The Dissertation Committee for Jiaxun Cui
certifies that this is the approved version of the following dissertation:

Communication and Generalization in Multi-Agent Learning

Committee:

Peter Stone, Supervisor

Yuke Zhu

Amy Zhang

Sandeep Chinchali

Yuandong Tian

Communication and Generalization in Multi-Agent Learning

by

Jiaxun Cui

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 2025

Acknowledgments

This dissertation would not have been possible without the support and contributions of many people. First and foremost, I would like to express my deepest gratitude to my advisor, Professor Peter Stone, for his unwavering support, constructive feedback, and encouragement throughout my Ph.D. journey. His vision, patience, and support have been instrumental in shaping both the depth of my research and my growth as a well-rounded researcher.

I am sincerely grateful to my dissertation committee: Prof. Amy Zhang, Prof. Yuke Zhu, Prof. Sandeep Chinchali, and Dr. Yuandong Tian, for their time, thoughtful feedback, and support in evaluating and refining my work. I especially thank Prof. Yuke Zhu and Dr. Yuandong Tian for their invaluable mentorship and technical guidance on research projects that contributed to this dissertation, as well as for the inspiration they provided during my graduate studies.

I have been fortunate to receive research guidance and mentorship through my internships, which have broadened my perspectives. I want to thank Haobo Fu and Jiechao Xiong at Tencent AI Labs for introducing me to game-theoretic reinforcement learning through the Mahjong game. I am grateful to mentors Xiaomeng Yang and Yuandong Tian at FAIR, whose guidance strengthened my multi-agent RL research by grounding it in a real-world problem. I would also like to thank my mentor at Facebook, Geng Ji, for providing the opportunity to develop a large-scale real-world application of RL. Finally, I am deeply appreciative of Jarrett Holtz and Alessandro Allievi at Bosch for their guidance and patience on the talking vehicles research.

My heartfelt thanks go to my collaborators, fellow Ph.D. students, labmates, and postdoctoral researchers at the Learning Agents Research Group (LARG), as well as to my co-organizers of the Reinforcement Learning Reading Group (RLRG). Their insightful discussions, camaraderie, and encouragement made challenging moments more manageable and successes more rewarding. I am especially grateful to Siddhant Agarwal, Dian Chen, Caleb Chuck, Cevahir Koprulu, Ishan Durugkar, Rolando Fernandez, Jiaheng Hu, Yuqian Jiang, Haresh Kanan, Brad Knox, Benjamin Lee, Geunbae Lee, Hsien-Hsin S. Lee, Alexander Levine, Bo Liu, Shuijing Liu, Mulong Luo, William Macke, Reuth Mirsky, Michael Munje, Carl Qi, Muhammad Arrasy Rahman, Max Rudolph, Harshit Sikchi, Chang Shi, Steven Shi, Shahaf Shperberg, Edward Suh, Yoonchang Sung, Chen Tang, Daniel Urieli, Zizhao Wang, Caroline Wang, Zhihan Wang, Garrett Warnell, Harel Yedidsion, Wenjie Xiong, Haoran Xu, Zifan Xu, Lingyun Xiao, Yulin Zhang, Yifeng Zhu, and many others. Moreover, I deeply appreciate the undergraduate and master's students I have had the privilege of mentoring, whose curiosity, enthusiasm, and energy have been a constant source of inspiration. Organizing the RLRG for three years has been one of the highlights of my time in the lab, and I am grateful to Prof. Peter Stone and Prof. Amy Zhang for their generous sponsorship. I also extend my sincere thanks to Megan Booth, Melanie Gulick, Thomas Atchity, and other university staff for their outstanding assistance with academic advising and administrative support.

Finally, I am deeply indebted to my family and friends for their unconditional love, patience, and encouragement. Their belief in me has been my anchor throughout this journey. I thank my parents, Zhendong Cui and Rong Lu, and my grandmother, Xingmei Zou, for providing a strong foundation at home that enabled me to pursue my dreams. I deeply thank my friends in Austin, as well as many other friends beyond Austin, whose encouragement and companionship have sustained me along the way.

Jiaxun Cui

August 2025

Abstract

Communication and Generalization in Multi-Agent Learning

Jiaxun Cui, PhD
The University of Texas at Austin, 2025

SUPERVISOR: Peter Stone

Multi-agent learning aims to allow artificial intelligence (AI) agents to learn from interactions with other agents in an environment. However, as AI increasingly integrates into real-world systems, significant challenges arise in how to robustly interact with and communicate with a variety of other agents, particularly in complex environments such as autonomous driving, where humans and AI agents coexist. This dissertation research investigates **how agents can be trained to effectively communicate with and generalize to diverse partners (including humans) in simulated real-world scenarios.**

Towards addressing this challenge, this dissertation explores three key dimensions: **(1)** learning communication-supporting representations that facilitate coordination, **(2)** developing multi-agent policies that generalize to new teammates or opponents, and **(3)** learning to collaborate with human-like agents or to use human language. This dissertation makes novel contributions along each dimension.

First, the dissertation presents **COOPERNAUT**, a framework that learns compact, transmittable representations from local observations to support communication among autonomous vehicles under bandwidth constraints. It also introduces **LLM+DEBRIEF**, which enables embodied agents to coordinate in driving scenarios by generating and interpreting natural language messages, paving the way for human-compatible agent communication.

Second, it introduces **MACTA**, a reinforcement learning and game-theoretic training framework that produces robust policies capable of generalizing to unseen and

adaptive opponents. In addition, **L-BRDiv** is introduced as a teammate generation strategy that promotes behavioral diversity during training, improving generalization and performance in ad hoc teamwork settings.

Third, the dissertation investigates **mixed-autonomy traffic coordination** through decentralized training in environments with both human-proxy and AI agents. Empirical results demonstrate that even a small number of trained autonomous vehicles can collaborate effectively to influence human behavior and improve overall traffic efficiency without requiring centralized control.

Collectively, these contributions advance multi-agent AI by unifying communication, generalization, and human-AI collaboration. Evaluated in both toy domains and realistic simulated environments, primarily focusing on autonomous driving and hardware security, the work demonstrates how agents can adapt to novel partners and communicate effectively in human-interpretable ways.

Table of Contents

List of Tables

List of Figures

Part I

Background

Chapter 1

Introduction

Future AI is not alone.

Artificial Intelligence (AI) has played and continues to play an important role in everyday life. While many capabilities have been intensively developed in settings modeled as single-agent problems (e.g., chatting, playing Atari games, generating artistic pieces), they are relatively underexplored in real-world scenarios where decision-making AI agents coexist with humans and other agents. For instance, there is a potential for autonomous vehicles to be able to work together to improve traffic safety and cooperate with human drivers to reduce traffic congestion.

The learning dynamics in multi-agent decision-making scenarios present a significant challenge. When other agents are viewed as part of the environment, the environment becomes non-stationary during learning, since other agents' policies may change over time. Game theory provides analytical tools to solve multi-agent games through equilibrium policies, but struggles with the computational complexity of large-scale games, and does not directly address challenges like processing high-dimensional multi-modal sensory inputs (in various formats, such as videos, sounds, HD maps, metadata, etc.) for policies. Multi-agent Reinforcement Learning (MARL), which optimizes expected individual or team return using Reinforcement Learning (RL), offers more flexibility than classical game-theoretic computation for handling

complex inputs and long-horizon decision-making. Notable techniques in MARL, such as population-based training, empirical game-theoretic analysis, and centralized training decentralized execution (CTDE), have achieved significant success in games like Go (Silver et al., 2016), StarCraft (Vinyals et al., 2019), and Diplomacy (Bakhtin et al., 2022a).

Despite MARL's ability to formulate strong policies that yield high expected returns, two specific aspects remain relatively less explored in the context of deep multi-agent reinforcement learning.

The first aspect is the generalization of a policy to emergent teammates or opponents that do not appear during the training phase, herein referred to as unseen agents. The cooperative counterpart of the problem is referred to as Ad-Hoc Teamwork (AHT) (Stone et al., 2010) in literature. This concern is especially crucial since AI may frequently encounter novel partners, such as humans or other AI agents in scenarios like autonomous driving. These partners might exhibit diverse policy styles or coordination conventions. Typically, the generalizability of a policy is often pursued either by computing equilibrium-based policies (e.g., Nash or correlated equilibria) that are robust to a range of opponent strategies, or by training with a diverse population of agents.

The second aspect involves communication in multi-agent learning. While many training frameworks focus on either fully centralized information sharing or fully decentralized observation, fewer explicitly address partially observable settings with constrained communication. However, real-world scenarios often permit communication, albeit with limitations in bandwidth. These restrictions complicate multi-agent learning, raising critical questions about when and whom to communicate with, what information to share, and how to respond to received information. Among communication protocols, natural language is often the most suitable for interacting with humans, as it is expressive, well-structured, and widely understood across languages with translation, though safety-critical domains may prefer standardized signals.

Where these two dimensions intersect, formulating a universally applicable policy that accommodates a range of policies, along with communication mechanisms, holds practical relevance in real-life applications such as autonomous driving. For instance, a vehicle with brake failure might broadcast a warning to nearby vehicles. An autonomous vehicle could parse this structured message directly, while a human-driven car might require a human-interpretable format such as natural language or standardized visual or auditory alerts.

With this motivation in mind, this dissertation navigates the complexities of multi-agent learning, addressing the critical question:

How can a decision-making agent learn to efficiently communicate with and create generalizable policies for novel AI or human teammates or opponents in simulated real-world scenarios?

This dissertation explores and answers the question along the following three dimensions:

A. **Communication-Supporting Representations.** In the realm of decentralized multi-agent systems, communication is a pivotal tool that facilitates the exchange of information, coordination of actions, negotiations, and the making of collective decisions among agents. The interchange of messages involves answering questions related to when, with whom, and what to communicate, and managing the tangible real-world constraints of limited bandwidth. This dissertation focuses on how to construct transmittable messages through a learned representation space in inter-agent decision-making scenarios.

B. **Multi-Agent Policy Generalization.** Multi-agent policy generalization pertains to interacting with unseen partners or opponents without the need for additional fine-tuning based on these interactions. Traditionally, this advantage has been derived from game theory by scrutinizing an equilibrium policy that ensures some conservative game values. However, real-world applications such as autonomous driving render traditional game analysis infeasible. With

the previous success of the empirical game-theoretic framework (Vinyals et al., 2019; Lanctot et al., 2017), which performs strategic reasoning through interleaved simulation and game-theoretic analysis, such robustness becomes approachable by ensuring a high degree of policy *diversity* during the training phase, thus preparing the agents to handle a broad spectrum of policy styles and coordination conventions.

C. **Collaborating with Human-Like Agents.** Multi-agent systems in real-world settings, such as mixed-autonomy traffic, are inherently dynamic: agents may enter or leave the environment at any time, and control may be exercised by either humans or autonomous agents. This dissertation investigates how reinforcement learning agents can collaborate effectively in such open systems, adapting to diverse partners, including both AI agents and humans, without requiring centralized coordination. In a separate line of inquiry, this work also examines the use of natural language as a medium for communication in multi-agent settings. By leveraging large language models as agents capable of producing and interpreting human language, we enable autonomous agents to convey intentions and share key observations in a human-interpretable form. This direction facilitates direct interaction with human users or their surrogates, offering a scalable path toward human-AI collaboration without relying on extensive human demonstration data.

By exploring the first two directions, we develop foundational methods for decision-making agents to communicate with and adapt to a variety of policies respectively. To facilitate agents' collaboration with both humans and AI agents, we then explore the integration of these two directions under the umbrella of natural languages. All contributions are evaluated in simulated environments that model real-world applications following the widely adopted practices in the literature. We primarily consider simulated autonomous driving for cooperative and mixed-motive scenarios, and we also include a hardware security problem for adversarial scenarios.

1.1 Contributions

This dissertation makes the following contributions to the multi-agent learning literature:

1. This dissertation introduces and evaluates COOPERNAUT (Chapter 3), an end-to-end driving framework that generates efficient transmittable representations of the local point-cloud observation of autonomous agents through imitation learning of an expert driving policy with access to comprehensive environmental information. We show that with COOPERNAUT, autonomous agents can significantly reduce collisions without compromising traffic efficiency compared to disconnected vehicles in accident-prone scenarios. This contribution addresses *what* information to communicate **(Dimensions A)** through a learned representation space under the available bandwidth in autonomous driving.

2. This dissertation presents and evaluates LLM+DEBRIEF (Chapter 4), a self-play learning framework for embodied large language model (LLM) agents to communicate and collaborate via natural language in autonomous driving scenarios. The pipeline trains agents to articulate intentions, share critical observations, and negotiate driving plans with nearby vehicles. Experimental results demonstrate that agents can coordinate effectively using human-comprehensible natural language, a prerequisite for seamless human-AI collaboration. This contribution addresses **Dimensions A and C** by enabling natural language communication between agents for joint decision-making in dynamic environments.

3. This dissertation introduces and evaluates a reinforcement and game-theoretic training framework, MACTA (Chapter 5), which uses Proximal Policy Optimization (Schulman et al., 2017) as the best response oracle and fictitious play (Brown, 1951) as the empirical game-theoretic tool. We show that the resultant policy is able to generalize to unseen opponents and is robust against a dedicated adaptive opponent in a simulated cache timing attack scenario. This contribution partially addresses the generalization dimension **(Dimension B)**.

4. This dissertation addresses the generalization challenge in cooperative multi-agent settings **(Dimension B)** by proposing that an agent can emulate a coverage set of the teammate policy space through exposure to a diverse set of training partners (Chapter 6). To this end, we introduce L-BRDiv, a teammate generation method that approximates a diverse subset of policies requiringthat elicit distinct best responses. We show that L-BRDiv produces qualitatively diverse teammates and enables ad hoc agents to achieve state-of-the-art performance on standard ad hoc teamwork benchmarks at the time of publication.

5. This dissertation conducts an empirical study (Chapter 7) of applying decentralized multi-agent reinforcement learning to work with both humans and AI agents to improve traffic efficiency in autonomous driving. We delve into the decentralized training of Reinforcement Learning (RL) agents in a mixed environment where human and AI agents coexist. Experimental findings indicate that a small presence of RL autonomous vehicles can effectively collaborate to influence human drivers and amplify overall traffic efficiency within an open environment. This contribution explores the agents' intelligent interactions with both humans and other AI agents **(Dimension C)** without communication.

Collectively, these contributions address the central research question along the three proposed dimensions. Contributions 1 and 2 examine communication-supporting representations, encompassing both compact latent messages and natural language. Contributions 3 and 4 focus on generalization to diverse or adaptive partners through reinforcement learning and teammate policy diversification. Contributions 2 and 5 investigate collaboration with human-like or human agents, through explicit communication or decentralized coordination. Although each contribution primarily targets a specific dimension, together they offer complementary insights into designing multi-agent systems that can communicate, generalize, and collaborate effectively in open, realistic environments.

1.2 Reading Guide to the Dissertation

This dissertation is organized into five parts:

Part I: Background (Chapters 1–2) Introduces the motivation, challenges, and foundational concepts in multi-agent learning, reinforcement learning, and communication.

Part II: Learning to Communicate (Chapters 3–4) Explores how agents can generate latent and natural language messages to support cooperation. Chapter 3 introduces COOPERNAUT for latent communication; Chapter 4 presents LLM+DEBRIEF for natural language collaboration among embodied agents.

