each layer, for weighting the modules of the base policy network. The weight vectors $w^j \in \mathbb{R}^{m^2}$ are calculated as

$$\begin{aligned} p^{j=1} &= W_d^{j=1} \left(\sigma \left(F\left(o\right) \cdot H\left(c\right)\right)\right), \\ p^{j+1} &= W_d^j \left(\sigma \left(W_u^j p^j \cdot \left(F\left(o\right) \cdot H\left(c\right)\right)\right)\right), \\ w^j &= \text{Softmax}(p^j), \end{aligned} \tag{5.5}$$

where F and H are the embedding layers, which map the observation vector o and the latent condition variable c into D-dimension embeddings. σ is the activation function (we use the ReLU activation). W_u and W_d are fully-connected layers of size $\mathbb{R}^{D \times m^2}$ and $\mathbb{R}^{m^2 \times D}$, respectively.

The base network takes the observation vector o as input and outputs policy logits / value, with the following relationship between the i-th module in the $j+1$ layer's input and the k-th module in the j-th layer's output,

$$\hat{f}_i^{j+1} = \sum_{l=1}^{m} w_{i,l}^j \left(\sigma \left(W_l^j \hat{f}_l^j\right)\right), \tag{5.6}$$

where $W_l^j \in \mathbb{R}^{d \times d}$ is the learnable module parameters.

Remark 2. *Modular networks are not the only choice for the knowledge sharing. Any multi-tasking network architecture that takes in a latent conditional variable can be a substitution to this modular network architecture.*

5.4 Experiments

5.4.1 Scenarios

We evaluate our approach on two types of 2-player (which we refer to as the blue agent and the red agent hereafter) multiagent scenarios.

- **Board-game**: turn-based games implemented in the PettingZoo multiagent environment[1] [104] and the RLCard toolkit[2] [105]: **Connect Four**, **Leduc Hold'em**, and **Texas Hold'em (Limit)**.

- **RoboSchool-Racer**: continuous problems modified from the robot racing scenarios in the RoboSchool environment[3]: **Ant** and **Cheetah**, where we decompose each robot into front and rear parts, and assign opposite rewards to each part. As such, the front is learning to move forward while the rear is learning to move backward.

5.4.2 Baselines

We compare our approach with the following baselines:

1. **Single Policy Training** (SPT): a standard multiagent training approach wherein each agent optimizes only one policy.

2. **Simple Ensemble Training** (SET): a standard ensemble training approach wherein each agent optimizes an ensemble of policies against each other. We use an ensemble size of 3 for each agent (an ensemble of 3 policies has been shown to improve robustness significantly in previous works [5] and [21]), which also increases the amount of computation by a factor of 3.

3. **Minimax Training** (MT): achieves worst-case robustness through optimizing one's policy against the worst-case opponents, which is formulated as a minimax

[1] `https://www.pettingzoo.ml`
[2] `https://github.com/datamllab/rlcard`
[3] `https://github.com/openai/roboschool`

optimization. We use the one-step approximation approach in [97] with the Multiagent Actor-attention-critic (MAAC) algorithm [106]

5.4.3 Implementation detail

We used the RLlib [107] implementation of Proximal Policy Optimization (PPO) with a mini-batch size of 256 and a learning rate of 5×10^{-5}. We use independent networks for the policy and the value function approximations and set the following hyperparameters for IET: $L = 10$ for the latent condition variable dimension; $H = 64$ for the hidden layer dimension in the shaping network; $n = 2$ and $m = 2$ for the number of layers and number of modules of the modular network; $D = 64$ and $d = 64$ for the embedding and module hidden dimension. For the other approaches, each of the policy and the value networks consists of two fully-connected layers with 256 hidden units.

In the simple ensemble training setting, each sub-policy within an ensemble only has a probability of 1/3 to be selected for execution. If the same number of environment rollouts is used to train the simple ensemble as used for the other training settings, the simple ensemble policy will perform poorly due to insufficient training. Therefore, we roll out two additional environment simulations (but counted as one training step when training the simple ensemble) for a fair comparison at the cost of additional computational overhead.

To evaluate the robustness of the learned policies, we adopted a similar approach as in [9, 20] by training an independent exploiter agent. Specifically, we launched two concurrent threads, one for the training, the other for the testing, and repeated the following steps:

1. Train the blue agent and the red agent in the training thread for one training epoch.

2. Copy the blue agent's policy to the testing thread and freeze it.

3. Train the red exploiter agent in the testing thread against the fixed blue agent.

As a result, the red exploiter agent learns to exploit any weakness of the blue policy, and the corresponding reward is an informative indicator of the adversarial robustness of the blue policy.

5.4.4 Results and comparisons

This section discusses experimental results and compares baseline approaches with our proposed IET approach.

Adversarial robustness

We investigate the adversarial robustness of the learned blue policy by training a red exploiter agent against the frozen blue agent policy.

Fig. 5-3 shows the training and testing rewards of both agents corresponding to the three baseline approaches and our proposed IET approach in the board game **Leduc Hold'em** (plots for the other scenarios show similar patterns). Since the red exploiter agent is trained against the fixed blue policy during the testing, these plots reflect the adversarial robustness (assuming the red exploiter agent can successfully learn the optimal exploiting policy against the blue policy, which is not always true, as will be discussed in later sections) of these four training settings. The plots show that there is always a gap between the training and testing reward. The training blue reward is always higher than the corresponding testing blue reward, while the testing red reward is always higher than the training red reward, which manifests the robustness issue of policies learned through MARL. Moreover, sometimes the training reward and the testing reward are not positively correlated, for example, in Fig. 5-3a, which indicates that the single policy training approach suffers the most from the adversarial exploitation.

The simple ensemble training mitigates this robustness issue by adding policy diversity into the training, reflected by the smaller gaps, higher testing blue reward, and lower testing red reward, at the cost of triple computation and memory consumption. In contrast, our proposed IET approach achieves high blue agent testing rewards,

Table 5.1: Competition scores between SPT, SET, and IET. IET achieves the best adversarial robustness in **Connect Four (CF)** and **Leduc Hold'em (LH)**. Higher score implies that the blue policy is stronger than the red policy. The lowest scores across the columns is bold for each row, and the highest bold score across each row is marked green, which corresponds to the blue policy that achieves best out-of-distribution robustness.

CF	SPT	SET	IET
SPT	**-1.0±0.0**	-0.16±0.04	0.29±0.04
SET	0.35±0.04	**-0.98±0.01**	0.65±0.03
IET	1.0±0.0	1.0±0.0	**-0.33±0.04**
LH	SPT	SET	IET
SPT	**-0.63±0.12**	-0.34±0.09	0.12±0.07
SET	0.55±0.14	-0.04±0.1	**-0.07±0.09**
IET	0.47±0.12	0.11±0.09	**-0.03±0.09**
TH	SPT	SET	IET
SPT	**-1.47±0.32**	-0.12±0.11	0.0±0.05
SET	**-0.6±0.27**	-0.16±0.18	0.05±0.10
IET	0.39±0.24	**-0.09±0.16**	0.13±0.12

low red agent testing rewards, as well as relatively small gaps between the training and the testing rewards, outperforming the other three baselines, using a comparable amount of computation and memory as SPT.

5.4.5 Competition scores between training settings

To evaluate the out-of-distribution robustness, we also include the competition scores between the blue agent and the red exploiter agent, calculated as $(r_{\text{blue}} - r_{\text{red}})/2$ for each pair of training settings in Tables 5.1. The blue agent uses the policy learned in the training thread, while the red agent uses the exploiter policy learned in the testing thread. As a result, the diagonal scores are in favor of the red agent, because the red exploiter policy is trained against the corresponding blue policy, but not the other way around. The diagonal scores measure the adversarial robustness of the learned blue policy. In contrast, the off-diagonal scores measure the out-of-distribution robustness of the blue policy, because neither the blue policy nor the red policy has been trained

against each other. However, drawing conclusions directly from Tables 5.1 about the relative strength of the policies learned by the three training settings is difficult, because the competition scores show that there is no single policy that can dominate all the rest policies. This non-transitive nature of real-world games has also been observed in previous work [108].

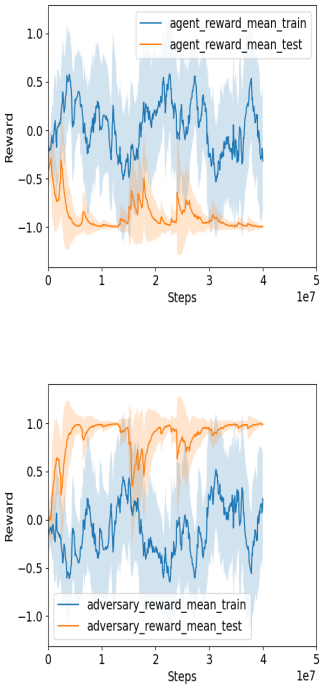
To quantitatively evaluate the robustness of the learned blue policy, we provide two metrics: (1) Normalized score: we find the worst-case reward gap between blue and red $\min_{\pi_r \in \Pi}[r_{\text{blue}}(\pi_b, \pi_r) - r_{\text{red}}(\pi_b, \pi_r)]$, $\Pi = [\text{SPT}, \text{SET}, \text{IET}]$ (higher is better) and normalize it to $[0, 1]$, (2) Nash policy probability: we follow the approach proposed in [109] by solving for the Nash-Equilibrium (NE) [34] of the meta-game [110], which involves a row-player and a column-player, whose actions are selecting which row/column policy to execute from the four available policies. The NE of the meta-game is a pair of stochastic policies. The probability of a policy being selected by the meta-player measures the strength (robustness) of this policy.

We show these two metrics in Table 5.2 and Table 5.3. The results show that our IET approach consistently outperforms the other ones across all the scenarios. Besides the improved robustness, the computational efficiency of our IET approach compares favorably to SET as shown by the normalized wall clock training time in Table 5.4.

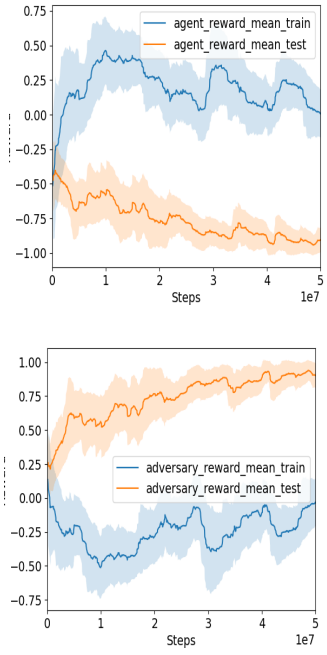
Comparison with the minimax robust approach

We also show in Table 5.5 the comparison between the ensemble training approaches with the minimax robust learning approach [97] which maximizes the worst-case reward to obtain adversarial robustness. We use the RoboSchool environment because the minimax approach requires differentiability with respect to the action (therefore continuous action environments).

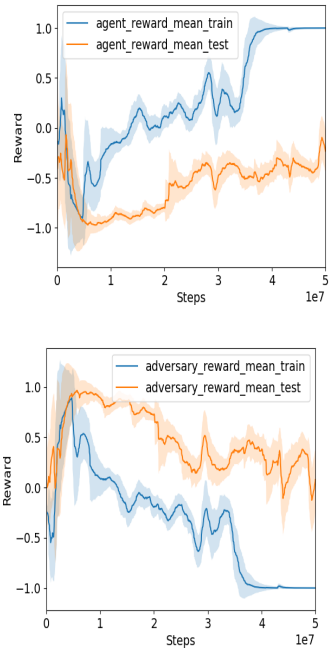
As the complexity of the environments increases, we see that the scores become noisier due to the difficulty of the reinforcement learning algorithm to optimize the policy (indicated by the fact that fewer minimal scores across the column are achieved at the diagonal). Again, we show the Nash meta-policy probability in Table 5.6, which



(a) Single Policy Training



(b) Simple Ensemble Training



(c) Implicit Ensemble Training

Figure 5-2: Training and testing reward of the blue/agent and the red/adversary in **Connect Four**

Table 5.2: Normalized scores that measure the adversarial robustness.