Part III: Learning to Generalize (Chapters 5–6) Investigates policy generalization in both adversarial and cooperative multi-agent settings. Chapter 5 focuses on robustness against adaptive attackers; Chapter 6 focuses on ad hoc teamwork with novel teammates.

Part IV: Learning with Human Proxies (Chapter 7) Studies autonomous agent-human collaboration using decentralized reinforcement learning and open traffic scenarios.

Part V: Related and Future Work and Conclusions (Chapters 8–10) Reviews related literature and discusses promising directions for extending the work, including embodied learning, open-ended teamwork, and real-world deployment, followed by concluding remarks.

Readers are encouraged to begin with **Part I**, which provides the motivation and necessary background for the dissertation. **Parts II, III,** and **IV** are organized thematically and are largely self-contained, each focusing on a distinct research dimension. For an overview of how this dissertation fits within the broader literature and how it may inspire future work, readers may refer to **Part V**.

Chapter 2

Background and Notation

This chapter formalizes the decision-making frameworks and learning objectives that guide the methods developed in this dissertation. Section 2.1 introduces the Markov Decision Process (MDP) formulation for single-agent reinforcement learning, including definitions of the Markov property, policies, value functions, and regret.

Section 2.2 extends this problem formulation to the multi-agent setting by presenting Partially Observable Stochastic Games (POSGs), which model strategic interactions among multiple agents under partial observability. It also introduces several relevant variants, including two-player zero-sum POSGs and communication-enabled policy classes that allow agents to exchange messages during decision-making.

Section 2.3 defines concepts in agent populations, distinguishing between the *focal population*, whose policies are optimized, and the *background population*, which models other agents in the environment. These distinctions support clear specification of learning objectives in heterogeneous multi-agent systems.

Finally, Section 2.4 introduces key learning objectives used throughout this dissertation, including Nash equilibrium, targeted optimality, generalizability to unseen partners, and exploitability by adaptive opponents. These criteria serve to evaluate the quality and robustness of the policies learned in different experimental settings.

2.1 Markov Decision Process

Reinforcement learning is learning what to do in different situations to maximize a reward signal through trial and error (Sutton and Barto, 2018). It uses the formal framework of a *Markov Decision Process* (MDP) to describe the interactions of a single agent with the environment. The key property of MDPs is the Markov property, which implies that an agent has all the information to make a decision based on the current state.

Definition 2.1.1 (**Markov Decision Processes**)**.** An MDP can be described using a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where:

- $\mathcal{S}$ is the state space of the environment,
- $\mathcal{A}$ is the action space of the agent,
- $\mathcal{P}$: $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the state transition function or the environment dynamics,
- $\mathcal{R}$: $\mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ the reward function, and
- $\gamma \in (0, 1]$ is the discount factor that defines how much immediate rewards are valued compared to future rewards.

Definition 2.1.2 (**Markov Property**)**.** The Markov property implies that the state transitions depend only on the current state and action:

$$P(s_{t+1} \mid s_t, a_t) = P(s_{t+1} \mid s_0, a_0, s_1, a_1, \ldots, s_t, a_t). \tag{2.1}$$

where t is the discrete time step in the interaction sequence between the agent and the environment

Definition 2.1.3 (**Policy**)**.** A policy π is a mapping from a state at time step t, $s_t \in \mathcal{S}$, to a probability distribution over actions a in the action space $\mathcal{A}$:

$$\pi(a|s_t) = P(a|s_t) \tag{2.2}$$

where P represent probability mass when $\mathcal{A}$ is discrete and is a probability density when $\mathcal{A}$ is continuous.

Definition 2.1.4 (**Optimal Policy**)**.** An optimal policy π^* maximizes the expected return of an episode:

$$\pi^* = \arg\max_{\pi} \ \mathbb{E}_{\pi}\big[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\big] \tag{2.3}$$

Definition 2.1.5 (**State Value Function (V)**)**.** The state value function under a policy π is defined as the expected cumulative future rewards starting from state s following the policy:

$$V^{\pi}(s) = \mathbb{E}_{\pi}\big[\sum_{t}^{\infty} \gamma^t R(s_t, a_t) \ \big| \ s_t = s\big]. \tag{2.4}$$

Definition 2.1.6 (**State-Action Value Function (Q)**)**.** The state-action value function under a policy π is defined as the expected cumulative future rewards obtained by taking action a in state s and then following the policy π:

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}\big[\sum_{t}^{\infty} \gamma^t R(s_t, a_t) \ \big| \ s_t = s, a_t = a\big]. \tag{2.5}$$

Definition 2.1.7 (**Regret**)**.** Regret measures the performance loss due to not following the optimal policy from the beginning. Given a time horizon T, the cumulative regret is defined as:

$$\text{Reg}(T) = \sum_{t=0}^{T} (V^*(s_t, a_t^*) - V(s_t, a_t)) \tag{2.6}$$

where $V^*(s)$ is the optimal state value function and R_t is the reward obtained at time t, $a_t^* = \arg\max_a Q^*(s_t, a)$ is the optimal action, and a_t is the action chosen by the agent at time t.

2.2 Partially Observable Stochastic Games

A Stochastic Game (**SG**) is a multi-agent extension of an MDP where multiple agents interact in a shared environment with individual rewards. Partially Observ-

able Stochastic Games (**POSG**s) extend SGs by introducing **partial observability**, meaning that agents do not have direct access to the full state but instead receive observations based on an observation function. If all the agent share the same rewards, then the POSG becomes Decentralized Partially Observable Markov Decision Process (**Dec-POMDP**).

Definition 2.2.1 (**Partially Observable Stochastic Games**)**.** A **POSG** can be described using a tuple $\langle \mathcal{I}, \mathcal{S}, \{\mathcal{A}_i\}_{i\in\mathcal{I}}, \mathcal{P}, \{\mathcal{R}_i\}_{i\in\mathcal{I}}, \{\mathcal{O}_i\}_{i\in\mathcal{I}}, \gamma \rangle$, where:

- $\mathcal{I}$ is the set of all N agents,
- $\mathcal{S}$ is the joint state space of the environment,
- $\mathcal{A}_i$ is the action space of agent i, and $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times ... \times \mathcal{A}_N$ is the joint action space of all agents,
- $\mathcal{P}$: $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the state transition functions or the environment dynamics,
- $\mathcal{R}_i$: $\mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function for agent i,
- $\mathcal{O}_i : \mathcal{S} \times \mathcal{A} \times \mathbb{O}_i \rightarrow [0, 1]$ is the observation function for agent i, where $\mathcal{O}_i(o_i|s, \mathbf{a})$ is the probability that agent i observes $o_i \in \mathbb{O}$ given state s and joint action $\mathbf{a}$.
- $\gamma \in (0, 1]$ is the discount factor that defines how much immediate rewards are valued compared to future rewards.

Note that when the observation functions reveal the full state, a POSG reduces to SG.

Definition 2.2.2 (**Two-Player Zero-Sum POSG**)**.** A **two-player zero-sum POSG** is a special case of a Partially Observable Stochastic Game where:

- $\mathcal{I} = \{1, 2\}$ denotes the set of two agents,

- the reward functions satisfy the zero-sum condition:

$$R_1(s, \mathbf{a}) = -R_2(s, \mathbf{a}), \quad \forall s \in \mathcal{S}, \mathbf{a} \in \mathcal{A}.$$

This reward condition implies that agent 1's gain is exactly agent 2's loss, and vice versa. The goal of each agent is to maximize its own expected return. Let $\pi = (\pi_1, \pi_2)$ denote the joint policy of both agents. The expected return for agent i is:

$$V^i(\pi_1, \pi_2) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_i(s_t, \mathbf{a}_t)\right],$$

and it holds that:

$$V^1(\pi_1, \pi_2) = -V^2(\pi_1, \pi_2).$$

Definition 2.2.3 (**Communication-Enabled Policy Class**)**.** A policy class Π_{comm} includes agents that, in addition to selecting actions based on their observations, can generate and interpret messages. Each agent's policy $\pi_i \in \Pi_{\text{comm}}$ consists of two components:

$$\pi_i(o_i) = (a_i, m_i) \tag{2.7}$$

where o_i is the agent's observation, a_i is the selected action, and m_i is a message communicated to other agents. These messages can be used by others to improve coordination and performance.

2.3 Agent Populations

In this dissertation, we divide agents into groups to build the evaluation metrics and learning objectives.

Definition 2.3.1 (**Background Population**)**.** The background population $\mathcal{B}$, excluded from the focal group, is a set of agents whose policies are not controlled by the learning algorithms and are treated as a part of environment dynamics.

Definition 2.3.2 (**Focal Population**)**.** The focal population $\mathcal{F}$ is a set of agents whose policies are controlled by the policies of interest to evaluate. In fully cooperative or mixed-motive games, a focal group should act like a team to maximize its **social welfare** — the cumulative return of the group given the background policies:

$$\max_{\{\pi_i\}_{i\in\mathcal{F}}} \mathbb{E}\Big[\sum_{i\in\mathcal{F}}\sum_{t=0}^{t=\infty} R_i(s_t, \mathbf{a}_t)\Big|\{\pi_j\}_{j\in\mathcal{B}}\Big] \tag{2.8}$$

where s_t is the state at time t, and $\mathbf{a_t} = (a_1^t, a_2^t, ..., a_N^t)$ is the joint action of all agents at time t.

2.4 Learning Objectives

In this dissertation, learning objectives guide both the evaluation and optimization of policies under different settings. We focus on objectives relevant to policy optimality, robustness, and generalization in multi-agent environments, particularly under conditions of partial observability and population diversity.

Definition 2.4.1 (**Nash Equilibrium**)**.** A Nash Equilibrium is a point in the space of joint policies (π_*^i, π_*^{-i}) where, for any player's policy π_*^i, we have

$$V^i(\pi_*^i, \pi_*^{-i}) \geq V^i(\pi^i, \pi_*^{-i}), \quad \forall i \in \mathcal{I}. \tag{2.9}$$

Namely, given all other agents' equilibrium policies π_*^{-i}, there is no motivation for agent i to unilaterally deviate from its current policy π_*^i to achieve higher returns.

Definition 2.4.2 (**Targeted Optimality**)**.** When interacting with a set of opponents or teammates, a policy or joint policy achieves *targeted optimality* if it yields the best possible reward (i.e., no other policy can do better) with high probability. Formally, a policy π^i achieves ε-approximate targeted optimality against a fixed set of background policies if:

$$V^i(\pi^i, \pi^{-i}) \geq V^i(\pi_*^i, \pi^{-i}) - \varepsilon, \quad \forall i \in \mathcal{I} \tag{2.10}$$

with high probability $1 - \delta$.

Definition 2.4.3 (**Generalizability**). A policy or policy group exhibits *generalizability* if it maintains strong performance when deployed with new or unseen background policies. Let μ be a distribution over possible background policies Π^{-i}. The expected performance of policy π^i is measured by the expected return when interacting with unseen policies:

$$\mathcal{G}(\pi^i) = V^i(\pi^i, \pi^{-i} \sim \mu(\Pi^{-i})) \tag{2.11}$$

Definition 2.4.4 (**Exploitability**). *Exploitability* quantifies the performance gap between a given policy and a best-response policy in a strategic environment. It measures how much an agent (or set of agents) can improve its return by deviating optimally while other agents' policies are fixed.

Formally, in an n-player Markov game with joint policy $\pi = (\pi_1, \ldots, \pi_n)$ and value function $V_i^\pi(s)$ for player i, the *one-sided exploitability* of player i from state s is:

$$\mathrm{Exp}_i(\pi \mid s) = \max_{\pi_i'} V_i^{(\pi_i', \pi_{-i})}(s) - V_i^\pi(s), \tag{2.12}$$

where π_{-i} denotes the policies of all players except i.

In the special case of a two-player zero-sum game where $V_1^\pi(s) = -V_2^\pi(s)$, the (symmetric) *exploitability* of the joint policy (π_1, π_2) is defined as the average of the two players' one-sided exploitabilities:

$$\mathrm{Exp}(\pi_1, \pi_2 \mid s) = \frac{1}{2}\left[\max_{\pi_1'} V_1^{(\pi_1', \pi_2)}(s) - V_1^{(\pi_1, \pi_2)}(s) + \max_{\pi_2'} V_2^{(\pi_1, \pi_2')}(s) - V_2^{(\pi_1, \pi_2)}(s)\right]. \tag{2.13}$$

This value is zero if and only if (π_1, π_2) is a Nash equilibrium.

Part II

Learning to Communicate

Chapter 3

Learning to Communicate in Latent Representations

Autonomous vehicles, as a class of embodied agents, typically rely on optical sensors to perceive their surroundings and make decisions based solely on their own observations. Despite recent advances in sensor technology and perception algorithms, these vehicles remain limited by their line of sight and often struggle to handle extreme or corner-case scenarios.

Recent advancements in telecommunication technologies have opened new opportunities for cooperative perception—a paradigm where vehicles share information through vehicle-to-vehicle (V2V) communication. This paradigm allows agents to move beyond single-agent perception and make more informed decisions by leveraging the observations of others (Figure 3.1). While the ideal scenario involves unrestricted information sharing among connected vehicles, real-world constraints—such as limited communication bandwidth—necessitate compact and efficient message representations.

This chapter introduces COOPERNAUT[1](Section 3.1), a learning-based, end-to-end framework that enables vehicles to generate and exchange ***latent represen-***

[1]COOPERNAUT is available at https://ut-austin-rpl.github.io/Coopernaut/.

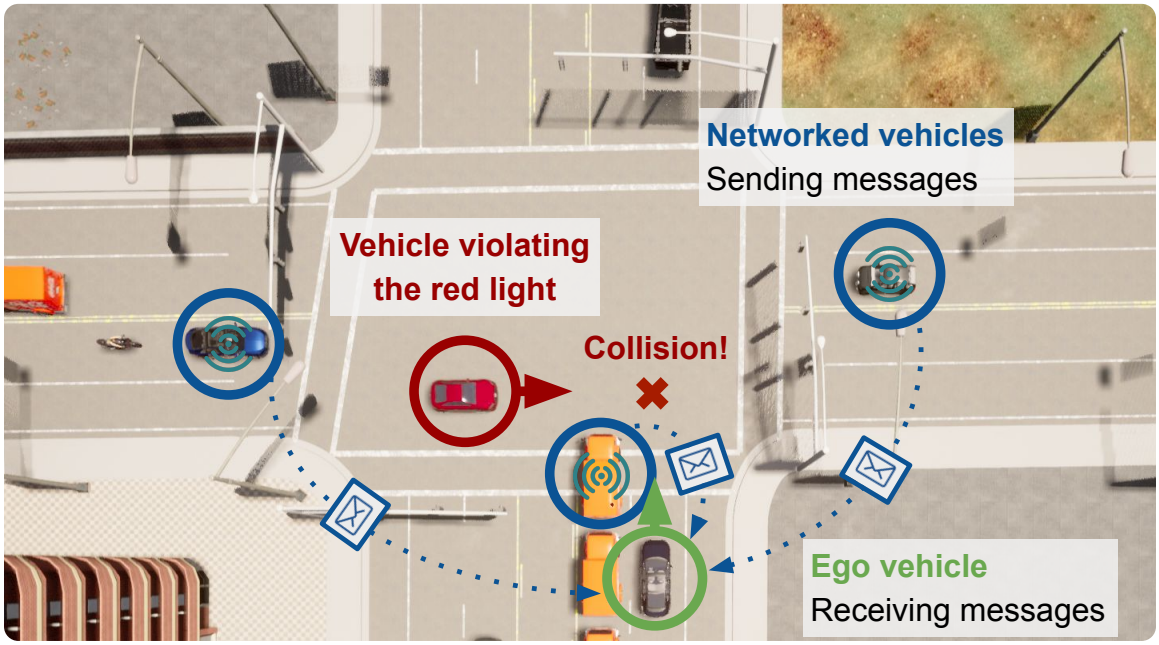


Figure 3.1: COOPERNAUT enables vehicles to communicate critical information beyond occlusion and sensing range for vision-based driving. The blue dashed arrows are information-sharing flows. Through cooperative perception, COOPERNAUT makes more informed driving decisions when line-of-sight sensing is limited.

tations of their local perceptions. Specifically, our model encodes LiDAR data into compact, point-based latent vectors that can be transmitted between vehicles via realistic wireless channels. The receiving agents aggregate these representations to make informed and coordinated driving decisions. Chapter 4 extends this framework to incorporate human-compatible communication by enabling the generation of natural language messages.

To evaluate our approach, we introduce AUTOCASTSIM (Section 3.2), a network-augmented driving simulation framework developed as part of the research reported in this chapter. It includes multiple accident-prone scenarios to test the benefits of cooperative perception. Chapter 4 will present an enhanced version of this environment, featuring negotiation-based scenarios and multi-agent learning capabilities.

Experimental results (Section 3.3) show that COOPERNAUT significantly reduces collision rates without compromising traffic efficiency, compared to agents with-

out communication. This contribution addresses **Dimension A** of the thesis in Chapter 1, focusing on *what* information should be communicated and *how* it can be efficiently represented under bandwidth constraints.

This work was published in the *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* 2022. The author developed the learning framework COOPERNAUT and conducted experiments; Hang Qiu contributed to the simulation environment AUTOCASTSIM; Dian Chen, Peter Stone, and Yuke Zhu provided valuable guidance and feedback throughout the project.

3.1 COOPERNAUT

Our goal is to learn a closed-loop policy that controls an autonomous ego vehicle, using LiDAR observations $O_t^{(\text{ego})}$ at time t. Assume that there exists a *variable* number of N_t neighboring vehicles in the range of V2V communications at time t, where $O_t^{(i)}$ is the raw 3D point cloud from the onboard LiDAR of the i-th vehicle. The ego vehicle's driving policy, based on cooperative perception, is defined as $\pi(a_t|O_t^{(\text{ego})}, O_t^{(1)}, \ldots, O_t^{(N_t)})$, where the ego vehicle makes control decisions a_t using its own observation $O_t^{(\text{ego})}$ and the encoded information received from N_t neighboring vehicles. Here π is parameterized by a deep neural network and trained end-to-end. In principle, we can transmit all cross-vehicle observations to the ego vehicle and process them as a whole. In practice, we have to take into account the networking bandwidth limit, which only allows for message sizes that are orders of magnitude smaller. We thus first process the raw point clouds into compact representations, which can be transmitted through the V2V channels in real-time.

3.1.1 Background: Point Transformers

COOPERNAUT's model backbone is the Point Transformer (Zhao et al., 2021), a neural network structure that learns compact point-based representations from 3D point clouds. It reasons about non-local interactions among points and produces

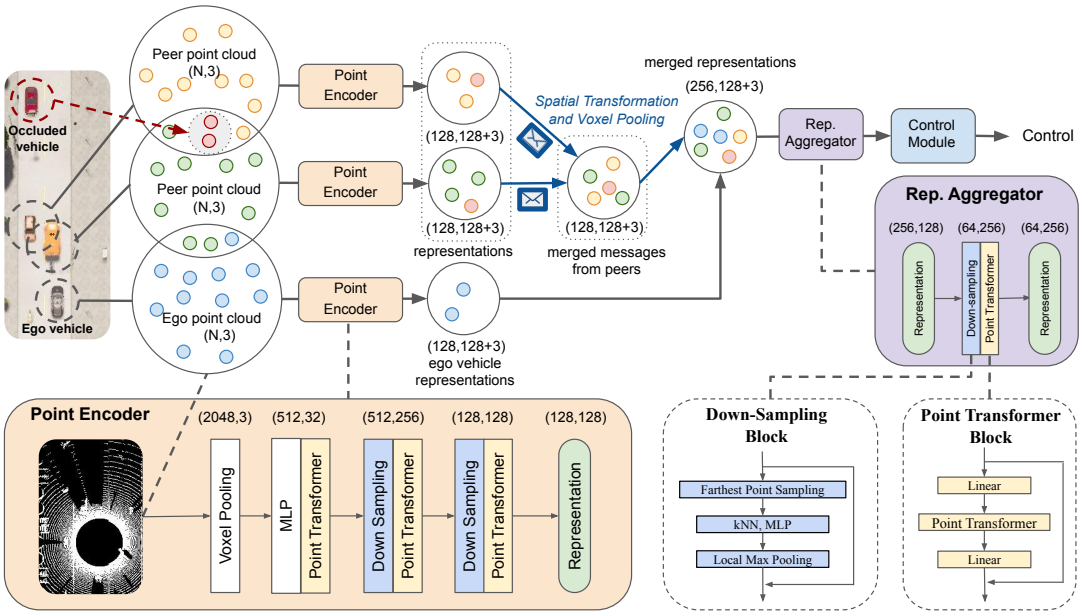


Figure 3.2: **COOPERNAUT** is an end-to-end vision-based driving model for networked vehicles. It contains a *Point Encoder* to extract critical information locally for sharing, a *Representation Aggregator* for merging multi-vehicle messages, and a *Control Module* to reason about the joint messages. Each message produced by the encoder has 128 keypoint coordinates and their associated features. The message is then spatially transformed into the ego frame. The ego vehicle merges incoming messages and computes aggregated representations through voxel max-pooling. Finally, the aggregator synthesizes joint representations from the ego vehicle and all its neighbors before passing them to the Control Module to produce control decisions. The numbers in parentheses denote data dimensions.

permutation-invariant representations, making itself effective in aggregating multi-vehicle point clouds. Here we provide a brief review of Point Transformers.

We adopt the same design as Zhao et al. (2021), which uses *vector self-attention* to construct the Point Transformer Layer. We also apply subtraction between features and append a position encoding function δ to both the attention vector γ and the transformed features α:

$$y_i = \sum_{x_j \in \mathcal{X}(i)} \rho(\gamma(\phi(x_i) - \psi(x_j) + \delta)) \odot (\alpha(x_j) + \delta) \tag{3.1}$$

Here x_i and x_j are input features of the point i and j respectively, y_i is the

output attention feature for point i, and $\mathcal{X}(i)$ represents the set of points in the neighborhood of x_i; ϕ, ψ and α are point-wise feature transformations implemented as multilayer perceptrons (MLPs). γ is also an MLP with two layers and a ReLU activation; δ is a position encoding function, and ρ is a normalization function *softmax*. Given the 3D coordinates $p_i, p_j \in \mathbb{R}^3$ for points i and j, the position-encoding function is formulated as follows:

$$\delta = \theta(p_i - p_j) \tag{3.2}$$

where θ is an MLP with two linear layers and one ReLU.

A *Point Transformer block* is shown in Figure 3.2, which integrates the self-attention layer, linear projections, and a residual connection. The input is a set of 3D points p with a feature x of each point. This block enables local information exchange among points, and produces new feature vectors for each point. The *down-sampling block* in Figure 3.2 is to reduce the cardinality of the point sets. We perform farthest point sampling (Eldar et al., 1997) to the input set to obtain a well-spread subset, and then use kNN graph and (local) max pooling in the neighborhood to further condense the information to smaller sets of points. The output is a subset of the original input points with new features.

3.1.2 COOPERNAUT

We use cross-vehicle perception to augment the sensing capabilities of the ego vehicle, enabling it to make more informed decisions under challenging situations than it could with onboard perception alone. The key challenges are to transmit sensory information efficiently through realistic V2V channels, to understand the traffic situation from the aggregated information, and to determine the reactive driving action in real-time. Our COOPERNAUT model, illustrated in Figure 3.2, is composed of a Point Encoder for each neighboring V2V vehicle to encode its sensory data into compact messages, a Representation Aggregator to integrate the messages from neighboring cars with the ego perception, and a Control Module which translates the integrated

representations to driving commands.

Point Encoder. To reduce communication burdens, every V2V vehicle processes its own LiDAR data locally and encodes the raw 3D point clouds into keypoints, each associated with a compact representation learned by the Point Transformer blocks. We construct the encoder with three Point Transformer blocks accompanied by two down-sampling blocks, both with a downsampling rate of $(1, 4, 4)$. The final cardinality of intermediate representations is $P/16$, where P is the number of points in the raw point cloud. In our experiments, we preprocess 65,536 raw LiDAR points to 2,048 points via voxel pooling, *i.e.,* representing the points in a voxel grid using their voxel centroid.

The message M_j produced by the j-th vehicle comprises a set of position-based representations M_j and is mathematically described as $M_j = \{(p_{jk}, R_{p_{jk}})\}^K$, where $p_{jk} \in \mathbb{R}^3$ for $k = 1, \ldots, K$ is the position of a keypoint in 3D space and $R_{p_{jk}}$ is its corresponding feature vector produced by the Point Encoder. We limit the size of M_j to be at most K tuples. These keypoints carrying features are in each vehicle's local frame. They preserve the spatial information as their coordinates are sampled from raw point clouds.

Representation Aggregator. Messages transmitted from other vehicles need to be fused and interpreted by the ego vehicle. The Representation Aggregator (RA) for cooperative perception is implemented as a voxel max-pooling operation and a point transformer block. RA first spatially transforms the keypoints in other vehicles' coordinates to the ego vehicle's frame using their relative poses. This operation assumes accurate vehicle localization (*e.g.,* using HD maps). It then aggregates the incoming messages that are spatially close via max-pooling all the points located inside the same voxel grid cell. Finally, it fuses the multi-view perception information with another Point Transformer block. The two operations above preserve the permutation invariance with respect to the ordering of other vehicles and can handle a variable number of sharing vehicles. For bandwidth control, COOPERNAUT receives messages

from three randomly chosen V2V vehicles in the vicinity.

Control Module. The control module is a fully-connected neural network designed to make control decisions based on the received messages. These control decisions include the throttle, brake, and steering, denoted as scalar T, B, S respectively. These values output from the model are first clipped to their valid ranges (e.g., [0,1] for throttle). To enforce compliance with speed limit regulations, we apply a PID-based speed controller that post-processes the model's outputs to prevent violations due to excessive acceleration.

3.1.3 Policy Learning: Imitation Learning

We train our model to imitate the expert policy with privileged information using DAgger (Ross et al., 2011b). To warm-start policy learning, we first train the model using behavior cloning.

Behavior Cloning. Behavior Cloning is designed to minimize the distribution gap between the training policy and the expert policy. The goal is to find an optimal policy $\hat{\pi}$ such that the loss w.r.t. the expert's policy π_{expert}, under its induced distribution of states S is minimized, *i.e.,*

$$\hat{\pi} = \arg\min_{\pi \in \Pi} \mathbb{E}_{s \sim S}[\ell_{\text{control}}(\pi(s), \pi_{\text{expert}}(s))]. \tag{3.3}$$

The objective function ℓ_{control} is a linear combination of ℓ_1-loss of throttle, brake, and steering between the policy's actions and the expert's actions:

$$\ell_{\text{control}} = \eta_1 \ell_{\text{throttle}} + \eta_2 \ell_{\text{brake}} + \eta_3 \ell_{\text{steer}} \tag{3.4}$$

where η_1, η_2, η_3 are the coefficients of the loss for each action. All three coefficients are set to 1 in our experiments.

DAgger. Limitations of behavior cloning for autonomous driving have been discussed in Codevilla et al. (2019). DAgger (Ross et al., 2011b) address the covariance shift issues via online training. The core idea is to let the student policy interact with the

environment under the supervision of the expert and record the expert's actions on the same states visited by the student. The training dataset is iteratively aggregated, using a mixture of the student's and expert's actions. The sampling policy π_i for the i-th iteration follows

$$\pi_i = \begin{cases} \pi_{\text{expert}}, & \text{w.p. } \beta_i \\ \pi_{\text{student},i}, & \text{w.p. } 1-\beta_i \end{cases} \tag{3.5}$$

where $\beta_i = \beta_0 \times \beta_{i-1}$ are exponentially decreasing from the initial β_0, representing the probability that the expert's action is executed at the i-th iteration.

3.1.4 Implementation Details

When more than three neighboring vehicles send messages, we randomly select messages from three of the vehicles. All the neighbors encode their processed point clouds locally by the 3-block Point Encoder and send the messages of size $128 \times (128, 3)$ and warp the coordinates to the ego frame. We aggregate the merged representations by another block of Point Transformer. After global max pooling, the features are concatenated with the ego speed feature before passing to the fully connected layer.

Our model has a 90ms latency on an NVIDIA GTX3090 GPU, where the point encoder takes 80ms. Our model training consists of two stages: behavior cloning and DAgger. We first train every scenario-specific model by behavior cloning, then the final policy of behavior cloning serves as an initial student policy for DAgger. We collect 4 new trajectories and append them to the Dagger dataset every 5 epochs using a sampling policy (see Section 3.1.3) with $\beta_0 = 0.8$ during the DAgger stage. For all data used for training, 25% of them are collected under accident-prone scenarios (with an occluded collider vehicle inserted) and 75% of them are normal driving trajectories.

3.2 Environment: AutoCastSim

We present AutoCastSim, a simulation framework which offers network-augmented autonomous driving simulation on top of CARLA (Dosovitskiy et al., 2017). This simulation framework allows custom designs of various traffic scenarios for training and evaluating cooperative driving models. The simulated vehicles can be configured with realistic wireless communications. It also provides a path-planning-based oracle expert with access to privileged environment information.

3.2.1 Scenarios

We designed three challenging traffic scenarios, shown in Figure 3.3, in AutoCastSim as our evaluation benchmark. These scenarios are selected from the pre-crash typology of the US National Highway Traffic Safety Administration (NHTSA) (Najm et al., 2013), where limited line-of-sight sensing affects driving decisions:

*** Overtaking.** A truck blocks the way of a sedan in a two-way single lane road with a dashed yellow lane divider. The truck also impedes the sedan's view of the opposite lane. The ego car has to overtake with a lane change maneuver.

*** Left Turn.** The ego car tries to turn left on a left-turn yield light but encounters another truck in the opposite left-turn lane, blocking its view of the opposite lanes and potential straight-driving vehicles.

*** Red Light Violation.** The ego car is crossing the intersection when another vehicle is rushing the red light. LiDAR fails to sense the other vehicle because of the lined-up vehicles waiting for the left turn.