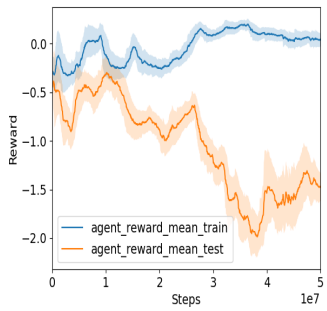
Scenarios / Settings	SPT	SET	IET
Connect Four	5e-3	1e-2	**0.33**
Leduc Hold'em	0.18	0.46	**0.48**
Texas Hold'em	0.0	0.2	**0.46**

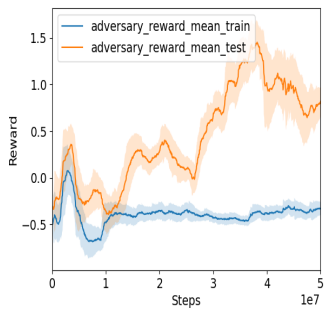
Table 5.3: Nash policy probability of the meta-player that measures the overall strength of the policy.

Scenarios / Settings	SPT	SET	IET
Connect Four	0.0	0.38	**0.62**
Leduc Hold'em	0.0	0.41	**0.59**
Texas Hold'em	0.0	0.0	**1.0**

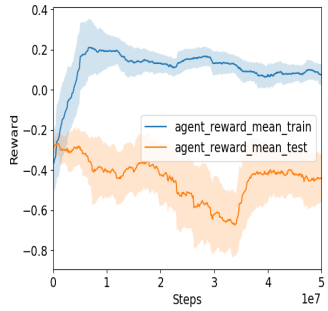
Table 5.4: Normalized wall clock time of training

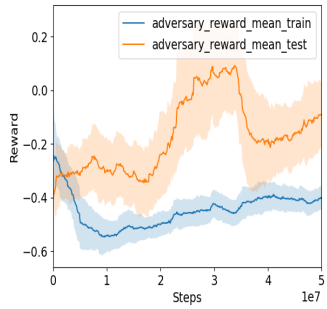
Settings	SPT	SET	IET
Normalized training time	1.0	3.0	1.53



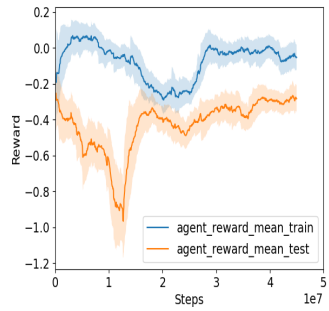


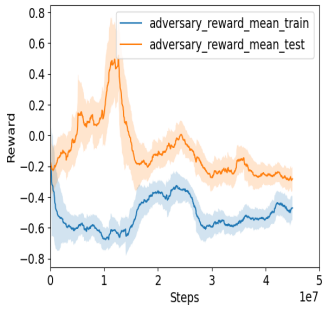
(a) Single Policy Training





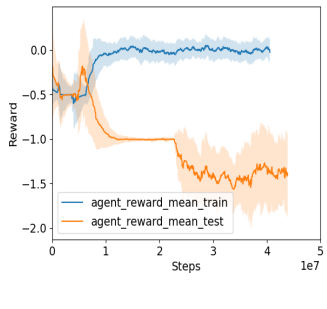
(b) Simple Ensemble Training

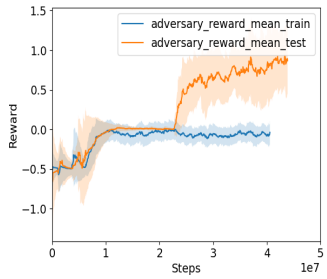




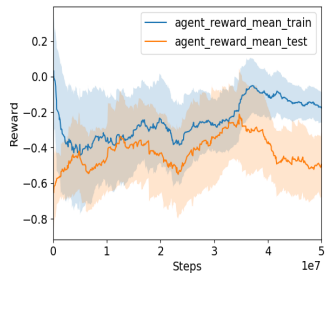
(c) Implicit Ensemble Training

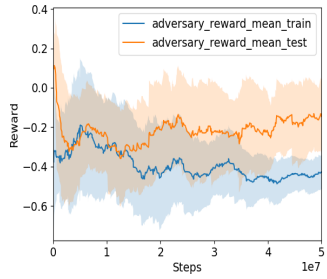
Figure 5-3: Training and testing reward of the blue/agent and the red/adversary in **Leduc Hold'em**



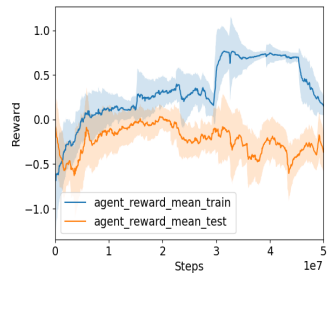


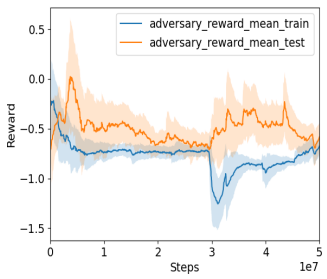
(a) Single Policy Training





(b) Simple Ensemble Training





(c) Implicit Ensemble Training

Figure 5-4: Training and testing reward of the blue/agent and the red/adversary in **Texas Hold'em**

Table 5.5: Competition scores between SPT, SET, IET, and MT. IET achieves the best adversarial robustness in **Ant (AT)** as well as best out-of-distribution robustness in both **Ant** and **Cheetah (CT)**.

AT	SPT	SET	IET	MT
SPT	8.42±0.26	26.2±0.82	1.95±0.54	**-4.49±2.86**
SET	**-35.3±1.54**	5.09±1.0	-23.7±0.86	-24.5±2.03
IET	**4.13±0.84**	33.64±1.03	15.2±0.38	37.7±0.81
MT	-25.37±1.73	-2.53±0.90	-13.85±1.44	**-25.7±0.50**
CT	SPT	SET	IET	MT
SPT	-15.54±0.68	**-15.95±0.72**	-7.16±0.34	4.26±0.22
SET	-9.05±0.21	**-12.60±0.64**	-1.76±0.41	6.45±0.29
IET	-9.22±0.22	**-11.88±0.27**	0.24±0.37	8.16±0.24
MT	-13.71±0.59	**-20.12±0.65**	-4.02±0.50	4.84±0.38

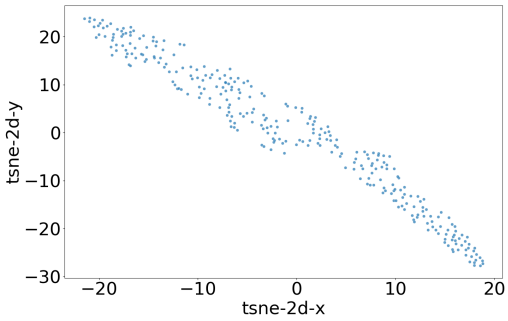
suggests our IET approach achieves the best robustness. The fact that our approach outperforms the minimax robust approach could be a result of two reasons: 1) The minimax formulation may not be a good approach to achieve out-of-distribution robustness since it aggressively optimizes for the worst-case reward, which sometimes may lead to overly-conservative behaviors that fail to exploit the weaknesses of the opponent; 2) The one-step solution technique for the minimax problem proposed in [97] is an approximate solution, which may find a sub-optimal solution due to the difficulty of selecting a suitable step-size parameter. We tuned this parameter by selecting from a few random values based on the testing reward, which may not be the best way of tuning this parameter.

Table 5.6: Nash meta-policy probability in the RoboSchool scenarios.

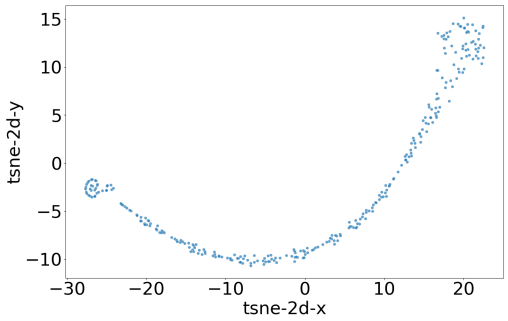
Scenarios / Settings	SPT	SET	IET	MT
Ant	0.0	0.37	**0.63**	0.0
Cheetah	0.0	0.0	**1.0**	0.0

5.4.6 Ablation studies

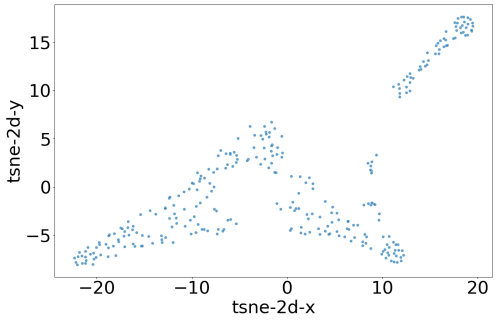
To further understand the role of the shaping network and the multi-tasking network, we conduct two additional experiments: 1) We show that the shaping network learns a



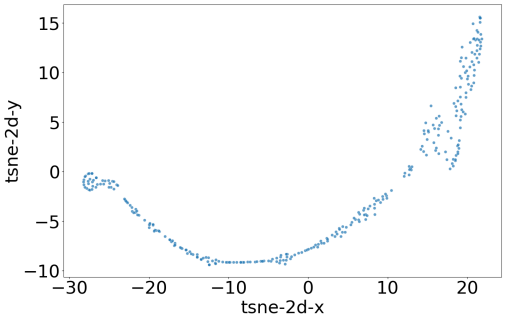
(a) MPE: Simple



(b) Connect Four



(c) Leduc Hold'em



(d) Texas Hold'em

Figure 5-5: t-SNE of the learned latent condition variable distributions, where the distribution in the single-agent (MPE: Simple) scenario is flat, while the distributions in the multiagent scenarios are complicated that involves multiple modes and clusters.

non-trivial mapping from the Gaussian noise to the latent variable only when diversity is required through visualizing the mappings represented by the shaping networks learned in multiagent scenarios and compare those with that learned in a single-agent scenario; 2) We show that we can improve the sample-efficiency of the PSRO algorithm by sharing the intermediate layers within a multi-tasking network, and the optimal performance is obtained at a medium level of sharing.

Visualization of latent variable distribution

Fig. 5-5 shows the 2D t-distributed stochastic neighbor embedding (t-SNE) visualization of the latent condition variable. In addition to the three board games, we also run our IET approach on the **Simple** scenario in the Multiagent Particle Environment (MPE) (a simple single-agent scenario where the agent is rewarded based on its distance to its goal at a random location). The figure shows two different pat-

terns of the latent condition variable distribution. The latent condition variable is on a flat hyper-plane in the single-agent scenario (MPE: Simple). In contrast, in the multiagent scenarios (Connect Four, Leduc Hold'em, and Texas Hold'em), the latent condition variables are on more complicated manifolds with curvatures and clusters, which indicates that the corresponding conditional policies have more diversity. This experiment suggests that our IET approach is capable of adaptively adjusting the degree of diversity through learning the shaping network end-to-end.