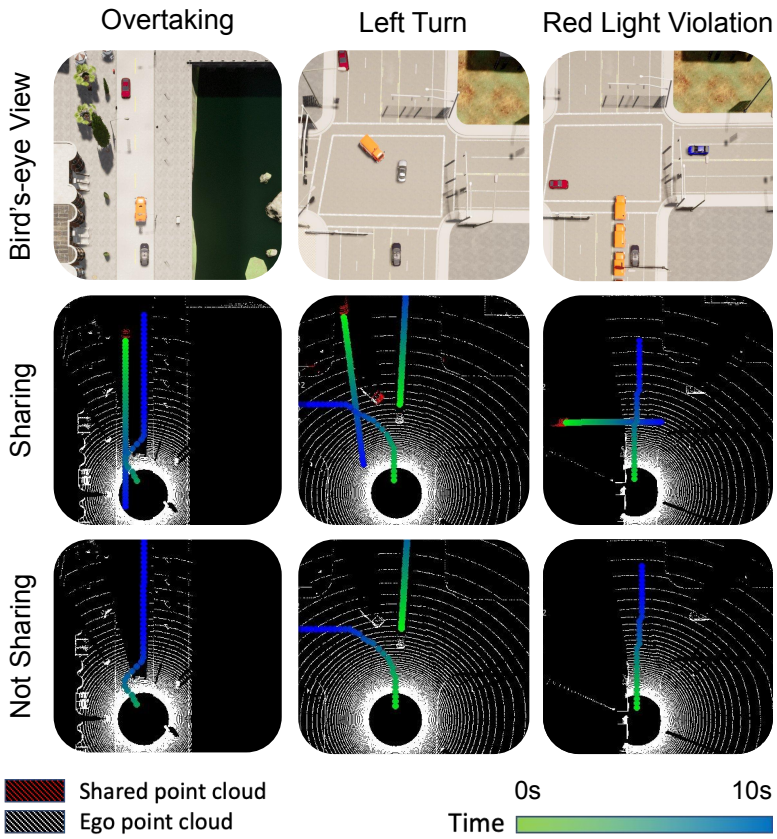


Figure 3.3: Benchmarking scenarios in AutoCastSim. The gray car is the ego vehicle controlled by our model. The orange trucks are large vehicles that partially block views of the environment. The red car is not networked and is likely to collide with the ego vehicle. All other vehicles are background traffic, either with or without sharing capability. The green-blue dots mark the planned temporal trajectories for any moving vehicle, with green dots being waypoints closer in the future than the blue dots. If two planned trajectories intersect at a similar color (time), it indicates that a collision may happen. For every scenario, an RGB bird's-eye view (BEV), an ego-centric LiDAR BEV image, and a multi-vehicle fused LiDAR BEV image are presented (left to right). We use relatively little background traffic here for illustration, and will study the effect of traffic density in Section 3.3.3.

3.2.2 V2V Communication

To simulate realistic wireless communication, we use real V2V wireless radios to profile wireless bandwidth capacity and packet loss rate due to channel diversity between mobile agents. Specifically, we use three iSmartways DSRC radios and

three C-V2X radios (iSmartWays Technology Inc., 2018), mounted on top of moving vehicles, to measure the maximum capacity of continuous wireless transmission in practice. Table 3.1 shows the tested throughput and packet loss. It also shows the throughput of WiFi (802.11n, ac) for context. Note that the 802.11 series is not designed for mobile scenarios. Table 3.1 shows that V2V bandwidth is two orders of magnitude smaller than the indoor wireless capacity. The extremely limited bandwidth, in practice, poses significant challenges for designing the representations for V2V communication. We use the Winner II wireless channel model (Meinilä et al., 2009) in our simulator and use the measured C-V2X radio capacity and packet loss rate in the channel model. We refer to prior work (Qiu et al., 2021) for the design and implementations of the coordination, scheduling, and the network transport layer.

Table 3.1: Measured wireless **throughput** and **packet loss rate** using off-the-shelf wireless radios.

	DSRC	C-V2X	802.11n	802.11ac
Throughput (Mbps)	2.0	7.2	~ 200	~ 900
Packet Loss (%)	< 5	< 5	> 90	> 90
Mobility support	Yes	Yes	No	No

3.2.3 Oracle Expert

The expert has access to the privileged information of the traffic scenarios. The information includes the point cloud from the LiDARs of all neighboring vehicles and the positions and speeds of these neighboring vehicles and other traffic participants. The expert transforms all of the point clouds from neighboring cars to its ego perspective (which is impractical due to the wireless bandwidth limit mentioned above). The transformed point cloud is fused for downstream obstacle detection and planning. The expert policy leverages all information above to analyze and avoid possible collisions. The path planning algorithm uses an A* trajectory planner (Hart et al., 1968) with pose and distance heuristics. The expert moves at a target speed of 20km/h.

3.3 Experiments

We first discuss the evaluation method and the experiment setup, and then give a brief overview of our baselines. Next we present the main quantitative evaluation results of our methods against baselines. Finally, we provide further analysis and visualization to understand the quality of our learned model. In this section, we answer the following research questions:

Q1 Can COOPERNAUT generate effective latent representations to facilitate safer and more efficient driving compared to other baselines? (Yes, it outperforms other methods in both safety and efficiency metrics.)

Q2 Does COOPERNAUT generalize across different traffic densities? (COOPERNAUT achieves the best generalization performance among all the compared methods, with degradation performance as traffic gets denser.)

3.3.1 Experimental Settings

Scenario Configuration. We generate traces from the three scenarios we implemented in AUTOCASTSIM (Section 3.2.1) for training and evaluation. These scenarios can be programmatically re-configured with key parameters, notably the number of vehicles, vehicle spawning locations, and vehicle cruising speeds. Random combinations of these parameters are sampled to procedurally generate traces with traffic situations of varying complexity — in some cases the ego vehicle has to take emergency actions to avoid potential collisions, while in other cases, cruising along the default route can reach the destination.

Dataset. Specifically, for each scenario, we use the expert agent (Section 3.2.3) to generate an initial training set of 12 traces with randomized scenario configurations, followed by another randomly configured 84 traces for DAgger. In the evaluation, we systematically test each model on a spectrum of 27 randomly selected accident-prone environment configurations over three repeated runs, each using different random

seeds for background traffic. For fair comparison, we use a fixed set of 27 test configurations to evaluate all models.

Metrics. We report three metrics, *Success Rate*, *Collision Rate*, and *Success weighted by Completion Time*:

Success Rate (SR). A *successful completion* of the scenario is defined as the ego agent reaching a designated target location in a permissible time without collision or prolonged stagnation. The success rate is defined as the percentage of *successful completions* among all evaluated traces.

Collision Rate (CR). Collision is the most common failure mode. *Collision rate* is defined as the percentage of evaluation traces where the ego vehicle collides with any entity, such as vehicles, buildings, etc.

Success weighted by Completion Time (SCT). SR reflects overall task success or failure. It does not differentiate the amount of time a driving agent needs to complete the traces. We introduce a third metric to weigh the success rate by the completion time ratio between the expert and the agent:

$$SCT = \mathbb{I}\{\text{agent success}\}\frac{T_{\text{expert}}}{T_{\text{agent}}} \tag{3.6}$$

where $\mathbb{I}$ is an indicator function, and T_{expert} and T_{agent} are the expert's and the agent's completion time, respectively. As the expert agent should require no longer completion time than the agent, the ratio resides in the range of $[0, 1]$.

3.3.2 Baselines

We compare COOPERNAUT with non-V2V and V2V driving baselines. For a fair comparison, we adopt the same neighbor selection process (Section 3.1.4) in all V2V approaches.

*** No V2V Sharing.** The non-sharing baseline makes decisions solely based on the onboard LiDAR data and ego speed. The model shares the same data processing scheme for an individual vehicle and point encoder architecture as our final model.

*** Early Fusion.** The Early Fusion model assumes an unrealistic communication bandwidth, with which it can transmit and fuse the entire raw point cloud data from all neighboring vehicles. While this method is intractable in practice, it serves as a baseline to examine our point-based architecture's effectiveness in representation learning. To fit this model in GPU memory, we limit the size of the fused input points to 4,096. Like the previous baseline, Early Fusion also uses a 3-block Point Transformer encoder.

*** Voxel GNN.** We adapt V2VNet (Wang et al., 2020), which is designed for 3D detection and motion forecasting, to learn end-to-end driving. Every vehicle processes its local point cloud onboard and shares a voxel representation with the ego vehicle for control. It uses a graph neural network (GNN) in the ego frame as the aggregator. The control actions are predicted from the GNN-fused representations.

For a fair comparison, all baselines and proposed approaches are independently trained over three repeated runs with the same training parameters (Section 3.1.4). We report the average performance over the three runs on the same scenario configurations (Section 3.3.1).

3.3.3 Quantitative Results

This section presents the empirical evaluations of all the models in the three benchmarking scenarios.

Scenario Completion. Table 3.2 shows the performance comparisons in each of the three traffic scenarios. In all three scenarios, the No V2V Sharing model has performed poorly, with less than 50% success rate for each scenario and high collision rates. All three cooperative driving models, including Early Fusion, Voxel GNN, and COOPERNAUT, have achieved substantially higher SR and SCT scores and lower collision rates than the No V2V Sharing baseline. It indicates that the V2V communication provides critical information about the traffic situation over the ego vehicle's line-of-sight sensing to make informed driving decisions. The Early Fusion method

Table 3.2: Quantitative results of different models over three repeated runs. SR: Success Rate, in percentage; SCT: Success weighted by Completion Time, in percentage; CR: Collision Rate, in percentage; In the Bandwidth column, we report the communication throughput required without data compression. The bandwidth is calculated by assuming 10 Hz LiDAR scanning frequency.

Metrics \ Methods	No V2V Sharing	Early Fusion	Voxel GNN	COOPERNAUT (Ours)
Bandwidth (Mbps)	–	60.0	5.60	**5.10**
Overtaking				
SR↑	45.3±0.6	81.9±7.2	70.0±4.8	**90.5±1.2**
SCT↑	43.6±0.7	81.2±5.2	67.8±4.2	**88.4±1.1**
CR↓	35.8±3.6	11.9±5.1	16.1±3.6	**4.5±3.1**
Left Turn				
SR↑	40.3±5.9	72.8±8.6	53.5±6.9	**80.7±5.2**
SCT↑	37.8±4.6	68.8±8.9	51.0±6.9	**76.2±3.9**
CR↓	55.6±9.6	26.3±8.1	33.3±7.3	**18.1±6.2**
Red Light Violation				
SR↑	47.3±18.7	78.6±11.8	64.2±25.3	**80.7±7.6**
SCT↑	46.1±18.4	75.8±9.1	62.0±24.8	**77.8±7.0**
CR↓	51.4±17.4	**17.7±15.2**	35.0±25.9	**17.7±7.8**

improves over the non-V2V baseline by over 30% in average success rate. However, the Early Fusion baseline requires transmitting raw point clouds across vehicles, leading to an unrealistic bandwidth requirement of 60Mbps (before data compression).

In contrast, pre-processing raw sensory data into representations has dramatically reduced the bandwidth requirements while improving driving performances. Both Voxel GNN and COOPERNAUT perform sensory fusion on the representation level. In comparison to the other cooperative driving models, COOPERNAUT outperforms both Early Fusion and Voxel GNN baselines for all three scenarios. We hypothesize that the point-based representation learning of COOPERNAUT makes it robust to localization errors compared with fusing raw points in Early Fusion. Furthermore, the explicit representation of point 3D locations and the point sampling module of COOPERNAUT retain a high spatial resolution of its intermediate representations in contrast to the voxel-based feature maps used by Voxel GNN.

Bandwidth Requirement. As shown in Table 3.2, sharing raw point cloud at the LiDAR scanning rate of 10fps would require a wireless bandwidth of 60Mbps, far beyond the achievable bandwidths in the current (DSRC) and future (C-V2X or LTE-direct) V2V communication technology (expected less than 10Mbps, see Table 3.1). V2VNet (Wang et al., 2020) claims a bandwidth requirement of 25 Mbps with point cloud compression, which is also beyond what current V2V radios can support. In our design, both Voxel GNN and COOPERNAUT require less than 6Mbps bandwidth, a 4× reduction of the communication data sizes of V2VNet without compression.

When developing the V2V models, we carefully explored the design space of the sharable representation size and its bandwidth requirement for both Voxel GNN and COOPERNAUT. For example, if COOPERNAUT were to share a 32×32 representation, it only needs 0.9 Mbps. However, the coarse information is insufficient for the model to achieve good performance. We find that a 128×128 point representation meets the bandwidth requirements (Table 3.1) without substantial performance degradation.

Sensitivity to Traffic Density. We further test COOPERNAUT under varied traffic densities in the most challenging Left Turn scenario. Figure 3.4 shows that our method generalizes to variable traffic densities, consistently outperforming the No V2V Sharing baseline. Qualitatively, we observe that No V2V Sharing drives slower in denser traffic, reacting better to emergency situations. In contrast, V2V methods do not improve much in denser traffic, as they tend to be impacted by the increased stochasticity of incoming messages from changing neighbors. Nonetheless, COOPERNAUT outperforms the baselines in all traffic densities with over 30% higher success rates than No V2V Sharing.

Qualitative Visualizations. Figure 3.5 shows an example evaluation trajectory from Left Turn. The left-turning ego vehicle (grey) can proactively avoid collision by yielding to the opposite-going cars with COOPERNAUT. A common failure pattern of the non-sharing model is that it drives ahead to its target location regardless of any traffic violators or potential colliders due to the limited line-of-sight of its ego

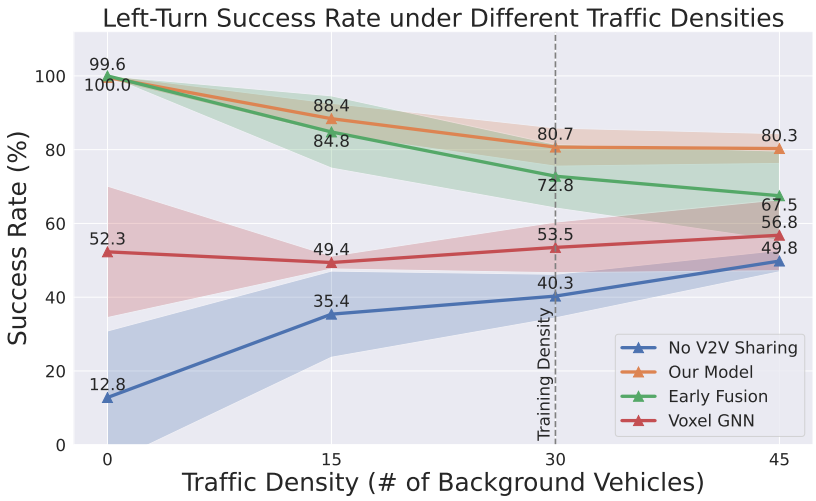


Figure 3.4: Sensitivity analysis on the varying levels of traffic densities in the Left Turn scenario.

perception. The transmitted messages through V2V channels help our model resolve the ambiguity with cross-vehicle perception, leading to safer driving decisions in this accident-prone situation.

3.4 Related Work

CoopernAut lies at the intersection of learning-based driving, 3D perception, and networked multi-agent systems. We summarize related literature across three key threads: learning driving policies from data, processing 3D perception for informed decisions, and leveraging inter-vehicle communication for cooperative behavior.

Deep Learning for Driving Policy. Learning a driving controller involves training closed-loop policies using deep networks, usually via imitation learning and/or reinforcement learning. Imitation learning for autonomous driving was pioneered by Pomerleau (1988), and has since then been extended to urban and more complex scenarios (Codevilla et al., 2018; Bansal et al., 2018; Sauer et al., 2018; Codevilla et al., 2019; Chen et al., 2020; Prakash et al., 2021). Very recently, reinforcement learning

Figure 3.5: Comparison of trajectories in the Left Turn scenario. The grey car in the pictures is the controllable ego vehicle. The red car is going straight in the opposite direction, occluded behind the orange truck. Our model avoids the collision as it is able to see the red light violating vehicle from cooperative perception (highlighted in the yellow box).

has also made progress in autonomous driving (Toromanoff et al., 2020; Chen et al., 2021), showing potential to train better policies in complex situations (Toromanoff et al., 2020; Chen et al., 2021). However, reinforcement learning is known to be more data-hungry and requires engineering a high-quality reward function. We follow the imitation learning paradigm but use an expert oracle with complete global information (Chen et al., 2020) for training efficiency.

3D Perception for Autonomous Driving. 3D perception has become more popular in autonomous driving due to the decreasing cost of commoditized LiDAR sensors. Zhou and Tuzel (2018) pioneered using 3D object detection in autonomous driving, and since then, it has been further developed as better models, and more advanced techniques have been discovered. Very recently, Prakash et al. (2021) also explored end-to-end driving using point cloud data. Two families of 3D perception backbones have been widely adopted: *voxel*-based methods discretize points to voxels (Zhou and Tuzel, 2018; Lang et al., 2019; Shi et al., 2020); and *point*-based methods directly op-

erate on coordinates (Qi et al., 2017a;b; Zhao et al., 2021). COOPERNAUT uses a transformer-based architecture (Vaswani et al., 2017) with point-based representations (Zhao et al., 2021; Qi et al., 2017a;b), which preserves high spatial resolutions with discretization and requires lower bandwidths to transmit without compression needed by prior work (Wang et al., 2020).

Networked Vehicles and Cooperative Perception. Network connectivity offers a great potential for improving the safety and reliability of self-driving cars. Vehicles can now share surrounding information via Vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2X) channels using wireless technologies, such as Dedicated Short Range Communication (DSRC) (Kenney, 2011) and cellular-assisted V2X (C-V2X) (Gallo and Harri, 2013; Qualcomm, 2019). These V2V/V2X communication devices are increasingly deployed in current and upcoming vehicle models (Thompson, 2016; Plungis, 2018)). The academic community has built city-scale wireless research platforms (COSMOS (Yu et al., 2019)) and large connected vehicle testbeds (*e.g.,* MCity (Bezzina et al., 2023), DRIVE C2X (Stahlmann et al., 2011)), to explore the feasibility of cooperative vehicles and applications. Prior work (Qiu et al., 2018; Chen et al., 2019) proposed cooperative perception systems that broaden the vehicle's visual horizon by sharing raw visual information with other nearby vehicles. Such systems can be scaled up to dense traffic scenarios leveraging edge servers (Zhang et al., 2021) or in an ad-hoc fashion (Qiu et al., 2021). Recent work (Wang et al., 2020; Li et al., 2021b; Xu et al., 2022) proposed multi-agent perception models to process sensor information and share compact representations within a local traffic network. In contrast, we focus on cooperative driving of networked vehicles with onboard visual data and realistic networking conditions, advancing towards real-world V2V settings.

3.5 Summary, Limitations, and Future Work

In summary, this chapter investigates vision-based driving using cooperative perception for networked vehicles in a newly designed simulation benchmark AutoCastSim. We introduce Coopernaut, an end-to-end driving policy that encodes, aggregates, and analyzes 3D LiDAR data from networked vehicles. The point encoder and representation aggregator of Coopernaut retain detailed spatial information and are robust to a varying numbers of communicating vehicles. Our empirical results show that our method improves the robustness of autonomous driving policies in risk-sensitive traffic scenarios. This chapter contributes to **Dimension A: Communication-Supporting Representation** of the core research problem.

This work has limitations and ample room for future extension.

Dependence on Oracle-Guided Learning. First, our method relies on a hand-engineered oracle for imitation learning. It leaves open questions to investigate adaptive strategies of *when* to communicate, *what* to encode in messages, and *how* to drive cooperatively, ideally without the need of an algorithmic oracle.

Idealized Communication Assumptions. While our cooperative perception model conforms to realistic wireless bandwidth, we do not take into account practical networking issues, including transmission latency, networking protocols, and repetitive or lost packets. Nonetheless, Coopernaut is robust to packet loss to a certain extent (5% as configured in AutoCastSim). Random neighbor selection also adds another layer to mitigate packet loss from individual transmitters.

Idealized Localization Assumptions. Furthermore, highly accurate vehicle localization is assumed, which is used by Coopernaut to transform the point-based representations from neighboring vehicles to the ego vehicle's reference frame, even though AutoCastSim simulates slight errors in the pose and height estimation of

a vehicle. In reality, without a high definition map (HDMap), localization error can yield up to meter-level displacement. Using HDMap can significantly improve location and pose estimation, which is commonly adopted in both industry and academia (Yang et al., 2018; Li et al., 2021a).

Limitations in Perceiving Small Objects. For fair comparison, we use the same down-sampling scheme for all point-based baselines and our approach, which proves to be effective in our scenarios with moving vehicles and large obstacles. For smaller objects like pedestrians, adaptive sampling schemes based on semantic information are a promising direction for future work. One could also extend the model architecture of COOPERNAUT to better incorporate temporal information for improving driving performance.

The next chapter will introduce a method that studies how to enable agents to speak human language to facilitate multi-agent cooperation.

Chapter 4

Learning to Communicate in Natural Language

Chapter 3 introduced a framework for encoding perceived point cloud data into latent representations for inter-agent communication. While effective for coordination among co-trained autonomous agents, it limits broader participation from those with different representations or language and leaves human drivers reliant solely on their local perceptions without being privy to the collaboration efforts. To enable collaboration that includes human participants, this chapter explores the use of ***natural language*** communication in multi-agent systems.

Facilitating natural language interaction among embodied agents presents several challenges. Agents must be capable of both generating human-interpretable language and understanding messages in a grounded, actionable manner to produce appropriate embodied behaviors. Traditionally, each of these capabilities would require extensive supervised training data. However, datasets featuring natural language-based inter-agent communication for collaboration are scarce.

Recent advances in large language models (LLMs) offer a promising alternative. By leveraging the agentic capabilities of LLMs, we enable self-play and reflective learning without relying on imitation from human-annotated communication data. This chapter explores the potential of LLM-based agents to autonomously

develop effective communication and cooperation strategies in multi-agent driving scenarios—using only self-generated interactions.

This chapter introduces LLM+DEBRIEF[1] (Section 4.2), a multi-agent learning method that enables LLM agents to engage in **centralized** debriefing post-interaction to reflect on and refine their **decentralized** communication and collaboration strategies. These improved strategies are then distilled into decentralized execution policies (Section 4.4.3). To support this work, we extend AUTOCASTSIM (Chapter 3) and develop TALKINGVEHICLESGYM (Section 4.3), a realistic simulation framework that models vehicle-to-vehicle communication in a suite of accident-prone driving scenarios that could be resolved by cooperative perception and negotiation. Our experiments (Section 4.4) show that even when LLM agents initially fail to coordinate effectively, LLM+DEBRIEF allows them to iteratively learn what messages to send and how to respond to improve task success.

This chapter focuses on the content and structure of natural language communication for embodied collaboration, with the goal of ultimately enabling human-agent interaction. It directly addresses **Dimension A** (Communication-Supporting Representations) and **Dimension C** (Collaborate with Human-Like Agents) as introduced in Chapter 1.

This work is currently under review. The author developed the learning framework LLM+DEBRIEF, the multi-agent simulator TALKINGVEHICLESGYM, and conducted all experiments. Chen Tang, Jarrett Holtz, Janice Nguyen, Alessandro G. Allievi, Hang Qiu, Peter Stone provided valuable feedback and guidance throughout the project.

[1] LLM+DEBRIEF is available at https://talking-vehicles.github.io/.

4.1 Problem Definition

In this chapter, we focus on the subset of agents that are actively participating in the cooperation. We assume that these cooperative vehicles implicitly aim to help each other, treating all other (referred to as "background") vehicles as uncontrollable elements of the environment. Therefore, we frame the problem of ***Talking Vehicles*** as a partially observable stochastic game (POSG, as defined in Chapter 2), focusing on optimizing the social welfare of a *focal population* ($\mathcal{F}$) (Agapiou et al., 2022) — defined as the joint reward of all participating agents — as the primary objective. The reward functions associated with each agent's individual tasks may or may not fully align, necessitating coordination among agents to achieve high joint rewards. Each agent's observation space is limited to a partial view of the full state, and agents make decisions in a decentralized manner based on their own partial observations and messages received from other agents. In this problem, each agent's action space comprises two main components: **(1) generating messages** and **(2) controlling the vehicle**. In this work, the message generation space is defined over natural language (English).

As a reminder, a POSG is defined by the tuple $\langle \mathcal{I}, \mathcal{S}, \{\mathcal{O}_i\}, \{\mathcal{A}_i\}, \mathcal{P}, \{\mathcal{R}_i\} \rangle$ where $\mathcal{I} = \{1, 2, ..., N\}$ refers to the identities of all agents in a scenario; $\mathcal{S}$ is the state space comprehensively describing the environment; $\mathcal{O}_i$ is the observation space describing agent i's view of the state; $\mathcal{A}_i$ is the action space of agent i; $\mathcal{P}$ is the state transition function $\mathcal{S} \times \mathcal{A}_1 \times \mathcal{A}_2 \times ... \times \mathcal{A}_N \to \mathcal{S}$; $\mathcal{R}_i$ is the reward function of agent i. The focal group of agents is denoted by $\mathcal{F} \subseteq \mathcal{I}$, representing a subset of all agents $\mathcal{I}$. The goal for each agent $i \in \mathcal{F}$ is to optimize a policy π_i to maximize the expected cumulative task returns of all the agents in $\mathcal{F}$, given background agent policies outside the focal group:

$$\max_{\{\pi_i\}_{i \in \mathcal{F}}} \mathbb{E}\Big[\sum_{i \in \mathcal{F}} \sum_{t=0}^{t=\infty} R_i(s_t, \mathbf{a}_t) \Big| \{\pi_j\}_{j \notin \mathcal{F}, j \in \mathcal{I}} \Big] \tag{4.1}$$

, where s_t is the state at time t, and $\mathbf{a_t} = (a_1^t, a_2^t, ..., a_N^t)$ is the joint action of all

agents at time t.

The agent's policy is structured to output both control and communication commands. Specifically, $\pi_i(O_i, \{M_j\}_{j\in\mathcal{F}}) \to \mathcal{A}_i$ maps the observation of agent i and the received messages $\{M_j\}_{j\in\mathcal{F}}$ to its action space $\mathcal{A}_i = \langle \mathcal{M}_i, \mathcal{C}_i \rangle$, where $\mathcal{M}_i$ represents the message generation space, which is constrained to natural language, and $\mathcal{C}_i$ denotes the vehicle control space with dimensions for throttle, brake, and steering inputs. At time step t, the message M_i generated by agent i is broadcast to all the connected agents within a certain communication radius, at the next time step $t+1$.

This problem presents the following technical challenges: (1) How can agents understand the situation and **generate** meaningful messages to collaboratively perceive the environment or negotiate in natural language; (2) How can agents **comprehend** incoming natural language messages and **incorporate** them into driving decision-making?

4.2 Method: LLM+Debrief

The core technical challenge of the *Talking Vehicles* problem is to enable agents to communicate in natural language in order to facilitate cooperation and act correspondingly. To establish an initial solution, we adopt an **LLM agent framework** (Figure 4.1) that prompts LLMs as a foundational prior for autonomous agents to engage in human-like communication, structuring the message within natural language space, allowing agents to interpret messages and make informed driving decisions. A key challenge of using LLMs lies in the fact that they are not specifically trained for driving tasks. To overcome this limitation, we introduce **LLM+Debrief** (Algorithm 2), a **novel multi-agent learning method for LLM agents** built upon feedback loops that allow LLM agents to iteratively refine their communication and motion control policies through trial-and-error interactions with confederate agents. Inspired by how humans reflect and debrief after a cooperative game such as Hanabi, we enable agents to discuss cooperative strategies after each

interaction episode.

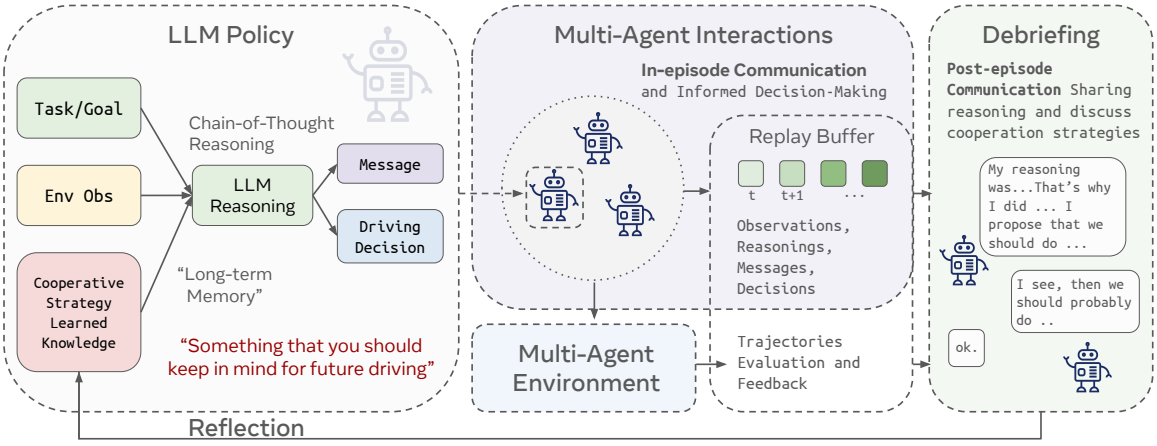


Figure 4.1: LLM+DEBRIEF agent framework and agent learning pipeline.

4.2.1 Agent Policy

An agent acts according to an LLM policy $\pi_i(O_i, \{M_j\}_{j\in\mathcal{F}}) \to \langle \mathcal{M}_i, \mathcal{C}_i \rangle$, where the distribution over actions follows the LLM used by the agent. Here, O_i represents a comprehensive text observation encompassing task and goal descriptions, environment details, and common traffic rules, expressed as a text sequence (**prompt**) $\{t_i^o\}$. A received message $M_j = \{t_j^m\}$ and a message to send $M_i = \{t_i^m\}$ are also text sequences generated by language agents. $C_i = \{t_i^c\}$ represents a text sequence for high-level commands. The joint probability of selecting a command and generating a message is expressed as $P_i(\{t_i^m\}; \{t_i^c\} | \{t_i^o\}; \{\{t_j^m\}\}_{j\in\mathcal{F}})$ where **";"** indicates text concatenation and the language model serves as the oracle to determine the probabilities.

In-Context Knowledge. Instead of fine-tuning the weights of LLMs via gradient-based methods, we adapt the policy by modifying contexts. Define $\boldsymbol{K_i} = \{\boldsymbol{t_i^k}\}$ as agent i's accumulated knowledge and $\boldsymbol{S_i} = \{\boldsymbol{t_i^s}\}$ as its cooperative strategy. The joint probability of generating commands and messages is then influenced by these additional prompt tokens: $P_i(\{t_i^m\}; \{t_i^c\} | \{\boldsymbol{t_i^k}\}; \{\boldsymbol{t_i^s}\}; \{t_i^o\}; \{\{t_j^m\}\}_{j\in\mathcal{F}})$.