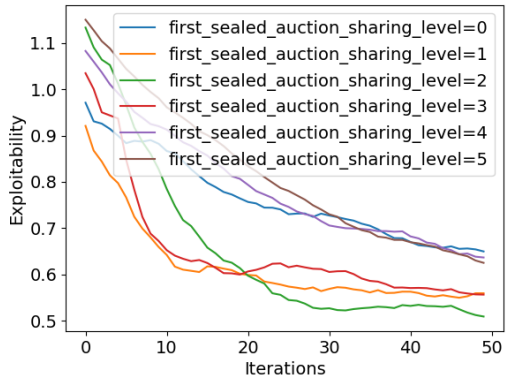
Ablation on the levels of sharing

We investigate how the level of sharing within the multi-tasking network influences the sample-efficiency. As our approach is not restricted to the modular network architecture, for the convenience of ablation, we instead use a more intuitive multi-tasking network architecture consisting of $L = 5$ fully connected layers, where the first L_{sharing} layers are shared and the last $L - L_{\text{sharing}}$ layers are independent for each policy.

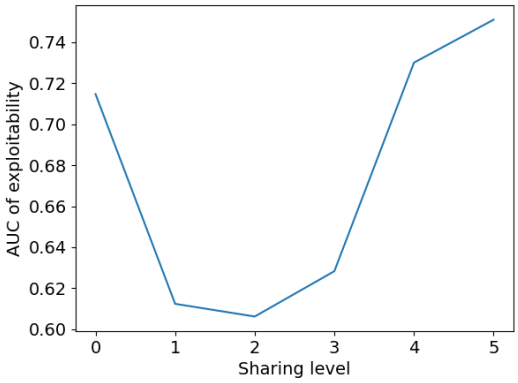
We show in Fig. 5-6a, 5-6b, 5-6c and 5-6d the exploitability and its area under the curve (AUC) of the joint policy when running the PSRO algorithm with the multi-tasking network of different sharing levels. With $L_{\text{sharing}} = 0$ (independent policies) corresponds to the standard PSRO, and $L_{\text{sharing}} = 5$ (identical policies) corresponds to self-play. We see that the best exploitability descent happens at $L_{\text{sharing}} = 2$, while the two extremes ($L_{\text{sharing}} = 0, 5$) perform poorly. This observation suggests a trade-off between knowledge sharing (positive transfer) and loss of flexibility (negative transfer), which is commonly observed in multi-task learning. This ablation study also verifies our design purpose that the multi-tasking network in our implicit ensemble approach is responsible for improving the sample-efficiency via sharing parameters.

5.5 Summary

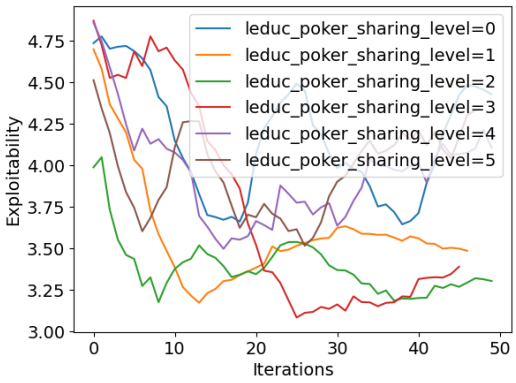
This work proposes IET, an implicit ensemble training approach that effectively reduces the computational complexity of ensemble training while maintaining the pol-



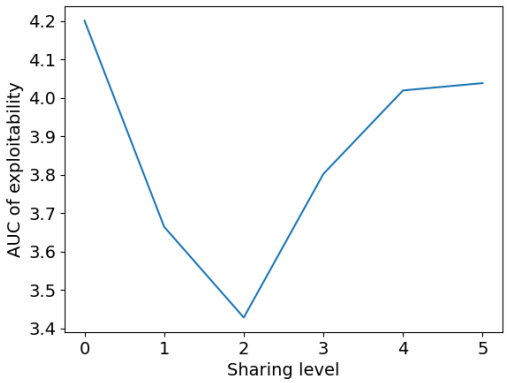
(a) Exploitability of FSA



(b) AUC of FSA



(c) Exploitability of LH



(d) AUC of LH

Figure 5-6: Exploitability (interpreted as a distance from Nash Equilibrium [32]) and the area under the curve (AUC) of the joint policies learned through running PSRO with the multi-task network of different sharing levels. The optimal exploitability descent happens at medium sharing levels ($L_{\text{sharing}} = 2$). Scenarios First Sealed Auction (FSA) and Leduc Hold'em (LH) implementation from OpenSpiel.

icy diversity for learning robust multiagent policies. In contrast to previous ensemble training approaches that require optimizing multiple policy networks, our IET approach optimizes a single shared network, which requires less computation and memory. Numerical results show that our approach improves both the learning efficiency and the robustness of the learned policy. For future works, we would like to explore network architectures that strike a better trade-off between scalability and optimality.

Chapter 6

Safe Adaptation through Ensemble Regularization

6.1 Introduction

One critical step towards building fully autonomous intelligent robots is to enable the capability of continual adaptation to new environments. In multiagent scenarios, besides the changing environment dynamics, agents must also adapt to the novel/evolving behaviors of other agents, some of which may not have been seen during training.

6.1.1 Related Work

There has been a lot of recent progress on fast adaptation to new tasks via meta-learning in both single-agent RL [11, 12, 111–113], and MARL [14, 15, 114]. While learning involves an algorithm that improves with experience on a task, meta-learning is an algorithm that is used across multiple tasks that improves with experiences and tasks [111]. During the meta-training phase, the agent meta-learns from tasks sampled from a task distribution how to quickly learn from a new task, which enables fast learning during the meta-testing phase, where testing samples are tasks drawn from the task distribution. One important assumption within the meta-learning frame-

work is that the task distribution is stationary [13]. As a result, the tasks encountered during the meta-testing phase are sampled from the same distribution as those encountered during the meta-training phase. However, in multiagent competitive settings, assuming access to the task distribution (sampling from the unknown opponent agent's policy distribution) is often unrealistic. Furthermore, as the ego-agent adapts to the opponent, the opponent may also adapt to the ego-agent concurrently, leading to non-stationarity from the ego-agent's perspective [114, 115]. Achieving effective adaptation requires knowledge about the opponent's learning dynamics [14, 114], which is a very strong assumption in competitive scenarios.

In addition to the lack of knowledge about the opponent, another common challenge of adaptation in competitive scenarios is the need to avoid being exploited by the opponent. In two-player zero-sum games, to exploit the opponent, the ego-agent has to deviate from the Nash-Equilibrium [116], leading to increased exploitability. Previous works demonstrated search-based safe exploitation in extensive-form games [117] (games that can be represented as a decision tree), where the ego-agent updates its strategy via subgame-resolving leveraging a model of the opponent's strategy without substantially increasing the exploitability [23, 118–120]. However, these approaches are specialized for extensive-form games with knowledge of the whole game tree, so it is unclear how to extend these approaches to handle more general multiagent settings, potentially with continuous dynamics, within the model-free reinforcement learning setting.

This work focuses on safe adaptation in two-player zero-sum scenarios (as an important initial step towards multiagent safe adaptation), where the ego-agent needs to update its policy based on the limited amount of interaction experience with an opponent agent. The goal of adaptation is to exploit the specific opponent that the ego-agent is interacting with while satisfying a safety requirement, which is that the ego-agent must also maintain high competitiveness against any other possible opponent (with either a stationary or evolving policy) during the whole adaptation phase. We investigate this problem under the framework of Markov game [29] and MARL in a model-free setting without explicit assumptions on the state or action

space. As such, the main contributions of this work are:

1. This work presents a novel Bayesian formulation of the safe adaptation problem within the MARL framework, which bridges the connection between robust MARL and safe adaptation.

2. This work proposes an optimization objective for modeling the opponent, with a behavior cloning term for adaptation and a novel ensemble-regularization term to achieve low exploitability, which is derived from the Bayesian formulation.

3. This work demonstrates that the proposed approach achieves adaptation by learning from a limited amount of interaction experience with the opponent while maintaining low exploitability against a second opponent that actively co-adapts to exploit the ego-agent.

6.2 Background

This work uses the Markov Game and MARL with policy distribution as the decision-making framework. Given the focus on two-player zero-sum games, Eq. 2.3 is re-written from the ego-agent's perspective as:

$$J_{\text{ego}} = \mathbb{E}_{\pi_{\text{ego}} \sim p(\mathbf{\Pi}_{\text{ego}}), \pi_{\text{oppo}} \sim p(\mathbf{\Pi}_{\text{oppo}})} \left[\mathbb{E}_{s \sim p^{\boldsymbol{\pi}}, \boldsymbol{a} \sim \boldsymbol{\pi}} \left[R_{\text{ego}}(s, \boldsymbol{a}) \right] \right], \tag{6.1}$$

where the ego-agent optimizes its policy distribution $\mathbf{\Pi}_{\text{ego}}$, subject to a given opponent policy distribution $\mathbf{\Pi}_{\text{oppo}}$. Since the optimal $\mathbf{\Pi}_{\text{ego}}$ with respect to Eq. 6.1 depends on the opponent policy distribution $\mathbf{\Pi}_{\text{oppo}}$, determining $\mathbf{\Pi}_{\text{oppo}}$ is critical. Here, we discuss some common choices for the opponent policy distribution $\mathbf{\Pi}_{\text{oppo}}$.

6.2.1 Oracle policy distribution

Suppose we know the true policy distribution of the opponent, we can optimize the ego-agent policy distribution against the opponent policy distribution to obtain $\mathbf{\Pi}_{\text{ego}}^{\text{oracle}}$. However, there are two problems with this approach: 1) Feasibility:

In competitive scenarios, it is unlikely to get access to the policy of the opponent. 2) Robustness: As the ego-agent over-fits its policy to the opponent, the resulting $\mathbf{\Pi}_{\text{ego}}^{\text{oracle}}$ may not perform well (or even poorly as shown in [20]) against an adversarial opponent that is trained against $\mathbf{\Pi}_{\text{ego}}^{\text{oracle}}$ to exploit its weakness.

6.2.2 Learned opponent policy distribution

As the ego-agent interacts with the opponent, the ego-agent can learn an internal model of the opponent policy $\mathbf{\Pi}_{\text{oppo}}^{\text{model}}$ distribution from the interaction experience as an approximation of the true opponent policy distribution [121–124]. Many previous works [121–123], assume access to the opponent's observation and action for this model learning, which is not a strong assumption in robotic domains with full-observability over the state space since the opponent's observation and action can be deducted from the state observation. Besides, [124] also demonstrates the possibility of learning an opponent model from the ego-agent's observation alone via variational inference over a hidden space that models the opponent's private information. However, this opponent modeling approach also suffers from the robustness problem mentioned before.

6.2.3 Nash Equilibrium policy distribution

Another way to model the opponent is to solve for the Nash Equilibrium policy distribution $\mathbf{\Pi}_{\text{oppo}}^{\text{nash}}$ (or equivalently, the minimax solution [125] in two-player zero-sum games [34]), with no prior knowledge of the opponent's policy distribution. The corresponding optimal policy distribution for the ego-agent is also the Nash Equilibrium $\mathbf{\Pi}_{\text{ego}}^{\text{nash}}$, which is the least exploitable policy. However, this approach does not attempt to adapt to the opponent that the ego-agent is interacting with, leading to sub-optimal performance against an opponent that is exploitable.