Chain-of-Thought (CoT) Reasoning. Research has shown that LLMs make decisions better when provided with sufficient context (Wei et al., 2022). To leverage this observation, we prompt LLMs to reason step-by-step about the environment, incorporating observations, received messages, and in-context knowledge. The reasoning process generates an output text sequence $\boldsymbol{R_i} = \{\boldsymbol{t_i^r}\}$. Following this reasoning, the LLM agent generates structured action tokens by combining the reasoning with the inputs: $P_i(\{t_i^m\}; \{t_i^c\} | \{t_i^k\}; \{t_i^s\}; \{t_i^o\}; \{\{t_j^m\}\}_{j \in \mathcal{F}}; \{\boldsymbol{t_i^r}\})$. The final output is in a JSON format with keys: `"command"` and `"message"`.

4.2.2 Agent Learning: Post-Episode Debriefing

The learning process is depicted in Figure 4.1. Initially, the LLM agents interact with each other in the scenarios, accumulating experience, which is stored in a replay buffer. Following the interaction phase, the agents engage in a debriefing session where they utilize past experiences as context to collaboratively refine a cooperative strategy. The outcomes of these discussions are distilled into two critical components: knowledge ($K_i = \{t_i^k\}$) and cooperative strategies ($S_i = \{t_i^s\}$). These components are subsequently integrated as in-context knowledge for future interactions, playing a pivotal role in shaping and improving the policy.

Replay Buffer. We store transition data $T_i = \langle o_{i,t}, a_{i,t}, o_{i,t+1} \rangle$, which includes current and next observations, commands, messages, and reasoning in a **replay buffer**, serving as a repository for further learning and iterative refinement. When an episode concludes, the environment evaluates each agent's performance and provides scalar rewards along with **verbal feedback**, such as `"Vehicle 109 collided with Vehicle 110 after 2 seconds."` or `"Vehicle 111 stagnated for too long to complete its task."` Each transition in the replay buffer is subsequently **retrospectively labeled** with enriched metadata, including responses from other agents, collision details (e.g., time to collision), time-out details, and final rewards and outcomes.

Batch Context Sampling. Before engaging in the post-episode discussion (debriefing), each learning agent analyzes its past experience first. While analyzing the entire trajectory would provide a comprehensive understanding of failure cases, computational constraints necessitate sampling a subset (**batch**) of keyframes from its replay buffer. To prioritize relevant data, the sampling process heuristically assigns (following Equation A.1) higher probabilities to transitions that occur immediately before collisions, involve actions contributing to collisions, or lead to stagnation due to agents slowing down. Additionally, transitions that feature more intensive multi-agent interactions are given more weight. These selected samples serve as the context for subsequent analysis and strategy formulation, allowing the agent to focus on critical timesteps for improving performance.

Debriefing. A debriefing session begins when an episode concludes in failure (collision or stagnation) and is conducted in a **turn-based** manner over N rounds, with a focus on improving cooperation in future interactions. The speaking order is deterministic in this work for each session, and agents take turns speaking in a round-robin format. The agent chosen to speak first is responsible for proposing a **joint** cooperative strategy ($\boldsymbol{S}_1, \boldsymbol{S}_2, ...\boldsymbol{S}_{i\in\mathcal{F}}$) for everyone participating in the debriefing (the focal group). This agent begins by reasoning through its transition data batch, analyzing the consequences and influence on other agents of its actions, and formulating a proposed strategy. Subsequently, the other agents take turns sharing their perspectives, providing feedback, or offering alternative insights based on their analysis of their own experience batch. After the discussion, each agent summarizes the discussion to develop **individual** cooperative strategies ($\boldsymbol{S}_i$) and knowledge ($\boldsymbol{K}_i$). These outcomes serve as in-context guidelines for future instances of the same driving tasks. This joint discussion for future individual decision-making structure mirrors the principles of the Centralized Training Decentralized Execution (CTDE) framework (Bernstein et al., 2002), a widely used approach in multi-agent learning. Our implementation details are available in Appendix A.1.1.

4.3 Environment: TalkingVehiclesGym

To provide concrete and typical driving scenarios that expose the *Talking Vehicles* challenge, we have developed a simulation environment, **TalkingVehiclesGym**, which is a multi-agent gymnasium environment for the closed-loop evaluation of urban driving policies. TalkingVehiclesGym supports a flexible configuration of multi-agent scenarios, incorporating heterogeneous agents such as language agents, sensory agents, human agents, heuristic behavior agents, etc. It also enables **in-episode** communication between agents using a realistically simulated communication protocol MQTT. The simulation dynamics are built on CARLA (Dosovitskiy et al., 2017), a high-fidelity urban driving simulator. Details about the simulation framework are described in Appendix A.2.

Figure 4.2: Overview of scenarios and agent roles. **Green circles:** Focal agents, agents aim at establishing coordination through communication; **Red circles:** Potential colliders; **Blue circles:** Background agents.

Scenarios ($\mathcal{P}$) and Rewards ($\mathcal{R}$). TalkingVehiclesGym has been set up with several accident-prone scenarios (details in Table 4.1) where multi-agent communication could be beneficial[2], as shown in Figure 4.2. Scenarios labeled with

[2]Note on the Communication Mechanism. We adopt task-specific communication mechanisms in this work. For cooperative perception tasks, we use a ***parallel*** communication protocol, allowing all vehicles to transmit messages simultaneously. In contrast, for negotiation tasks, we employ a ***turn-based*** communication scheme managed by a mediator, ensuring that only one agent communicates

Table 4.1: Example scenarios. Here we describe the fundamental composition of each accident-prone scenario, where the background agents can be configured in terms of density, controlling policies, and communication capabilities.

Interaction Type	Scenario Name	Description
Cooperative Perception	Overtake	A vehicle plans to overtake a broken and stopped truck by moving into the opposite lane first and then returning to its original lane. The truck can still communicate, but the opposite-going car can not communicate.
	Left Turn Yield	A vehicle tries to turn left on a left-turn yield light when a line of trucks is blocking the view of the opposite lane. The leading truck is able to communicate.
	Red Light	A vehicle is crossing the intersection when there is another vehicle running the red light. The optical sensor failed to sense the other vehicle because of the lined-up vehicles waiting for a left turn, one of which was able to communicate.
Negotiation	Overtake	A vehicle is going to borrow the opposite lane to overtake a stopped truck. The truck is not able to connect, but an opposite-direction car can communicate.
	Highway Merge	A vehicle is going to merge onto the highway, but the target lane has continuous traffic flows. A vehicle on that lane is able to communicate and alter plans.
	Highway Exit	A vehicle is going to exit the highwa,y but it needs to cross lanes where there is a traffic flow. A vehicle in the flow is able to communicate and alter plans.

Cooperative Perception are cases where agents can benefit from receiving information about regions beyond their own line of sight and scenarios tagged with Negotiation are cases where it is necessary for the agents to discuss and resolve in their plans. In each scenario, a focal group ($\mathcal{F}$) of agents is defined. They operate alongside background agents with pre-scripted behaviors. Each focal agent is assigned a task described in natural language, with success defined as reaching its target location

at a time. This turn-based mechanism enhances stability in negotiation scenarios. In all negotiation tasks, the first-defined vehicle is designated to initiate communication.

within a time limit without collisions. Agents without motion targets, such as a stationary truck in cooperative perception tasks, do not earn rewards directly for themselves. However, the optimization objective encourages these agents to send messages that assist others in achieving their tasks.

Observation Space ($\mathcal{O}$). Our environment integrates a diverse range of sensor and simulator inputs inherited from CARLA. To focus on reasoning and multi-agent learning, we simplify environmental perception for **text-based agents** by introducing a rule-based, **partially observable captioner**. This module abstracts away the perception task, which would otherwise require object detection or vision-language models, by directly converting scenario information—such as the states of the ego vehicle and others, lane details, and road conditions—into natural language descriptions that convey **factual** information while maintaining the partial observability imposed by the agent's line-of-sight sensors. For agents equipped with a transmitter/receiver device (**transceiver**), real-time communication is enabled during episodes, and the message dialog is included as part of ther observation. An example of a text-based observation is provided in Appendix A.3.

Action Space ($\mathcal{A}$). The action space for each agent encompasses both vehicle control and communication. The control space $\mathcal{C}$ is three-dimensional, consisting of throttle, brake, and steering. To reduce decision-making frequency, agents execute high-level vehicle motion commands represented as temporal sequences of low-level vehicle controls $(C_t, C_{t+1}, ..., C_{t+k})$, where each command spans k time steps. These high-level commands are atomic actions such as `go (adapt to a target speed)`, `stop`, `slow down`, `speed up`, and `change to the left lane`. The message generation space $\mathcal{M}$ is restricted to natural language tokens in this work, but is flexible enough to support other communication modes. In this work, messages are generated alongside the high-level control commands every 0.5 seconds ($k = 10$ simulation steps).

4.4 Experiments

This section presents an empirical evaluation of LLM+DEBRIEF and baseline approaches across different cooperative driving scenarios. We investigate the following research questions:

Q1 Can LLM agents establish collaboration through chain-of-thought reasoning without prior interactions? (No.)

Q2 Does decentralized reflection enable LLM agents to improve their collaborative ability as they gain more interaction experiences? (Yes.)

Q3 Does centralized discussion among LLM agents provide additional improvements in collaboration and communication compared to decentralized reflection? (Yes.)

Q4 Can natural language communication enhance the performance and coordination of LLM agents compared to those without communication? (Only if well trained.)

Metrics. Evaluation metrics are established based on the outcomes of agents who can incur reward (reward-eligible) for their tasks in the focal group, which is scenario-specific. For a scenario with N reward-eligible agents in the focal group, evaluated over M episodes, we utilize two key metrics: 1. the **average collision rate** (**CR**), normalized by the group size, is $\frac{1}{N} \cdot \frac{1}{M} \sum_{m=1}^{M} \sum_{i \in \mathcal{F}} \mathbb{1}(\text{agent } i \text{ involved in a collision})$, where collisions may involve both focal and background agents; 2. the **average success rate** (**SR**), also normalized by the group size, is $\frac{1}{N} \cdot \frac{1}{M} \sum_{m=1}^{M} \sum_{i \in \mathcal{F}} \mathbb{1}(\text{agent } i \text{ succeeded})$. Here, $\mathbb{1}$ is the indicator function, equal to 1 if the event occurs and 0 otherwise. The remaining failure cases, where agents exceed the time limit, heuristically determined to represent the upper bound for efficient task completion, without success or collision, are captured by the **average time out rate**, which can be derived as $TR = 1 - SR - CR$.

Experimental Setup. For each baseline, We consider two settings labeled as "Silent" and "Comm". In the **"Silent"** setting, LLM agents focus solely on controlling the vehicle based on their individual perception and reasoning without communication. The **"Comm"** setting allows a method to generate either only messages or both messages and driving commands. For each LLM-based learning method, we allow agents to interact for up to 60 episodes per scenario, which is a random sequence alternating between safe (or randomized agent positions for highway negotiation settings) and accident-prone configurations. We define a "solved" criterion for learning success in a scenario as 20 consecutive successful episodes. Due to the uncontrollable randomness in the OpenAI models, we give each learning method 3 knowledge reset[3] opportunities to either report the "solved" result, otherwise the last run for each seed is considered as its output. After learning, each method is evaluated for 30 episodes per scenario configuration per seed. We report experimental results aggregated with 3 seeds.

Baselines. We established several baselines and scenarios to answer the research questions:

1. an LLM agent using Chain-of-Thought (CoT) reasoning only (**Zero-shot**),
2. an LLM agent with CoT reasoning contextualized with knowledge from decentralized reflection (**Reflection**),
3. an LLM agent that corrects past actions via self-reflection, storing these corrections in a vector-based, retrievable memory and uses few-shot retrieved ex-

[3]**Note on Knowledge Reset:** Due to the inherent stochasticity in OpenAI models during the time of our experiments, the knowledge acquired by the LLM agents may become unpredictably corrupted throughout training. Therefore, we allow each LLM agent learning method up to three *knowledge resets* (clearing the knowledge) before reaching a *solved state indicator* (defined by 20 consecutive successful episodes) or using the output policy of the final attempt after the last reset. This strategy resembles the best-of-N sampling evaluation; however, the ground-truth evaluation of the learning outcomes is costly to obtain, so our knowledge selection relies on the heuristic indicator during training.

ample augmented generation (**Correction+RAG**). The retrieval augmented method without communication (**Correction+RAG (Silent)**) adapts DiLU (Wen et al., 2023a), a non-communicating single-agent LLM-based approach that drives via reflection, to our environment. The multi-agent communication extension of DiLU, AgentsCoDriver (Hu et al., 2024), resembles the **Correction+RAG (Comm)** method, but it does not actively optimize the messages. For a fair comparison across baseline LLM agents, we do not initialize the knowledge with human data, nor is there human involvement during the learning process.

Moreover, we apply the same batch context sampling method for reflection or correction for all LLM agent baselines as our method. Additionally, we include COOPERNAUT (Cui et al., 2022), a LiDAR-based cooperative driving method, as an aspirational reference point for cooperative perception. Note that Coopernaut is not directly comparable because it processes sensory data and communicates intermediate neural representations rather than natural language.

4.4.1 Quantitative Results

Table 4.2 and Table 4.3 present the quantitative evaluation of all methods across tasks. Notably, in this proof of concept, none of the LLM methods compared operate in real-time, requiring approximately 10 real-world seconds per decision step (0.5 seconds equivalent in simulation) using `gpt-4o-mini`. The inference latency primarily depends on reasoning, but we demonstrate an approach towards real-time inference in Section 4.4.3. On average, the natural language message bandwidth remains below 300 bytes per decision step, requiring less than 0.01 Mbps communication bandwidth. Table A.1 in Appendix A.1 provides detailed latency measurements and message size statistics. Based on these results, we provide responses to the research questions posed at the start of the section.

R1: LLM agents with CoT examined in this chapter do not establish

Table 4.2: `Cooperative Perception` scenarios. `mean` ± `std` over 3 trials, each using 30 evaluation episodes.

Method \ Scenario			Overtake (Perception)		Red Light		Left Turn	
Name	LLM	Comm	CR (%) ↓	SR (%) ↑	CR (%) ↓	SR (%) ↑	CR (%) ↓	SR (%) ↑
Zero-shot	Yes	No	93.3 ± 3.4	0.0 ± 0.0	93.3 ± 6.7	6.7 ± 6.7	93.3 ± 5.8	6.7 ± 5.8
+Reflection	Yes	No	87.8 ± 3.4	0.0 ± 0.0	94.4 ± 6.9	5.6 ± 6.9	76.7 ± 20.8	23.3 ± 20.8
+Correction+RAG	Yes	No	62.0 ± 31.9	4.4 ± 7.7	93.3 ± 3.3	6.7 ± 3.3	64.4 ± 15.0	35.6 ± 15.0
Zero-shot	Yes	Yes	91.1 ± 5.1	4.4 ± 5.1	60.0 ± 11.5	38.9 ± 10.7	85.6 ± 8.4	14.4 ± 8.4
+Reflection	Yes	Yes	63.3 ± 14.5	34.4 ± 10.7	37.8 ± 18.4	47.8 ± 18.4	51.1 ± 37.2	47.8 ± 36.0
+Correction+RAG	Yes	Yes	4.4 ± 1.9	90.0 ± 6.7	13.3 ± 12.0	66.7 ± 27.3	43.3 ± 38.4	38.9 ± 22.7
+Debrief	Yes	Yes	**1.1 ± 1.9**	**94.4 ± 6.9**	**0.0 ± 0.0**	**93.3 ± 5.8**	**6.7 ± 3.3**	**92.2 ± 3.8**
Coopernaut	No	Yes	4.5 ± 3.1	90.5 ± 1.2	17.7 ± 7.8	80.7 ± 7.6	18.1 ± 6.2	80.7 ± 5.2

Table 4.3: `Negotiation` scenarios. `mean` ± `std` over 3 trials, each using 30 evaluation episodes.