6.3 Approach

A better approach to modeling the opponent's policy distribution should leverage the available interaction experience for adaptation as well as stay close to the equilibrium distribution to ensure robustness against adversarial exploitation. We develop this approach by reformulating the sub-problem within Eq. 6.1 of modeling the opponent policy distribution $\pi_{\text{oppo}} \sim p\left(\mathbf{\Pi}_{\text{oppo}}\right)$ into a Bayesian inference problem over the space of policy distributions given the interaction experience $\mathcal{D}$ between the ego-agent and the opponent:

$$\begin{aligned} \pi_{\text{oppo}} \sim p\left(\mathbf{\Pi}_{\text{oppo}}|\mathcal{D}\right) &\propto p\left(\mathbf{\Pi}_{\text{oppo}}|\emptyset\right) p\left(\mathcal{D}|\mathbf{\Pi}_{\text{oppo}}\right), \\ &= p^{\text{NE}}\left(\mathbf{\Pi}_{\text{oppo}}\right) p\left(\mathcal{D}|\mathbf{\Pi}_{\text{oppo}}\right), \end{aligned} \tag{6.2}$$

where $\emptyset$ denotes the empty set, so $p^{\text{prior}}\left(\mathbf{\Pi}_{\text{oppo}}\right) = p\left(\mathbf{\Pi}_{\text{oppo}}|\emptyset\right)$ is the prior distribution over the opponent's policy space before obtaining any interaction experience. We argue that the Nash Equilibrium policy distribution is a sensible choice for this prior, i.e. $p^{\text{prior}}\left(\mathbf{\Pi}_{\text{oppo}}\right) = p^{\text{NE}}\left(\mathbf{\Pi}_{\text{oppo}}\right)$, because, with no information about the opponent, the best choice is to minimize the ego-agent's exploitability. As the ego-agent receives more interaction experience with the opponent, the posterior distribution $p\left(\mathbf{\Pi}_{\text{oppo}}|\mathcal{D}\right)$ is updated through the likelihood term $p\left(\mathcal{D}|\mathbf{\Pi}_{\text{oppo}}\right)$ while being regularized by the prior term, which ensures adaptation to the opponent while maintaining low exploitability.

However, the posterior inference problem Eq. 6.2 is challenging for two reasons: 1) Solving for the Nash Equilibrium policy distribution $p^{\text{NE}}\left(\mathbf{\Pi}_{\text{oppo}}\right)$ is a challenging problem in itself; 2) representing and parameterizing the policy distribution is challenging. Therefore, we apply the following two approximations,

1. The Nash Equilibrium policy distribution is approximated by an ensemble of policies generated via Algorithm 3, which has been shown in [5, 20, 21] to produce robust agent behaviors that are much less exploitable than policies generated without ensembling.

2. The posterior distribution over policy Eq. 6.2 is approximated by the maximum a posteriori probability (MAP) estimate via a single opponent policy model.

With these two approximations, we propose an alternative formulation for the estimated opponent policy as the optimization problem

$$\hat{\pi}_{\text{oppo}} = \arg\min_{\pi}[\mathbb{D}(\mathbf{\Pi}_{\text{oppo}}^{\text{ensemble}} \mid \pi) + \lambda_1 \mathbb{L}_{\text{likelihood}}(\mathcal{D} \mid \pi) \quad + \lambda_2 \mathbb{L}_{\text{RL}}(\mathbf{\Pi}_{\text{ego}}^{\text{ensemble}}, \pi)], \quad (6.3)$$

where λ_1 and λ_2 are hyperparameters. The first term $\mathbb{D}(\mathbf{\Pi}_{\text{oppo}}^{\text{ensemble}} \mid \pi)$ denotes a distance metric between the opponent policy ensemble generated via Algorithm 3 and the estimated opponent model, which regularizes the opponent policy to stay close to the robust ensemble policy distribution. This term corresponds to the prior term $p^{\text{NE}}(\mathbf{\Pi}_{\text{oppo}})$ in Eq. 6.2. The second term $\mathbb{L}_{\text{likelihood}}(\mathcal{D} \mid \pi)$ is the log-likelihood of observing the interaction experience $\mathcal{D}$ given the opponent policy, which corresponds to the likelihood term $p(\mathcal{D}|\mathbf{\Pi}_{\text{oppo}})$ in Eq. 6.2. The third term $\mathbb{L}_{\text{RL}}(\mathbf{\Pi}_{\text{ego}}^{\text{ensemble}}, \pi)]$ is the reinforcement learning loss that optimizes the opponent policy against the ego-agent's policy ensemble,

$$\mathbb{L}_{\text{RL}}(\mathbf{\Pi}_{\text{ego}}^{\text{ensemble}}, \pi) \quad = -\mathbb{E}_{\pi_{\text{ego}} \sim p\left(\mathbf{\Pi}_{\text{ego}}^{\text{ensemble}}\right), \pi_{\text{oppo}} = \pi} \left[\mathbb{E}_{s \sim p^{\boldsymbol{\pi}}, \boldsymbol{a} \sim \boldsymbol{\pi}} \left[R_{\text{oppo}}(s, \boldsymbol{a})\right]\right], \quad (6.4)$$

where the minus sign ensures that minimizing this loss results in maximization of the reward. This term did not appear in Eq. 6.2, but this term enables the opponent policy to continually evolve as we update the ego-agent's policy ensemble to adapt to the opponent via optimizing its reinforcement learning objective,

$$J_{\text{ego}} = \mathbb{E}_{\pi_{\text{ego}} \sim p\left(\mathbf{\Pi}_{\text{ego}}^{\text{ensemble}}\right), \pi_{\text{oppo}} = \hat{\pi}_{\text{oppo}}} \left[\mathbb{E}_{s \sim p^{\boldsymbol{\pi}}, \boldsymbol{a} \sim \boldsymbol{\pi}} \left[R_{\text{ego}}(s, \boldsymbol{a})\right]\right]. \quad (6.5)$$

We include $\mathbb{L}_{\text{RL}}$ in Eq. 6.3 because we found that although $\mathbb{L}_{\text{RL}}$ does not make much difference when adapting to a stationary opponent, it could be critical for achieving high competitiveness against an evolving opponent, as is shown later in the experiment section 6.4.2.

Now we discuss our choice for the first two terms in Eq. 6.3. The first term $\mathbb{D}(\mathbf{\Pi}_{\text{oppo}}^{\text{ensemble}} \mid \pi)$ measures the discrepancy between the policy ensemble $\mathbf{\Pi}_{\text{oppo}}^{\text{ensemble}}$ and our estimated opponent policy π. There are several closed-form metrics to measure

Algorithm 3 Ensemble training

Require: Ensemble size N, number of training iterations K

Randomly initialize policy ensembles: $\mathbf{\Pi}_{\text{ego}}^{\text{ensemble}} = \{\pi_{\text{ego}}^{i}\}_{i=1:N}$, $\mathbf{\Pi}_{\text{oppo}}^{\text{ensemble}} = \{\pi_{\text{oppo}}^{i}\}_{i=1:N}$
for k = 1:K **do**
    Randomly sample policy index: $j \sim \{1, \ldots, N\}, l \sim \{1, \ldots, N\}$
    Environment_rollout(π_{ego}^{j}, π_{oppo}^{l})
    Update π_{ego}^{j} and π_{oppo}^{l} to optimize objective Eq. 2.3
end for
return $\mathbf{\Pi}_{\text{ego}}^{\text{ensemble}}$ and $\mathbf{\Pi}_{\text{oppo}}^{\text{ensemble}}$

Algorithm 4 Safe adaptation

Require: Policy ensembles $\mathbf{\Pi}_{\text{ego}}^{\text{ensemble}}$ and $\mathbf{\Pi}_{\text{oppo}}^{\text{ensemble}}$, interaction experience $\mathcal{D}$, number of iterations K

freeze $\mathbf{\Pi}_{\text{oppo}}^{\text{ensemble}}$
Initialize $\hat{\pi}_{\text{oppo}}^{0}$ as random policy
$\mathbf{\Pi}_{\text{oppo}}^{\text{ensemble},0} \leftarrow \mathbf{\Pi}_{\text{oppo}}^{\text{ensemble}}$
for k = 1:K **do**
    $\hat{\pi}_{\text{oppo}}^{k} \leftarrow update\_opponent(\hat{\pi}_{\text{oppo}}^{k-1}, \mathbf{\Pi}_{\text{oppo}}^{\text{ensemble}},$
    $\mathbf{\Pi}_{\text{ego}}^{\text{ensemble},k-1}, \mathcal{D})$ ▷ one gradient step of Eq. 6.3
    $\mathbf{\Pi}_{\text{ego}}^{\text{ensemble},k} \leftarrow update\_ego\_agent(\hat{\pi}_{\text{oppo}}^{k}, \mathbf{\Pi}_{\text{ego}}^{\text{ensemble},k-1})$ ▷ one gradient step of Eq. 6.5
end for
return $\mathbf{\Pi}_{\text{ego}}^{\text{ensemble},K}$

the discrepancy between two policies, including KL-divergence discrepancy [126], total variation distance [127] and maximum mean discrepancy [128]. However, it is unclear how to select one metric over another given a specific application domain, and whether the selected metric can optimally discriminate between two policies. To resolve this ambiguity and achieve strong discriminative power, we choose to learn the discrepancy metric via adversarial learning following the paradigm of generative adversarial imitation learning (GAIL) [129]. In this approach we train a discriminator $D_w(o, a) : \mathcal{O}_{\text{oppo}} \times \mathcal{A}_{\text{oppo}} \to [0, 1]$ to minimize the following discrimination loss,

$$\mathbb{E}_{\tau_\pi}[\log D_w(o, a)] + \mathbb{E}_{\tau_{\mathbf{\Pi}_{\text{oppo}}^{\text{ensemble}}}}[\log(1 - D_w(o, a))], \tag{6.6}$$

such that

$$\mathbb{D}(\mathbf{\Pi}_{\text{oppo}}^{\text{ensemble}} \mid \pi) = -\mathbb{E}_{\tau_\pi}[\log D_w(o, a)], \tag{6.7}$$

where the shorthand notation $\tau_{(\cdot)}$ denotes the trajectory distribution when the ego-agent follows policy $\pi_{\text{ego}} \sim p\left(\mathbf{\Pi}_{\text{ego}}\right)$, while the opponent follows policy $(\cdot)$. This loss function is minimized by maximizing an imitation reward $r_{\text{imit}} = \log D_w(o, a)$.

The second term in Eq. 6.3 is the log-likelihood of observing the experience $\mathcal{D}$ given the opponent policy π. Since the opponent policy can only affect the probability of the opponent's taken action, this term can be reduced to behavior cloning loss,

$$p\left(\mathcal{D} \mid \pi\right) = \sum_{(a,o)\in\mathcal{D}} \log \pi(a \mid o). \tag{6.8}$$

In practice, we use mini-batch to calculate the gradient of this loss.

6.4 Experiments

6.4.1 Experiment setting

We evaluate our safe adaptation approach on the Multiagent Mujoco domain [130], where each robot is decomposed into parts that are controlled by individual agents as illustrated in Fig. 6-1. We use a zero-sum reward where the ego-agent tries to maximize the reward for moving forward and the opponent agent tries to minimize this reward.

To evaluate the capability of safe adaptation to a previously unseen opponent, we describe the following procedure to set up the evaluation.

Off-line training phase:

1. Alternating for K_1 iterations, between one-step ($K = 1$) policy ensemble training of size $N = 5$ for both agents, $\mathbf{\Pi}_{\text{ego}}^{\text{ensemble}}, \mathbf{\Pi}_{\text{oppo}}^{\text{ensemble}}$ via Algorithm 3 and one-step training of exploiter opponent $\pi_{\text{oppo}}^{\text{exp}}$ via Algorithm 5.
2. Freeze $\mathbf{\Pi}_{\text{ego}}^{\text{ensemble}}, \mathbf{\Pi}_{\text{oppo}}^{\text{ensemble}}$, and training exploiter opponent $\pi_{\text{oppo}}^{\text{exp}}$ against $\mathbf{\Pi}_{\text{ego}}^{\text{ensemble}}$ for an additional K_2 iterations.

On-line adaptation phase (for $\mathbf{\Pi}_{\text{ego}}^{\text{ensemble}}$ to adapt to $\pi_{\text{oppo}}^{\text{exp}}$):

1. Freeze $\pi_{\text{oppo}}^{\text{exp}}$, $\mathbf{\Pi}_{\text{oppo}}^{\text{ensemble}}$, and $\mathbf{\Pi}_{\text{ego}}^{\text{ensemble}}$.

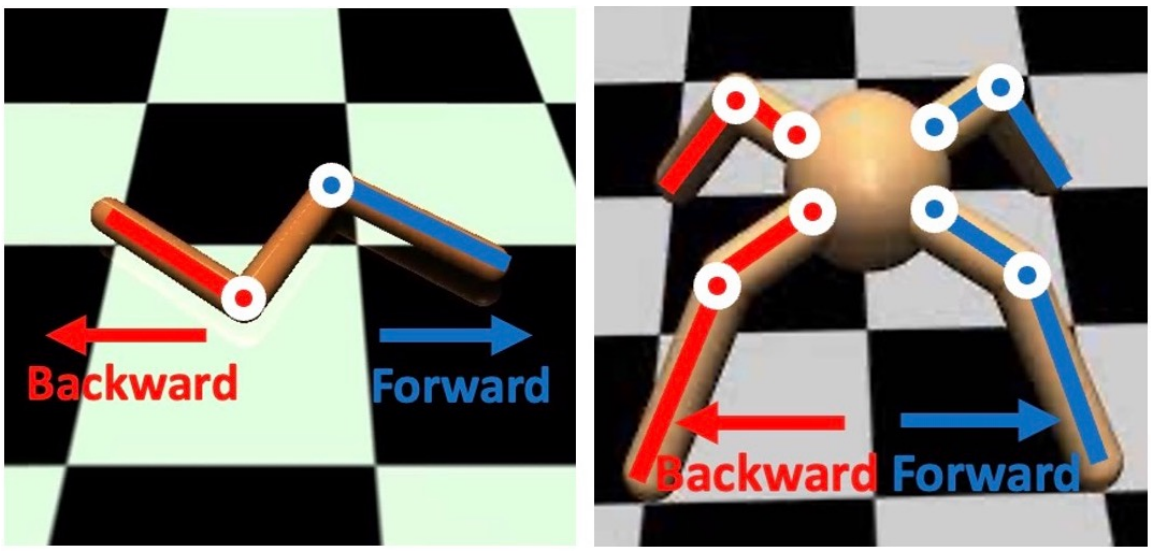


Figure 6-1: Mujoco environments (left: swimmer; right: ant), where the blue part of the body is controlled by the ego-agent and the red part of the body is controlled by the opponent agent. The white circles are the joints where agents can apply torques on. The ego-agent is rewarded for moving forward while the opponent agent is rewarded for moving backward.

2. Collect interaction experience $\mathcal{D}$ between $\mathbf{\Pi}_{\text{ego}}^{\text{ensemble}}$ and $\pi_{\text{oppo}}^{\text{exp}}$.
3. Unfreeze $\mathbf{\Pi}_{\text{ego}}^{\text{ensemble}}$.
4. Initialize second exploiter opponent $\pi^{*\text{exp}}_{\text{oppo}}$ from $\pi_{\text{oppo}}^{\text{exp}}$.
5. Alternating for K_3 iterations, between one-step safe adaptation for $\mathbf{\Pi}_{\text{ego}}^{\text{ensemble}}$ to adapt to $\pi_{\text{oppo}}^{\text{exp}}$ via Algorithm 4 and one-step training of the second exploiter opponent $\pi^{*\text{exp}}_{\text{oppo}}$ to exploit $\mathbf{\Pi}_{\text{ego}}^{\text{ensemble}}$ via Algorithm 5.

During the off-line training phase, we alternate between ensemble training $\mathbf{\Pi}_{\text{ego}}^{\text{ensemble}}$ and $\mathbf{\Pi}_{\text{oppo}}^{\text{ensemble}}$, and training of exploiter opponent $\pi_{\text{oppo}}^{\text{exp}}$ to mitigate the well-known problem of training imbalance [21, 131] in competitive/adversarial training. The additional training of the exploiter opponent ensures that the exploiter is sufficiently trained to exploit the ego-agent, which motivates the ego-agent to adapt to this exploiter in the adaptation phase. During the whole off-line training phase, the ego-agent does not collect experiences against the exploiter opponent $\pi_{\text{oppo}}^{\text{exp}}$. As a result, this exploiter is a previously unseen opponent from the ego-agent's perspective.

During the on-line adaptation phase, the ego-agent policy ensemble $\mathbf{\Pi}_{\text{ego}}^{\text{ensemble}}$

Algorithm 5 Train exploiter opponent

Require: Ego-agent ensemble $\mathbf{\Pi}_{\text{ego}}^{\text{ensemble}}$, exploiter opponent policy $\pi_{\text{oppo}}^{\text{exp}}$, number of training iterations K

1: freeze $\mathbf{\Pi}_{\text{ego}}^{\text{ensemble}}$
2: **for** k = 1:K **do**
3: Train $\pi_{\text{oppo}}^{\text{exp}}$ against $\mathbf{\Pi}_{\text{ego}}^{\text{ensemble}}$ by gradient decent on $\mathbb{L}_{\text{RL}}(\mathbf{\Pi}_{\text{ego}}^{\text{ensemble}}, \pi_{\text{oppo}}^{\text{exp}})$
4: **end for**
5: Unfreeze $\mathbf{\Pi}_{\text{ego}}^{\text{ensemble}}$
6: **return** $\pi_{\text{oppo}}^{\text{exp}}$

adapts to the exploiter opponent $\pi_{\text{oppo}}^{\text{exp}}$ given a fixed size interaction experience $\mathcal{D}$, with regularization from $\mathbf{\Pi}_{\text{oppo}}^{\text{ensemble}}$. Concurrently, the second exploiter $\pi^{*\text{exp}}_{\text{oppo}}$ is trained to exploit the $\mathbf{\Pi}_{\text{ego}}^{\text{ensemble}}$. As a result, the reward against the first exploiter opponent $\pi_{\text{oppo}}^{\text{exp}}$ during the adaptation phase measures the capability of adaptation against a stationary opponent, while the reward against the second exploiter opponent $\pi^{*\text{exp}}_{\text{oppo}}$ measures robustness/safety against an evolving adversarial exploiter.

Each agent uses a stochastic policy with Gaussian distribution parameterized by a feedforward network with two fully-connected hidden layers, each with 128 hidden units followed by ReLU activate layer. A Pytorch implementation[1] of Soft Actor-Critic [132] (SAC) with dual critic networks and automatic tuning of the entropy parameter is used to train the ensemble networks and the two exploiter opponents in both the off-line training phase and the on-line adaptation phase. The replay buffer size is one million for the off-line training phase and half a million for the on-line adaptation phase to save memory. The adaptation implementation is modified from the GAIL implementation in PyTorch-RL[2] with Proximal Policy Optimization [133] (PPO) for training $\hat{\pi}_{\text{oppo}}$, with PPO rollout batch size of 1000, mini-batch size of 128, and 10 gradient updates per PPO batch. The learning rate is 0.001 for both training and adaptation. In both environments, the number of steps per episode is fixed at 500. Each agent can observe the joints/bodies position and velocity of its own and its opponent's, so the agents have full observability.

In the off-line training phase, we use $K_1 = 10000$ iterations (episodes), and $K_2 =$

[1] https://github.com/p-christ/Deep-Reinforcement-Learning-Algorithms-with-PyTorch
[2] https://github.com/Khrylx/PyTorch-RL

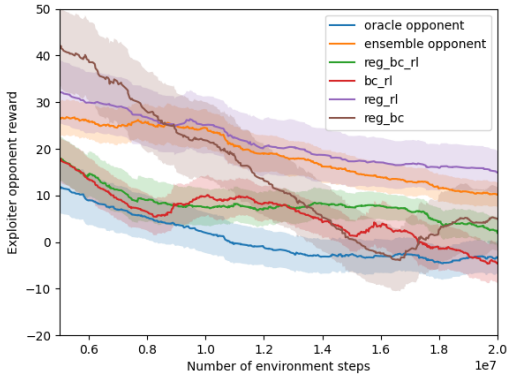
5000 iterations in the swimmer environment and $K_2 = 2000$ for the ant environment. In the on-line adaptation phase, we use different K_3 for the two environments until the average rewards become steady. We collect 10 episodes of interaction experience for adaptation, which corresponds to $|\mathcal{D}| = 5000$ environment steps. We tune the hyper-parameters λ_1 and λ_2 independently for each environment. We select from the following values: $\{0.1, 0.5, 1.0, 5.0\}$, and manually choose the best one by looking at the adaptation rewards against both of the two exploiter opponents. The selected hyper-parameters are: $\lambda_1 = 1.0, \lambda_2 = 1.0$ for the swimmer environment and $\lambda_1 = 0.1, \lambda_2 = 1.0$ for the ant environment.

6.4.2 Results

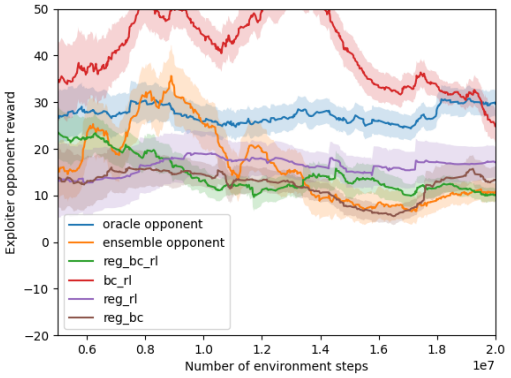
The exploiter opponent rewards during the adaptation phase are shown in Fig. 6-2, which includes the following settings:

1. oracle opponent, where the ego-agent is trained against the first exploiter directly. This setting is unrealistic since the first exploiter's policy is unknown to the ego-agent;
2. ensemble opponent, where the ego-agent is trained against the opponent ensemble policy generated from the off-line training phase;
3. reg_bc_rl: our proposed approach including all the three terms in Eq. 6.3;
4. bc_rl: ablation of our approach without the ensemble regularization term;
5. reg_rl: ablation of our approach without the behavior cloning term;
6. reg_bc: ablation of our approach without the RL term.

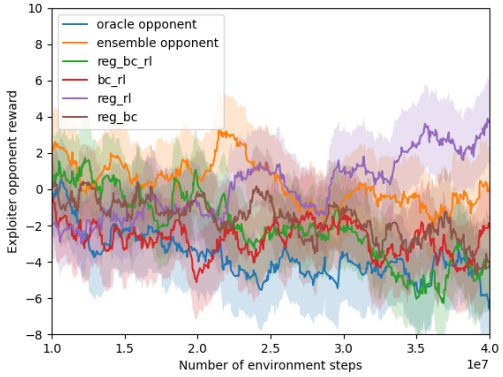
Fig. 6-2a shows that the oracle opponent setting (reference) achieves the best adaptation against the first exploiter opponent, while all the settings with the behavior cloning loss achieve comparable adaptation performance as the reference. For all the settings, the opponent reward decreases due to the fact that the opponent policy is fixed but the ego-agent policy is updating. However, there remains a gap between those settings without interaction experience with the first exploiter opponent (ensemble opponent and reg_rl) and the other settings.