Method \ Scenario			Overtake (Negotiation)		Highway Merge		Highway Exit	
Name	LLM	Comm	CR (%) ↓	SR (%) ↑	CR (%) ↓	SR (%) ↑	CR (%) ↓	SR (%) ↑
Zero-shot	Yes	No	89.9 ± 2.8	7.2 ± 3.8	100.0 ± 0.0	0.0 ± 0.0	33.3 ± 9.3	66.1 ± 9.2
+Reflection	Yes	No	32.8 ± 29.4	36.7 ± 52.1	15.0 ± 23.1	84.4 ± 22.6	32.8 ± 13.4	67.2 ± 13.4
+Correction+RAG	Yes	No	46.7 ± 21.9	33.3 ± 28.0	35.6 ± 29.4	64.4 ± 29.4	33.9 ± 28.4	51.1 ± 14.2
Zero-shot	Yes	Yes	87.8 ± 5.9	11.7 ± 6.7	67.2 ± 27.1	32.8 ± 27.1	53.3 ± 11.5	46.7 ± 11.5
+Reflection	Yes	Yes	55.6 ± 38.9	43.3 ± 37.1	20.0 ± 1.7	80.0 ± 1.7	53.9 ± 24.1	45.6 ± 23.6
+Correction+RAG	Yes	Yes	38.3 ± 6.0	61.1 ± 5.4	40.0 ± 18.0	60.0 ± 18.0	49.4 ± 49.2	43.3 ± 39.8
+Debrief	Yes	Yes	**3.3 ± 3.3**	**95.6 ± 3.8**	**6.7 ± 11.5**	**93.3 ± 11.5**	**18.3 ± 21.7**	**81.1 ± 21.2**

collaboration through communication in zero-shot interactions. Our experiments show that Zero-Shot agents (`gpt-4o-mini`), even with communication enabled, fail to coordinate effectively. The failure modes are (1) agents do not communicate effectively to understand each other's needs in perception or achieve agreement in negotiation; or (2) even when the messages make sense to humans, agents do not respond with appropriate driving commands. This result suggests that without prior training or explicit strategies, chain-of-thought reasoning alone is insufficient to foster effective coordination. Future work could examine whether reasoning models like `gpt-o4` can overcome these failures.

R2: Decentralized learning can enable LLM agents to improve their collaborative ability as they gain more interaction experiences. The decentralized learning methods, Reflection and Correction+RAG, show significant im-

provement in reducing collision rates from Zero-Shot across tasks. Reflection allows agents to individually analyze their experience to generate knowledge, but the knowledge is often more reactive than proactive (see Appendix A.4.7 for example). The Correction+RAG method records successful episodes to preserve successful coordination patterns and correct commands and messages at key frames selected through heuristic batch sampling. However, although the method improves the control response strategy, we find that it qualitatively does not always generate messages that are consistent with the actions, possibly due to the open-loop revisions. Both methods show promise but have room for improvement.

R3: Centralized debriefing enhances coordination more than decentralized reflection. The debriefing method, which focuses on generating explicit cooperation strategies, enables LLM agents to achieve more stable collaboration compared to decentralized reflection or zero-shot approaches, evidenced by higher success rates than baselines across tasks. The main performance boost comes from the formalized coordination strategy, which both defines how to communicate and how to respond given a dialogue. Interestingly, this method reveals that LLMs sometimes fail to understand complex natural language messages, so the agents eventually develop concise communication protocols (like "hold" and "go" in Appendix A.4.8) to ensure their intentions are easily interpretable among themselves. However, open challenges still remain. For example, the debriefing process can go awry where no agents can find issues with the cooperation strategies in harder and longer-horizon tasks like `negotiation-highway-exit`. We further provide detailed qualitative analysis in Section 4.4.2.

R4: Natural language communication in cooperative driving can be effective, but may pose safety risks without good communication strategies. Our method, which operates with natural language communication, provides a proof of concept for natural-language-based multi-agent coordination across scenarios. However, learning to communicate effectively remains challenging. In cooperative perception tasks, communication-enabled methods consistently outperform silent

ones, highlighting the critical role of information sharing. In contrast, in negotiation scenarios such as `highway-merge` and `highway-exit`, agents generally perform better in silent mode. This result suggests that communication adds complexity and can hinder coordination when not well-optimized. We speculate that the root cause lies in the suboptimal communication strategies learned under decentralized training, where messages may introduce noise rather than useful signals.

4.4.2 Qualitative Analysis

This section uses the `negotiation-highway-merge` scenario as an example scenario to analyze policy behaviors and qualitatively examines the learned knowledge and cooperation strategies. For detailed insights into the learned knowledge, please refer to Appendix A.4, and refer to the videos (`https://talking-vehicles.github.io/`) for comprehensive demonstrations of policy behaviors.

First, we present a recorded communication exchange between agents in a demonstrative video:

```
Vehicle 121 (on the highway): Vehicle 120, I am slowing down to
create a gap for your merge. Please proceed safely.

Vehicle 120 (merging) replied: Thank you, Vehicle 121, I will
speed up to merge into the gap you create. Please maintain your
speed to facilitate my merge.
```

This form of communication is human-interpretable, paving the way for future human participation in multi-agent collaboration. In contrast, the $(x, y, z, feature)$ latent representation generated by COOPERNAUT lacks interpretability for humans and requires all vehicles in the collaboration system share the same encoder, limiting its flexibility in mixed-autonomy settings. While in this work we do not enforce that the communication be suitable for humans to participate in the collaboration directly, the results suggest that it may be possible to move in that direction in the future by enforcing short, real-time messages.

Second, the in-context knowledge developed through the debriefing process demonstrates a **clear and coherent cooperation strategy**, defining each agent's role and their coordination mechanisms (Appendix A.4.5), in contrast to the purely reactive policies formed through self-reflection without explicit discussion of cooperation strategies (Appendix A.4.7).

Third, **agents behave according to their learned knowledge and cooperation strategy**. In the `negotiation-highway-merge` scenario, the debriefing-based policy's behavior follows the developed structured cooperation strategy: when the merging vehicle requests to enter the highway, highway vehicles explicitly slow down to create a gap, enabling a smooth and coordinated merge. In contrast, under the `Correction+RAG (Silent)` mode, the lack of a clear cooperation strategy leads to uncertainty. Both merging and highway vehicles struggle to determine the right of way, often resulting in either a collision or a prolonged, indecisive interaction at the junction. We encourage readers to watch the supplementary videos accompanying this chapter for a deeper understanding of the qualitative differences between policies.

4.4.3 Cross-Scenario Generalization and Distillation towards Real-Time

Up to this point, a separate policy was trained to handle each of the TalkingVehiclesGym scenarios. However, for practical deployment, it is desirable to develop a single policy that can handle a broad range of challenging driving scenarios. We explore two independent approaches for achieving cross-scenario generalization: **Centralized Memory** and **Distillation**.

In the **Centralized Memory** approach, we aggregate all agents' *most effective* knowledge—identified by the highest estimated success rate across learning trials—into a unified vector memory. This memory supports unified policy execution across multiple tasks by retrieving relevant experiences based on the agent's current observation.

In the **Distillation** approach, we conduct full-parameter fine-tuning of a compact language model, DistilGPT2 (Radford et al., 2019; Sanh et al., 2019), to directly **imitate** the behavior of the large memory-augmented LLM+DEBRIEF agent. The imitation dataset is constructed by aggregating all *successful* evaluation episodes across scenarios. The distillation model is trained using token-level cross-entropy loss to match the output distribution of the large model. During inference, decisions are generated through softmax sampling with a temperature of 0.2.

We evaluate the performance of each unified policy independently across different scenarios using three random seeds, reporting the mean performance and the standard error of the mean (1 SEM) in Table 4.4. For *reference*, we also report the performance statistics of the individually selected knowledge (Debrief (per-scenario)).

The Centralized Memory policy maintains strong performance across tasks. However, we observe a performance drop in overtaking scenarios (`perception-overtake` and `negotiation-overtake`). We hypothesize that this drop is due to structural and objective similarities across tasks, leading the memory to occasionally retrieve mismatched strategies. Moreover, negotiation-based tasks require proactive communication initiation by the agent, whereas perception-focused tasks do not, further exacerbating mismatches in coordination strategies. These findings raise important challenges for future research on ad hoc teamwork and generalizable communication protocols.

The Distillation model achieves decision generation times between **100 ms** and **470 ms** on an NVIDIA A40 GPU, depending on message generation length (**50 bytes** to **300 bytes**), getting close to the 500 ms decision-making frequency (as shown in Table 4.5). Remarkably, the distilled model generalizes well across scenarios and even surpasses the performance of its teacher model in some cases. We observe that it tends to behave overly conservatively in `perception-overtake` scenarios, suggesting room for further improvement, potentially through expert-guided correction methods such as DAgger (Ross et al., 2011a).

Table 4.4: Experimental results for `Generalization` across scenarios. Each policy is evaluated using three random seeds, with 30 episodes per seed. We report the mean performance over the 30 episodes, along with one standard error of the mean across seeds. Debrief (per-scenario) represents policies learned individually for each scenario and serves as an oracle baseline for comparison with the generalization performance of Centralized Memory and Distillation.

Method \ Scenario	Overtake (Perception)		Red Light		Left Turn	
	CR (%) ↓	SR (%) ↑	CR (%) ↓	SR (%) ↑	CR (%) ↓	SR (%) ↑
Debrief (per-scenario)	1.1 ± 1.1	**98.9 ± 1.1**	0.0 ± 0.0	96.7 ± 0.0	4.4 ± 2.9	94.4 ± 2.2
Centralized Memory	2.2 ± 1.1	93.3 ± 1.9	0.0 ± 0.0	**100.0 ± 0.0**	4.4 ± 2.9	93.3 ± 3.3
Distillation	**0.0 ± 0.0**	83.3 ± 1.9	0.0 ± 0.0	91.1 ± 4.4	**0.0 ± 0.0**	**96.7 ± 0.0**
Method \ Scenario	Overtake (Negotiation)		Highway Merge		Highway Exit	
	CR (%) ↓	SR (%) ↑	CR (%) ↓	SR (%) ↑	CR (%) ↓	SR (%) ↑
Debrief (per-scenario)	10.0 ± 3.8	87.2 ± 3.9	2.2 ± 2.2	97.8 ± 2.2	13.3 ± 6.0	86.7 ± 6.0
Centralized Memory	12.2 ± 2.9	86.7 ± 1.9	1.1 ± 1.1	98.9 ± 1.1	16.1 ± 4.8	82.8 ± 5.3
Distillation	**10.0 ± 3.3**	**88.9 ± 4.4**	**0.0 ± 0.0**	**100.0 ± 0.0**	**3.3 ± 0.0**	**96.7 ± 0.0**

Table 4.5: Decision latency, message size using `distilled` LLM policy

Latencies \ Scenario	Overtake	Left Turn	Red Light	Overtake	Highway Merge	Highway Exit
Decision Latency (s)	0.45	0.44	0.38	0.14	0.19	0.20
Message Size (bytes)	223.3	297.9	223.0	28.0	59.0	59.0

4.5 Related Work

This research is closely aligned with recent advancements in LLM agents for autonomous driving. Our key contribution is a novel multi-agent learning framework in which LLM-based agents learn to communicate and collaborate using natural language. Unlike prior approaches that rely on imitation from human data, our method leverages self-play interactions to develop effective communication protocols and cooperative behaviors. In the following, we provide a review of the development of LLM agents in the context of autonomous driving.

LLM Agents for Autonomous Driving. LLM agents have shown potential to address various autonomous driving tasks. In particular, they are promising in tackling corner cases (Wen et al., 2023b) due to their reasoning ability and the common-

sense knowledge embedded, yielding a more generalizable autonomous driving stack. Recent studies have explored various approaches to tailor state-of-the-art LLMs for driving (Wen et al., 2023a; Hu et al., 2024). However, a foundational challenge lies in grounding LLM agents in the real world—they need to perceive and understand the traffic scenarios. A straightforward approach is to obtain the observations from oracle perception models (Mao et al., 2023b) and convert them to textual descriptions (Mao et al., 2023a; Sha et al., 2023; Jin et al., 2023; Cui et al., 2023b). Some other studies tackled this challenge by introducing Visual Language Models (VLMs), which are adapted to driving domains through in-context instruction tuning (Ma et al., 2023) or fine-tuning (Wayve, 2023; Xu et al., 2023b; Ding et al., 2023; Yang et al., 2023). To enhance LLM agents' reasoning ability, prior works have investigated incorporating handcrafted guidance and examples in the prompts (Sha et al., 2023; Jin et al., 2023; Cui et al., 2023b), structuring the reasoning procedure (Mao et al., 2023b; Sima et al., 2023), and fine-tuning the models on driving datasets. Notably, fine-tuning LLMs and VLMs requires an extensive amount of driving data with language labels. Several approaches have attempted to adapt existing language-driving datasets for LLM fine-tuning (Ding et al., 2023; Xu et al., 2023b; Ma et al., 2023) or augment large-scale multimodal driving datasets (Caesar et al., 2020; Sun et al., 2020; Mao et al., 2021) with language labels (Qian et al., 2023; Shao et al., 2023; Sima et al., 2023; Nie et al., 2023). In contrast, our work generates scalable driving data through agent self-play. Note that existing models were predominantly evaluated in an *open-loop* fashion. In contrast, similar to some prior work (Shao et al., 2023; Sha et al., 2023; Jin et al., 2023), we conduct closed-loop evaluation of the proposed method and baseline methods in CARLA (Dosovitskiy et al., 2017). More importantly, none of the existing work has explored optimizing LLM agents in a multi-agent setting with natural language vehicle-to-vehicle communication. AgentsCoDriver (Hu et al., 2024) and LangCoop (Gao et al., 2025) represent concurrent but distinct efforts.

4.6 Summary, Limitations, and Future Work

This chapter introduces the Talking Vehicles problem, a novel setting for multi-agent autonomous driving in which vehicles communicate and coordinate using natural language. We contribute a multi-agent simulation environment, TalkingVehiclesGym, and present LLM+Debrief, a self-play learning framework that equips LLM agents with the ability to generate, interpret, and act upon natural language messages through reflective and collaborative debriefing (contributes along **Dimension A** (Communication-Supporting Representations) and **Dimension C** (Collaborate with Human-Like Agents) introduced in Chapter 1). Our experiments demonstrate that, while zero-shot LLMs fail to establish effective coordination, iterative learning via decentralized reflection and centralized debriefing substantially improves cooperative performance across both perception and negotiation scenarios. Furthermore, we show that these learned behaviors can be distilled into efficient models capable of generalizing across diverse driving tasks under real-time constraints. This study represents an important step toward integrating natural language as a universal protocol for V2V communication, bridging the gap between human-understandable coordination and autonomous decision-making.

While our work provides promising initial evidence of the potential of LLM agents and LLM+Debrief's in addressing the *Talking Vehicles* problem, it comes with several limitations and opens up rich directions for future research.