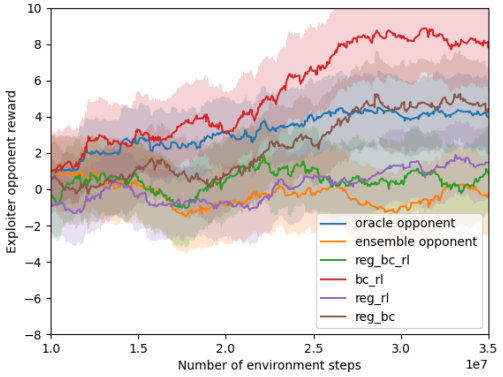
(a) Swimmer adaptation against first exploiter



(b) Swimmer adaptation against second exploiter



(c) Ant adaptation against first exploiter



(d) Ant adaptation against second exploiter

Figure 6-2: Exploiter opponent rewards during the adaptation phase. Lower opponent reward against the first (6-2a, 6-2c) and the second (6-2b, 6-2d) exploiter indicates better adaptation and safety of the ego-agent policy, respectively. Our proposed approach (reg_bc_rl) achieves near-optimal performance (w.r.t. the references) at both adaptation and safety, while the references (oracle opponent and ensemble opponent) achieve good performance only at one of these two metrics, but poor performance at the other.

Fig. 6-2b shows that those settings without the ensemble regularization term (oracle opponent and bc_rl) are unable to achieve robustness against the second exploiter opponent which actively exploits the ego-agent's policy. Combining Fig. 6-2a and 6-2b, we can conclude that our approach (reg_bc_rl) strikes a better trade-off between adaptation and safety, compared with the two reference approaches (oracle opponent: good adaptation but poor robustness; ensemble opponent: good robustness but poor adaptation).

Fig. 6-2c and 6-2d show consistent results as those in Fig. 6-2a and 6-2b: the

ego-agent's policies that are adapting to the first exploiter opponent are also more susceptible to be exploited by the second exploiter opponent. Besides, Fig. 6-2d shows that the setting without the RL loss (reg_bc) is also exploited after the second exploiter is sufficiently trained. Our conjectured reason for this observation is that the RL term enables the estimated opponent policy $\hat{\pi}_{\text{oppo}}$ to discover the weakness of the ego-agent's policy, which also helps reduce the ego-agent's exploitability because the ego-agent is trained against the estimated opponent policy.

To quantitatively measure the adaptation and robustness of different settings, we calculate the area under curve (AUC) metrics of both the first exploiter's (measures adaptation) and the second exploiter's (measures robustness) reward curves normalized by the two reference settings (oracle opponent and ensemble opponent), as shown in Table 6.1. We calculate the AUC using reward curves from steps 1e7 to 2e7 for the swimmer environment and from steps 1e7 to 3.5e7 for the ant environment. These results are consistent with our hypothesis that learning from the interaction experience with the opponent enables adaptation, but without regularization from the ensemble policy, this adaptation could be highly exploitable. The regularization term is effective for achieving safe adaptation. As a result, our approach achieves the best overall metric which combines adaptation and robustness.

From Table 6.1, we can also see that the adaptation metric and the robustness metric tend to be negatively correlated. To further analyze the relationship between adaptation (exploitation) and robustness (exploitability), we show the normalized area between curves (ABC) in Table 6.2, which is the gap between the reward against the second exploiter opponent and the first exploiter opponent. Lower ABC indicates less sensitivity to opponent exploitation. This result, together with the result shown in Table 6.1, verifies the well-known trade-off between exploitation and exploitability [23]: the settings with both the ensemble regularization term and the behavior cloning terms (reg_bc_rl and reg_bc) are slightly more exploitable than their counterpart without the behavior cloning term (reg_rl), which is an inevitable consequence of exploiting the interaction experience against the first exploiter opponent.

Table 6.1: Adaptation and robustness metrics of different settings, where the oracle opponent and the ensemble opponent settings are taken as references. The best two settings are highlighted.

Swimmer

Settings/Metrics	Adaptation	Robustness	Overall (A+R)
Oracle opponent	1.0	0.0	1.0
Ensemble opponent	0.0	1.0	1.0
Reg_bc_rl	0.50	**1.01**	**1.51**
Bc_rl	**0.67**	-0.92	-0.25
Reg_rl	-0.16	0.69	0.53
Reg_bc	**0.52**	**1.07**	**1.59**

Ant

Settings/Metrics	Adaptation	Robustness	Overall (A+R)
Oracle opponent	1.0	0.0	1.0
Ensemble opponent	0.0	1.0	1.0
Reg_bc_rl	**0.60**	**0.79**	**1.39**
Bc_rl	**0.76**	-0.60	0.16
Reg_rl	0.06	**0.89**	**0.95**
Reg_bc	0.48	0.25	0.73

Table 6.2: Normalized ABC (area between curves) between the second exploiter reward and the first exploiter reward. Lower ABC score indicates that the ego-agent's policy is less sensitive to opponent exploitation.

Swimmer

Settings	Normalized ABC
Oracle opponent	1.0
Ensemble opponent	0.0
Reg_bc_rl	0.27
Bc_rl	1.23
Reg_rl	**0.05**
Reg_bc	0.25

Ant

Settings	Normalized ABC
Oracle opponent	1.0
Ensemble opponent	0.0
Reg_bc_rl	0.37
Bc_rl	1.07
Reg_rl	**0.11**
Reg_bc	0.58

6.5 Summary

This work investigates safe adaptation which is an important problem in competitive MARL. In contrast to the widely-studied fast adaptation problem, our focus is on maintaining low exploitability during the adaptation. Our key innovation is the derivation of a novel ensemble regularization term from a Bayesian formulation of the MARL objective function. We show empirically that our proposed approach is effective both at adaptation to a previously unseen opponent given experience from a few interaction episodes and at maintaining low exploitability against an adversarial opponent that actively exploits the weakness of the ego-agent. Our ablation study and analysis reveal the effect of each term in our proposed loss function, as well as verify the well-known trade-off between exploitation and exploitability. Our work contributes to an important step towards building reliable intelligent robots that are able to operate safely in competitive multiagent scenarios against ever-changing adversarial opponents.

Chapter 7

Conclusion and Future Directions

7.1 Conclusions

This thesis focuses on robust and scalable MARL in adversarial scenarios. The approaches presented in this thesis answer the following questions: 1) How to robustly identify the intent of an opponent agent in adversarial scenarios? 2) How to scale up MARL for learning near-optimal joint policy in scenarios involving more than two, possibly many agents? 3) How to reduce the computation complexity of ensemble training to achieve efficient robust policy learning? 4) How to safely adapt to an opponent while maintaining low exploitability during the adaptation?

Chapter 3 presents our approach for addressing the problem of robust intent recognition against a previously unseen opponent, using a game-theoretic opponent modeling approach that captures sophisticated adversarial behaviors and a diversity-driven ensemble training approach that captures a wide spectrum of possible adversarial behaviors including deception. The experiment on an urban security game shows that our approach significantly improves the intent recognition accuracy, and as a result, enables robust decision-making against a deceptive adversarial opponent.

Chapter 4 presents our approach for learning near-optimal joint policy in scenarios with a large number of agents. We proposed a novel sparse attention mechanism, which enables selective attention to small subset of peer agents' information that is critical to the ego-agent's decision-making. This sparse attention, together with

a GNN architecture that exploits the permutation-invariance of those scenarios involving symmetry, leads to significantly improved sample-efficiency. As a result, we managed to scale up MARL to scenarios with many agents without significant compromise on optimality. We demonstrated learning of joint policy that achieves high performance with this new approach in scenarios with 20-30 agents, which increases the success rate from $\sim 20\%$ to $\sim 80\%$ in cooperative settings, and achieves a high winning rate in competitive settings compared against the state of the art baselines.

Chapter 5 presents our approach for computationally-efficient robust multiagent learning. We proposed a novel multi-tasking deep generative model for representing a policy distribution implicitly within a single network architecture, which preserves the policy diversity achieved from ensemble approaches while improving the sample-efficiency due to parameter-sharing. In contrast to the previous ensemble training approach that suffers from a trade-off between parameter-efficiency and diversity, our new approach achieves both parameter-efficiency and diversity. Experiment results on both board game domains and robotic domains show that our approach learns the most robust policy with much less computation, compared with baseline ensemble training. With this new approach, we are able to learn robust policy within hours which would have taken weeks using previous ensemble training approaches.

Chapter 6 presents our approach for safe adaptation in adversarial scenarios against a previously unseen, possibly evolving, opponent. We proposed a novel Bayesian formulation of MARL and an ensemble-regularized opponent modeling approach. Training against this regularized opponent results in learning of robust ego-agent policy that stays close to an equilibrium policy distribution, which minimizes the exploitability. Meanwhile, the learned ego-agent policy also exploits a sub-optimal opponent. This new approach enables continuous adaption safely to new opponents. Experiment results show that our approach achieves both effective adaptation and high robustness against an actively exploiting adversary.

Collectively, these four approaches enable computationally-efficient learning of intelligent agents that can make reliable decisions based on the understanding of the intents of peer agents as well as adapting safely to newly-encountered agents.

7.2 Future Directions

We discuss a few future directions to extend our approaches.

7.2.1 Modeling Agent Intent with Continuous Intent Space and Approximate Inference

Our robust intent recognition adopts a belief-space approach, which relies on the exact Bayesian update of the belief over possible intent variable. This approach works well for problems that involve a few discrete intents. However, in some real-world applications, we might want to capture a continuous spectrum of agent's latent preferences. For example, in autonomous driving scenarios, most drivers should not be simply classified as conservative or aggressive, since there might not exist a clear boundary between these two driving styles. Instead, a more appropriate way to model the driving style is a continuous latent variable that might be interpreted as the degree of aggressiveness of the driving style. However, it is difficult for the current framework to learn this latent space and do inference over this continuous latent space. Variational inference is one approach for learning and inference over a latent space. There has been some initial work on using variational auto-encoder to represent agent policy [124]. This approach could be potentially extended for joint learning of a latent space of agent intent and a conditional policy conditioned on the intent variable in this latent space so as to minimize the sub-optimality introduced by hand-crafting a discrete intent space with a finite (possibly very few) number of intents.

7.2.2 Scalable Heterogeneous Multiagent Policy Learning with Heterogeneous GNN

One of the key techniques that leads to the high sample-efficiency of our sparse attentional GNN approach is exploiting the permutation-invariance through the GNN architecture. This technique, however, can only be applied to homogeneous multi-

agent system, while a lot of real-world multiagent scenarios involves heterogeneous agents. Extending this technique to scale up the joint policy learning within homogeneous multiagent system is therefore a very desirable capability. Heterogeneous graph attentional neural network [134, 135] is a natural extension of GNN to encode heterogeneity. In typical heterogeneous multiagent systems, although agents are heterogeneous in general, the sub-system of different agents might share some common property. For example, two robot arms with different numbers of joints and links are heterogeneous but the joints and links could be homogeneous. A heterogeneous GNN could capture the homogeneity within this heterogeneous system through node embedding and edge embedding to model the homogeneous property of and the heterogeneous semantics and structure of the sub-system so that sample-efficiency can be preserved through homogeneous sharing, without significant compromise on optimality which could be a result of blindly sharing between heterogeneous agents.

7.2.3 Diversity-aware Implicit Ensemble Training

Our current implicit ensemble training approach has the capacity of representing a diverse set of policies attributed to the deep generative model architecture. However, policy diversity is achieved in a passive way in the sense that there is no explicit loss function or mechanism to encourage the exploration of the policy space. Theoretical studies [8, 136] suggest that many real-world games exhibit a spinning top structure, which implies that behavioral diversity is critical for learning strong policy. Therefore, it is desirable to extend our implicit ensemble training approach with diversity-awareness. The first question is how to characterize the diversity of the policy distribution parameterized by an implicit ensemble. There are a few choices for quantifying the difference between two policies, for example, KL-divergence [126], total variance distance [127], and maximal mean discrepancy metric [128], but there is no clear answer to which one of these metrics, if at all, is appropriate for measuring diversity. More studies need to be done towards a better understanding of how to measure diversity, how to generate diversity-aware implicit ensemble and how to shape an implicit ensemble to induce diverse and strong policies.

7.2.4 Meta-learning for safe adaptation

Our safe adaptation approach addresses the problem of adapting to an opponent without increasing one's exploitability. It is desirable to enable safe and fast adaptation by combining our approach with meta-learning. However, as we have discussed in Section 6.1.1, the difficulty of applying meta-learning in adversarial settings is the lack of information on the distribution of the opponent policy, while our approach uses an equilibrium distribution to achieve robustness. A natural question to ask is whether there exists an equivalent equilibrium task distribution for robust meta-learning. There have been works on task-robust meta-learning [137], Bayesian meta-learning [138], adversarially-robust few-shot learning [139] which minimizes performance degradation due to shift in the task distribution in supervised learning domains. One interesting research problem MARL is how to characterize the sensitivity of the meta-learned policy performance with respect to adversarial exploitation from a previously unseen opponent and how to shape the task distribution to minimize this sensitivity metric for achieving safe and fast adaptation.

7.2.5 Extension to real-world multiagent systems

The techniques developed in the thesis have been demonstrated in simulations where the simulator provides an ideal environment with an unlimited amount of data for training and evaluation. One desirable future direction is to enable the application of the decision-making policy learned in simulation to real-world physical systems. The challenges in this setting include hard safety constraints and the distributional shift between simulation and the real world.

There are quite a few related research fields for ensuring safety in real-world multiagent decision-making, including safe RL with certifiably robustness [140], interpretable RL [141] and hybrid planning [142]. Safe RL focuses on respecting safety constraints during the learning and/or deployment processes while learning policies that maximize the expected returns [143]. In the context of learning decision-making for real-world physical systems, it is reasonable to require that a deep RL policy

leads to collision-free trajectories. This requirement can be certified by finding the forward-reachable set [140] following the deep RL policy and making sure that this set does not intersect with obstacles. However, in real-world scenarios, the environment involves other agents that are not under the control of the deep RL policy, and they are treated as dynamic obstacles. A high accuracy agent trajectory prediction module is required to check feasibility, which is itself a challenging prediction task.

Control and planning-based approaches have better theoretical guarantees of safety but do not scale as well as RL-based approaches in multiagent systems. One promising direction is to combine the advantages of both through a hybrid approach where RL is combined with an online look-ahead planner to check for safety [142]. This approach also results in better interpretability of the decision-making, which is important in safety-critical domains.

Distributional shift between simulation and the real world is another challenge for deploying the decision-making policy learned in simulation to real-world physical systems, where sensing/perception, system dynamics, and actuation can be the source of domain mismatches [144]. Transfer learning/domain adaptation [144] attempts to reduce this domain gap by pre-training on the source domain (simulation) and then fine-tuning on the target domain (real-world physical systems). This transfer can be achieved through matching the latent feature representation [145, 146], which can account for the domain mismatch of sensing/perception. Robust RL can explicitly take into account the uncertainty in sensing, system dynamics, and actuation during the training on the source domain. This uncertainty can be modeled as random perturbations [147] (a.k.a. domain randomization) or adversarial perturbations [148]. The policy learned through robust RL in simulation mitigates performance degradation when deployed in real-world physical systems. Combining transfer learning/domain adaptation and robust RL in a multiagent decision-making framework is a promising way to efficiently and effectively reduce the sim-to-real gap, which is another interesting future direction of this work.

Bibliography

[1] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[2] Noam Brown and Tuomas Sandholm. Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018.

[3] Adarsh Subbaswamy, Roy Adams, and Suchi Saria. Evaluating model robustness and stability to dataset shift. In *International Conference on Artificial Intelligence and Statistics*, pages 2611–2619. PMLR, 2021.

[4] Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615*, 2019.

[5] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

[6] Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, et al. Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. *arXiv preprint arXiv:1807.01281*, 2018.

[7] Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. *arXiv preprint arXiv:1710.03748*, 2017.

[8] Wojciech Marian Czarnecki, Gauthier Gidel, Brendan Tracey, Karl Tuyls, Shayegan Omidshafiei, David Balduzzi, and Max Jaderberg. Real world games look like spinning tops. *arXiv preprint arXiv:2004.09468*, 2020.

[9] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *arXiv preprint arXiv:1911.08265*, 2019.

[10] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. Mean field multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 5571–5580. PMLR, 2018.

[11] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. $\mathbf{RL}^2$: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[12] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[13] Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *arXiv preprint arXiv:2012.13490*, 2020.

[14] Jakob N Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. *arXiv preprint arXiv:1709.04326*, 2017.

[15] Maruan Al-Shedivat, Trapit Bansal, Yuri Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. *arXiv preprint arXiv:1710.03641*, 2017.

[16] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[17] Weilong Song, Guangming Xiong, and Huiyan Chen. Intention-aware autonomous driving decision-making in an uncontrolled intersection. *Mathematical Problems in Engineering*, 2016.

[18] Macheng Shen and Jonathan P How. Robust opponent modeling via adversarial ensemble reinforcement learning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 31, pages 578–587, 2021.

[19] Thanh H Nguyen, Yongzhao Wang, Arunesh Sinha, and Michael P Wellman. Deception in finitely repeated security games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2133–2140, 2019.

[20] Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615*, 2019.

[21] Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. *arXiv preprint arXiv:1710.03748*, 2017.

[22] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[23] Sam Ganzfried and Tuomas Sandholm. Safe opponent exploitation. *ACM Transactions on Economics and Computation (TEAC)*, 3(2):1–28, 2015.

[24] Macheng Shen and Jonathan P How. Active perception in adversarial scenarios using maximum entropy deep reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3384–3390. IEEE, 2019.

[25] Chuangchuang Sun, Macheng Shen, and Jonathan P How. Scaling up multiagent reinforcement learning for robotic systems: Learn an adaptive sparse communication graph. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11755–11762. IEEE, 2020.

[26] Macheng Shen and Jonathan P How. Implicit ensemble training for efficient and robust multiagent reinforcement learning. *2021 International Conference on Machine Learning Workshop on Uncertainty and Robustness in Deep Learning (ICML-UDL)*, URL: `http://www.gatsby.ucl.ac.uk/~balaji/udl2021/accepted-papers/UDL2021-paper-019.pdf`, 2021.

[27] Macheng Shen and Jonathan P How. Safe adaptation in multiagent competition. *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, under review, URL: `https://arxiv.org/pdf/2203.07562.pdf`, 2022.

[28] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[29] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.

[30] Taksu Cheon and Azhar Iqbal. Bayesian nash equilibria and bell inequalities. *Journal of the Physical Society of Japan*, 77(2):024801, 2008.

[31] Max Jaderberg, Wojciech Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Castaneda, Charles Beattie, Neil Rabinowitz, Ari Morcos, Avraham Ruderman, et al. Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019.

[32] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. *arXiv preprint arXiv:1711.00832*, 2017.

[33] Eugene Vinitsky, Yuqing Du, Kanaad Parvate, Kathy Jang, Pieter Abbeel, and Alexandre Bayen. Robust reinforcement learning using adversarial populations. *arXiv preprint arXiv:2008.01825*, 2020.

[34] John F Nash et al. Equilibrium points in n-person games. *Proceedings of the national academy of sciences*, 36(1):48–49, 1950.

[35] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[36] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposium Series*, 2015.

[37] Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*, 2018.

[38] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisỳ, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

[39] OpenAI. Openai five. *OpenAI blog*, 2018.

[40] Alan J Lockett, Charles L Chen, and Risto Miikkulainen. Evolving explicit opponent models in game playing. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 2106–2113, 2007.

[41] He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé III. Opponent modeling in deep reinforcement learning. In *International conference on machine learning*, pages 1804–1813, 2016.

[42] David Carmel and Shaul Markovitch. Model-based learning of interaction strategies in multi-agent systems. *Journal of Experimental & Theoretical Artificial Intelligence*, 10(3):309–332, 1998.

[43] Doran Chakraborty and Peter Stone. Multiagent learning in the presence of memory-bounded agents. *Autonomous agents and multi-agent systems*, 28(2):182–213, 2014.

[44] Nolan Bard, Michael Johanson, Neil Burch, and Michael Bowling. Online implicit agent modelling. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 255–262, 2013.

[45] Ronald V Bjarnason and Todd S Peterson. Multi-agent learning via implicit opponent modeling. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 2, pages 1534–1539. IEEE, 2002.

[46] Michael Fagan and Pádraig Cunningham. Case-based plan recognition in computer games. In *International Conference on Case-Based Reasoning*, pages 161–170. Springer, 2003.

[47] Shirin Sohrabi, Anton V Riabov, and Octavian Udrea. Plan recognition as planning revisited. In *IJCAI*, pages 3258–3264, 2016.

[48] Jonathan Rubin and Ian Watson. Implicit opponent modelling via dynamic case-base selection. In *Workshop on case-based reasoning for computer games at the 19th international conference on case-based reasoning*, pages 63–71, 2011.

[49] Jack Serrino, Max Kleiman-Weiner, David C Parkes, and Josh Tenenbaum. Finding friend and foe in multi-agent games. In *Advances in Neural Information Processing Systems*, pages 1249–1259, 2019.

[50] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

[51] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.

[52] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, 30(9):2805–2824, 2019.

[53] Adam Gleave, Michael Dennis, Neel Kant, Cody Wild, Sergey Levine, and Stuart Russell. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615*, 2019.

[54] Inaam Ilahi, Muhammad Usama, Junaid Qadir, Muhammad Umar Janjua, Ala Al-Fuqaha, Dinh Thai Hoang, and Dusit Niyato. Challenges and countermeasures for adversarial attacks on deep reinforcement learning. *arXiv preprint arXiv:2001.09684*, 2020.

[55] Yi Han, Benjamin IP Rubinstein, Tamas Abraham, Tansu Alpcan, Olivier De Vel, Sarah Erfani, David Hubczenko, Christopher Leckie, and Paul Montague. Reinforcement learning for autonomous defence in software-defined networking. In *International Conference on Decision and Game Theory for Security*, pages 145–165. Springer, 2018.

[56] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

[57] Vahid Behzadan and Arslan Munir. Vulnerability of deep reinforcement learning to policy induction attacks. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 262–275. Springer, 2017.

[58] Tong Chen, Wenjia Niu, Yingxiao Xiang, Xiaoxuan Bai, Jiqiang Liu, Zhen Han, and Gang Li. Gradient band-based adversarial training for generalized attack immunity of a3c path finding. *arXiv preprint arXiv:1807.06752*, 2018.

[59] Adam Gleave, Michael Dennis, Neel Kant, Cody Wild, Sergey Levine, and Stuart Russell. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615*, 2019.

[60] Linan Huang and Q. Zhu. Dynamic bayesian games for adversarial and defensive cyber deception. *ArXiv*, abs/1809.02013, 2018.

[61] Shauharda Khadka, Somdeb Majumdar, Santiago Miret, Evren Tumer, Tarek Nassar, Zach Dwiel, Yinyin Liu, and Kagan Tumer. Collaborative evolutionary reinforcement learning. *arXiv preprint arXiv:1905.00976*, 2019.

[62] Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 4496–4506, 2017.

[63] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[64] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[65] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[66] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[67] Maximilian Hüttenrauch, Adrian Šošić, and Gerhard Neumann. Guided deep reinforcement learning for swarm systems. *arXiv preprint arXiv:1709.06011*, 2017.

[68] Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4):819–840, 2002.

[69] Lucian Bu, Robert Babu, Bart De Schutter, et al. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

[70] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pages 6379–6390, 2017.

[71] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2085–2087. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[72] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1803.11485*, 2018.

[73] Akshat Agarwal, Sumit Kumar, and Katia Sycara. Learning transferable cooperative behavior in multi-agent teams. *arXiv preprint arXiv:1906.01202*, 2019.

[74] Arbaaz Khan, Ekaterina Tolstaya, Alejandro Ribeiro, and Vijay Kumar. Graph policy gradients for large scale robot control. *arXiv preprint arXiv:1907.03822*, 2019.

[75] Jiechuan Jiang, Chen Dun, and Zongqing Lu. Graph convolutional reinforcement learning for multi-agent cooperation. *arXiv preprint arXiv:1810.09202*, 2(3), 2018.

[76] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[77] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. In *Advances in neural information processing systems*, pages 577–585, 2015.

[78] Andre Martins and Ramon Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*, pages 1614–1623, 2016.

[79] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018.

[80] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. Mean field multi-agent reinforcement learning. *arXiv preprint arXiv:1802.05438*, 2018.

[81] Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. *arXiv preprint arXiv:1810.02912*, 2018.

[82] Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Michael Rabbat, and Joelle Pineau. Tarmac: Targeted multi-agent communication. *arXiv preprint arXiv:1810.11187*, 2018.

[83] Vlad Niculae and Mathieu Blondel. A regularized framework for sparse and structured neural attention. In *Advances in Neural Information Processing Systems*, pages 3338–3348, 2017.

[84] Mathieu Blondel, André FT Martins, and Vlad Niculae. Learning classifiers with fenchel-young losses: Generalized entropies, margins, and algorithms. *arXiv preprint arXiv:1805.09717*, 2018.

[85] Anirban Laha, Saneem Ahmed Chemmengath, Priyanka Agrawal, Mitesh Khapra, Karthik Sankaranarayanan, and Harish G Ramaswamy. On controllable sparse alternatives to softmax. In *Advances in Neural Information Processing Systems*, pages 6422–6432, 2018.

[86] Gonçalo M Correia, Vlad Niculae, and André FT Martins. Adaptively sparse transformers. *arXiv preprint arXiv:1909.00015*, 2019.

[87] Andres M Kowalski, Raul D Rossignoli, and Evaldo MF Curado. *Concepts and recent advances in Generalized Information Measures and Statistics*. Bentham Science Publishers, 2013.

[88] Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. Incorporating functional knowledge in neural networks. *Journal of Machine Learning Research*, 10(Jun):1239–1262, 2009.

[89] Fritz Heider. Attitudes and cognitive organization. *The Journal of psychology*, 21(1):107–112, 1946.

[90] David Easley, Jon Kleinberg, et al. Networks, crowds, and markets: Reasoning about a highly connected world. *Significance*, 9:43–44, 2012.

[91] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[92] Erol Şahin. Swarm robotics: From sources of inspiration to domains of application. In *International workshop on swarm robotics*, pages 10–20. Springer, 2004.

[93] Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[94] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

[95] Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. *arXiv preprint arXiv:2003.13661*, 2020.

[96] Kaiqing Zhang, Tao Sun, Yunzhe Tao, Sahika Genc, Sunil Mallya, and Tamer Basar. Robust multi-agent reinforcement learning with model uncertainty. *Advances in Neural Information Processing Systems*, 33, 2020.

[97] Shihui Li, Yi Wu, Xinyue Cui, Honghua Dong, Fei Fang, and Stuart Russell. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4213–4220, 2019.

[98] Carlo D'Eramo, Davide Tateo, Andrea Bonarini, Marcello Restelli, and Jan Peters. Sharing knowledge in multi-task deep reinforcement learning. In *International Conference on Learning Representations*, 2019.

[99] Wenlong Huang, Igor Mordatch, and Deepak Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In *International Conference on Machine Learning*, pages 4455–4464. PMLR, 2020.

[100] Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2169–2176. IEEE, 2017.

[101] Tuomas Haarnoja, Kristian Hartikainen, Pieter Abbeel, and Sergey Levine. Latent space policies for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pages 1851–1860. PMLR, 2018.

[102] Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *arXiv preprint arXiv:1907.00953*, 2019.

[103] Janith C Petangoda, Sergio Pascual-Diaz, Vincent Adam, Peter Vrancx, and Jordi Grau-Moya. Disentangled skill embeddings for reinforcement learning. *arXiv preprint arXiv:1906.09223*, 2019.

[104] Justin K Terry, Benjamin Black, Ananth Hari, Luis Santos, Clemens Dieffendahl, Niall L Williams, Yashas Lokesh, Caroline Horsch, and Praveen

Ravi. Pettingzoo: Gym for multi-agent reinforcement learning. *arXiv preprint arXiv:2009.14471*, 2020.

[105] Daochen Zha, Kwei-Herng Lai, Yuanpu Cao, Songyi Huang, Ruzhe Wei, Junyu Guo, and Xia Hu. Rlcard: A toolkit for reinforcement learning in card games. *arXiv preprint arXiv:1910.04376*, 2019.

[106] Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 2961–2970. PMLR, 2019.

[107] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. Rllib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning*, pages 3053–3062. PMLR, 2018.

[108] Wojciech Marian Czarnecki, Gauthier Gidel, Brendan Tracey, Karl Tuyls, Shayegan Omidshafiei, David Balduzzi, and Max Jaderberg. Real world games look like spinning tops. *arXiv preprint arXiv:2004.09468*, 2020.

[109] David Balduzzi, Karl Tuyls, Julien Pérolat, and Thore Graepel. Re-evaluating evaluation. *CoRR*, abs/1806.02643, 2018.

[110] Karl Tuyls, Julien Perolat, Marc Lanctot, Edward Hughes, Richard Everett, Joel Z Leibo, Csaba Szepesvári, and Thore Graepel. Bounds and dynamics for empirical game theoretic analysis. *Autonomous Agents and Multi-Agent Systems*, 34(1):1–30, 2020.

[111] Anusha Nagabandi, Chelsea Finn, and Sergey Levine. Deep online learning via meta-learning: Continual adaptation for model-based rl. *arXiv preprint arXiv:1812.07671*, 2018.

[112] Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018.

[113] Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2(3):4, 2018.

[114] Dong Ki Kim, Miao Liu, Matthew D Riemer, Chuangchuang Sun, Marwa Abdulhai, Golnaz Habibi, Sebastian Lopez-Cot, Gerald Tesauro, and Jonathan How. A policy gradient algorithm for learning to learn in multiagent reinforcement learning. In *International Conference on Machine Learning*, pages 5541–5550. PMLR, 2021.

[115] Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *arXiv preprint arXiv:2012.13490*, 2020.

[116] Charles A Holt and Alvin E Roth. The nash equilibrium: A perspective. *Proceedings of the National Academy of Sciences*, 101(12):3999–4002, 2004.

[117] Ross Cressman, Christopher Ansell, and Ken Binmore. *Evolutionary dynamics and extensive form games*, volume 5. MIT Press, 2003.

[118] Mingyang Liu, Chengjie Wu, Qihan Liu, Yansen Jing, Jun Yang, Pingzhong Tang, and Chongjie Zhang. Safe opponent-exploitation subgame refinement, 2022.

[119] Noam Brown and Tuomas Sandholm. Safe and nested subgame solving for imperfect-information games. *Advances in neural information processing systems*, 30, 2017.

[120] Noam Brown and Tuomas Sandholm. Safe and nested subgame solving for imperfect-information games. *Advances in neural information processing systems*, 30, 2017.

[121] He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé III. Opponent modeling in deep reinforcement learning. In *International conference on machine learning*, pages 1804–1813. PMLR, 2016.

[122] Roberta Raileanu, Emily Denton, Arthur Szlam, and Rob Fergus. Modeling others using oneself in multi-agent reinforcement learning. In *International conference on machine learning*, pages 4257–4266. PMLR, 2018.

[123] Aditya Grover, Maruan Al-Shedivat, Jayesh Gupta, Yuri Burda, and Harrison Edwards. Learning policy representations in multiagent systems. In *International conference on machine learning*, pages 1802–1811. PMLR, 2018.

[124] Georgios Papoudakis and Stefano V Albrecht. Variational autoencoders for opponent modeling in multi-agent systems. *arXiv preprint arXiv:2001.10829*, 2020.

[125] Shihui Li, Yi Wu, Xinyue Cui, Honghua Dong, Fei Fang, and Stuart Russell. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4213–4220, 2019.

[126] Chuheng Zhang, Yuanqi Li, and Jian Li. Policy search by target distribution learning for continuous control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6770–6777, 2020.

[127] James Queeney, Ioannis Paschalidis, and Christos Cassandras. Generalized proximal policy optimization with sample reuse. *Advances in Neural Information Processing Systems*, 34, 2021.

[128] Muhammad A Masood and Finale Doshi-Velez. Diversity-inducing policy gradient: Using maximum mean discrepancy to find a set of diverse policies. *arXiv preprint arXiv:1906.00088*, 2019.

[129] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

[130] Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. Facmac: Factored multi-agent centralised policy gradients. *Advances in Neural Information Processing Systems*, 34, 2021.

[131] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[132] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[133] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[134] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *The world wide web conference*, pages 2022–2032, 2019.

[135] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of The Web Conference 2020*, pages 2331–2341, 2020.

[136] Yaodong Yang, Jun Luo, Ying Wen, Oliver Slumbers, Daniel Graves, Haitham Bou Ammar, Jun Wang, and Matthew E Taylor. Diverse auto-curriculum is critical for successful real-world multiagent learning systems. *arXiv preprint arXiv:2102.07659*, 2021.

[137] Liam Collins, Aryan Mokhtari, and Sanjay Shakkottai. Task-robust model-agnostic meta-learning. *Advances in Neural Information Processing Systems*, 33:18860–18871, 2020.

[138] Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. *Advances in neural information processing systems*, 31, 2018.

[139] Micah Goldblum, Liam Fowl, and Tom Goldstein. Adversarially robust few-shot learning: A meta-learning approach. *Advances in Neural Information Processing Systems*, 33:17886–17895, 2020.

[140] Michael Everett, Bjorn Lutjens, and Jonathan P How. Certified adversarial robustness for deep reinforcement learning. *arXiv preprint arXiv:2004.06496*, 2020.

[141] Claire Glanois, Paul Weng, Matthieu Zimmer, Dong Li, Tianpei Yang, Jianye Hao, and Wulong Liu. A survey on interpretable reinforcement learning. *arXiv preprint arXiv:2112.13112*, 2021.

[142] Harshit Sikchi, Wenxuan Zhou, and David Held. Learning off-policy with online planning. In *Conference on Robot Learning*, pages 1622–1633. PMLR, 2022.

[143] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[144] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744. IEEE, 2020.

[145] Thomas Carr, Maria Chli, and George Vogiatzis. Domain adaptation for reinforcement learning on the atari. *arXiv preprint arXiv:1812.07452*, 2018.

[146] Jinwei Xing, Takashi Nagata, Kexin Chen, Xinyun Zou, Emre Neftci, and Jeffrey L Krichmar. Domain adaptation in reinforcement learning via latent unified state representation. *arXiv preprint arXiv:2102.05714*, 2021.

[147] Fabio Muratore, Fabio Ramos, Greg Turk, Wenhao Yu, Michael Gienger, and Jan Peters. Robot learning from randomized simulations: A review. *Frontiers in Robotics and AI*, 9, 2022.

[148] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, pages 2817–2826. PMLR, 2017.