Idealized Agent Perception. In this work, the LLM agents rely on text observations, assuming an idealized perception system. This choice stems from the strong performance of current LLMs and the early-stage development of multi-modal models. However, TalkingVehiclesGym environment supports multi-modal sensory input. Future work may develop agents that integrate multi-sensor perception and reasoning to fully leverage the rich context in realistic observations.

Communication Challenges. We assume that agents intend to communicate truthfully, reliably conveying their intentions and following through on their stated decisions. However, real-world vehicle-to-vehicle communication faces numerous challenges, including time delays that result in outdated information and the risk of adversarial or deceptive messages. Future research chould develop methods to handle these challenges, ensuring timely and secure exchanges of information. Techniques such as real-time data verification and robust communication protocols will be critical for enhancing the reliability and safety of vehicle-to-vehicle communication systems.

Training Scalability and Ad Hoc Teamwork. While this chapter reports on a successful proof-of-concept, the scalability of LLM+Debrief to learn in diverse traffic scenarios and different environmental conditions has not been exhaustively tested. A limitation of our method is that, although it is sample-efficient, requiring only a few interaction episodes, the analysis must occur immediately after an episode concludes, which hinders scaling up learning. Future work could potentially scale up the training using multi-agent reinforcement learning to finetune the agents while using strong Kullback–Leibler regularization with a foundation model to ensure the agents speak human language during self-play. Additionally, future work may address ad hoc teamwork, where agents adapt to collaborators following different conventions.

Human Interface and Human Evaluation. To extend these methods to human-autonomous cooperation, intuitive and user-friendly interfaces (e.g., speech) are essential. Although our research opens up the potential for autonomous cars to cooperate with human drivers, the complexity of effective communication interfaces for humans is substantial. Comprehensive human-centered evaluations using human-friendly interfaces are deferred to future studies.

Next, we will move to the topic of generalizing to previously unseen opponents or teammates in multi-agent systems.

Part III

Learning to Generalize

Chapter 5

Generalizing to Adversarial Opponents

In this dissertation, **policy generalization** refers to the agent's ability to maintain acceptable performance when interacting with previously unseen agents, which is an essential capability for agents to adapt to the future human-AI society. To promote generalization, this dissertation adopts the philosophy of population-based training, encouraging the generation of **a diverse population of training partners** with whom the agent can interact and learn.

This chapter investigates policy generalization in adversarial settings, using **cache-timing attacks** (CTA) as a representative example (Section 5.1). Cache-timing attacks exemplify a class of computer security problems where decades of research have involved manually identifying new attacks and designing heuristic-based detectors. We aim to automate this attack–defense cycle to enhance the generalization ability of learned detectors. In contrast, Chapter 6 focuses on generalization in cooperative multi-agent settings.

Leveraging Empirical Game-Theoretic Analysis (EGTA), this chapter introduces **MACTA** (Section 5.3), a population-based method that generates increasingly diverse opponents through iterative training. It integrates Fictitious Play (Brown, 1951) with Proximal Policy Optimization (PPO) (Schulman et al., 2017) to advance the study of CTA problems. In addition, this chapter contributes a re-

alistic multi-agent simulation environment, **MA-AutoCAT** (Section 5.2), which models the interactions among attackers, benign programs, and detectors.

Experimental results (Section 5.4) demonstrate that MACTA produces a wide range of attackers that effectively camouflage themselves as benign programs. Detectors trained with MACTA exhibit strong resilience against novel, adaptive attackers, reducing the worst-case number of successful attacks by 20%.

Overall, this chapter presents a population-based multi-agent learning framework that improves generalization to unknown adversaries in the context of cache-timing attack detection. This directly contributes to **Dimension B: Multi-Agent Policy Generalization** of the core research question explored in this dissertation.

This work was published in the *Proceedings of the 11th International Conference on Learning Representations (ICLR)* in 2023. The author developed the MACTA framework and led the experimental evaluation under the supervision of Xiaomeng Yang, Wenjie Xiong, and Yuandong Tian. The author, together with Mulong Luo, Geunbae Lee, and Wenjie Xiong, contributed to the design and construction of the MA-AutoCAT environment. Mulong Luo additionally validated the discovered attack patterns on real hardware. Peter Stone, Hsien-Hsin Lee, Benjamin Lee, and G. Edward Suh provided mentorship, guidance, and in-depth feedback throughout the project.

5.1 Problem Statement: Cache Timing Attacks

The cache timing attack challenge is a fundamental problem to address as such kinds of attacks are stealthy but powerful. We introduce the domain knowledge and problem formulation in this section.

5.1.1 Domain Description

A cache is a small and fast on-chip memory device commonly used in modern processor designs to reduce latency of memory accesses. Accessing memory addresses whose data are available in a cache is fast (called a "*cache hit*"). If the data is not in the cache, data has to be retrieved from the main memory, which is much slower (called a "*cache miss*").

Surprisingly, this timing difference in memory accesses due to caching could leak information across different programs/processes executing with a shared cache, a vulnerability known as *cache timing attacks* (CTA).

As shown in Figure 5.1(a), CTA involves the attacker process and the victim process both sharing the same cache. An example (Prime+Probe CTA (Liu et al., 2015)) is given in Figure 5.1(b). The victim's memory access will evict the attacker's cache line from the cache, causing latency changes in the attacker's future memory accesses. Thus, the attacker can infer whether the victim made access to a specific *memory address* by observing its own memory access latency, and thus be able to infer the victim's private information.

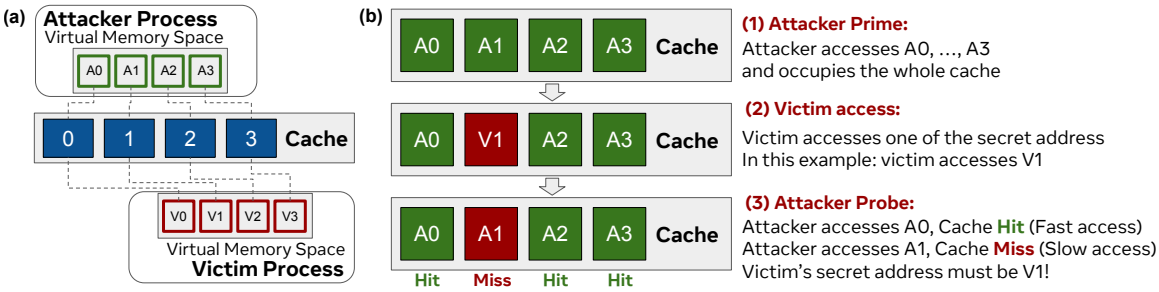


Figure 5.1: **(a)** Cache timing channel attack is formed when the attacker process and the victim process use the same locations of a shared cache for their memory accesses. **(b)** An example of Prime+Probe CTA in a 4-set direct-mapped cache. The attacker process can infer which memory address the victim process accesses by observing the latency.

5.1.2 Problem Statement

In this work, our goal is to jointly find novel attackers and robust detector policies that can generalize to unseen opponents, leading to insights for future cache design. The problem of joint learning can be formulated as a general-sum Partially Observable Stochastic Game (POSG) (Chapter 2) , where the attacker and detector have limited observations and optimize for their own cumulative return. Given the finite set of policies, the resultant attacker is the best response to a mixture of all detector policies explored, and the resultant detector is the best response to a mixture of all attacker policies explored.

Partially Observable Stochastic Games (POSGs) Formally, an n-player episodic POSG can be described using a tuple $\{\mathcal{I}, \mathcal{T}, \mathcal{S}, \mathcal{P}, \{\mathcal{A}\}_{i=1}^{n}, \{\mathcal{O}\}_{i=1}^{n}, \{\mathcal{R}\}_{i=1}^{n}, \gamma\}$, where $\mathcal{I}$ is the finite set of players, $\mathcal{T}$ is the episode length, $\mathcal{S}$ is the true state space, $\mathcal{P}$ is the state transition probability. $\mathcal{A}_i$ is the action space of player i, and the joint action space of all agents is $\{\mathcal{A}\}_{i=1}^{n} = \mathcal{A}_1 \times \mathcal{A}_2 ... \times A_n$. Similarly, $\mathcal{O}_i$ is the observation space of player i, and $\mathcal{R}_i$ is the reward function for player i. Lastly, $\gamma \in [0, 1]$ is a reward discount factor. In POSGs, each agent only has access to its own observations and actions, and its goal is to maximize the cumulative episodic reward for itself given the opponents' policies, $\mathcal{J}^i(\pi^i, \pi^{-i}) = \mathbb{E}\big[\sum_t^T \gamma^t r_t^i | s_0, a_t^i \sim \pi^i(s_t), a_t^{-i} \sim \pi^{-i}(s_t)\big]$.

A Nash Equilibrium (NE) (Chapter 2) is one solution concept to POSGs. Formally, a NE is defined as a saddle point for any player's policy π^i, we have $\mathcal{J}^i(\pi_*^i, \pi_*^{-i}) \geq \mathcal{J}^i(\pi^i, \pi_*^{-i}), \forall i \in \mathcal{N}$. Namely, given all other agents' equilibrium policies π_*^{-i}, there is no motivation for agent i to unilaterally deviate from its current policy π_*^i to achieve higher returns.

5.2 Environment: MA-AutoCAT

To study the learning dynamics of the attackers and the detectors in CTA, we develop MA-AutoCAT, a gym (Brockman et al., 2016a) environment that mod-

els realistic multi-agent CTA interactions. We build the environment based on a cache simulator, which faithfully models cache state changes, following practices in prior works on CTA detection schemes (Harris et al., 2019; Mirbagher-Ajorpaz et al., 2020). Note that experimenting detectors on real processors requires hardware modifications, which is prohibitively expensive. Figure 5.2 demonstrates the environment components and game mechanism.

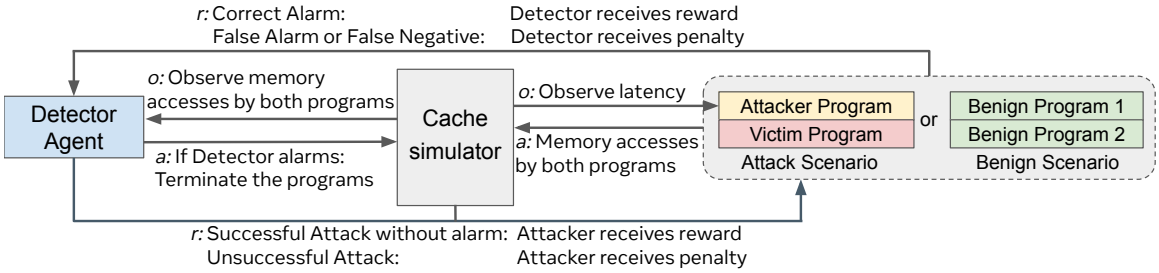


Figure 5.2: We propose MA-AutoCAT, a multi-agent environment to jointly explore and optimize the policies of the attacker and the defender processes in CTA. In this environment, multiple agents can play different roles and learn from each other. The end goal is to learn policies that can generalize to deal with previously unseen opponents (e.g., those designed by human heuristics).

In MA-AutoCAT, each agent plays a different role, and each role has a specific goal (*i.e.*, reward), a different level of privileged accessibility (*i.e.*, observation) to the information of the environment, and a different way to take actions (*i.e.*, action space), listed as below:

Benign Program (B) accesses memory in a regular way, implemented by replaying an offline log of memory accesses from regular programs (*e.g.*, a standard benchmark suite such as SPEC (Bucek et al., 2018)). It has no observation and no policy needs to be learned.

Victim (V) accesses memory with addresses that depend on a secret. Studies have shown that such secret-dependent memory accesses are common in real-world applications (*e.g.*, HTTP parser), libraries (*e.g.*, OpenSSL), and Linux kernel (Johannesmeyer et al., 2022; Qi et al., 2021; Oleksenko et al., 2020). In CTA, a victim's

secrets usually contain multiple bits, and attackers target one bit at a time; after guessing one bit of a secret, the attacker moves to the next bit. To model this in our environment, the secret bit is reset after the attacker's attempt to guess the secret and the victim accesses an address depending on the secret when triggered.

Attacker (A) aims to obtain the secret memory address of the victim process, by checking the patterns of latency of memory accesses. An attacker may learn a policy to *pick* which memory addresses to access, and observes the binary latency signal (slow/fast). The attacker can also *trigger* the victim process to execute, regain control after its execution, and *guess* the secret address of the victim if it is confident to do so. Importantly, the attacker can only see the latency of its own accesses.

Detector (D) aims to raise the alarm as soon as possible when an attacker is present while avoiding a false alarm for benign programs. As a system process, we assume that the detector can observe memory accesses to the cache sets of all running processes in the environment. The detector will *terminate* an episode if an alarm is raised.

See Appendix B.2 for detailed specifications of the observations, actions, and rewards.

In each episode, we may pick multiple agents of different roles to be in the environment and let them interact. In this work, we mainly test the following two possible scenarios:

- Attack Scenario (**DAV**). The environment contains a detector, an attacker, and a victim. The attacker aims to obtain the secret address of the victim. The detector aims to detect the presence of an attacker and terminate processes as soon as possible.
- Benign Scenario (**DBB**). The environment contains a detector and two benign programs with no malicious intent. In this case, the detector should not raise any false alarms.

We leave more complicated settings, such as scenarios with both victims and benign programs (e.g., DAVB) as future work.

5.3 Method: MACTA

The CTA that we consider is a POSG with three fundamental characteristics: **(1) Partial Observability**. In CTA, the attacker knows which program to attack but can only see the attacker's own actions and latencies, while the detector does not know if there is any attacker nor which program the attacker is targeting. **(2) Sparse-Reward Markov Game**. The CTA game can have a long episode length, and agents have to come up with a good action sequence before receiving the reward. Especially, the attacker must learn both low-level *skills* to perform attacks and high-level *strategies* to avoid defenders. **(3) Environment Randomness**. Such randomness comes from randomized victim secret addresses and the random trajectory sampling of benign programs. We propose our method based on the three crucial features.

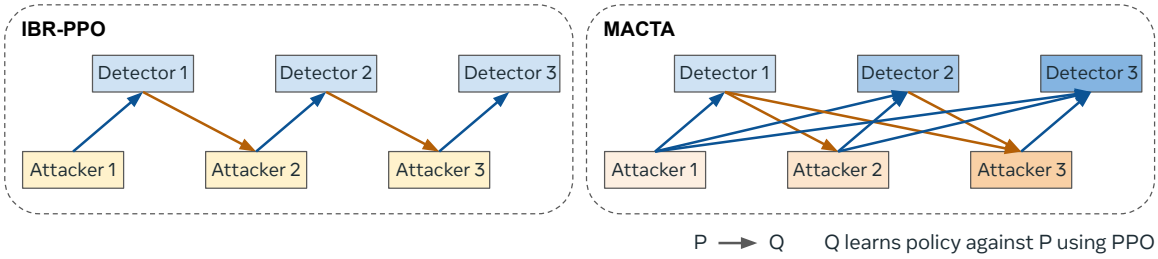


Figure 5.3: Method. Iterated Best Response PPO (IBR-PPO) learns the best response to the previous opponent only, while MACTA learns the best response to a uniform mixture of all historical opponents.

In this chapter, we introduce our approach, MACTA (Figure 5.3 Right), as an initial solution to the CTA challenge using MARL. MACTA adopts Transformers (Vaswani et al., 2017) as the neural encoder of policy nets, Proximal Policy Optimization (PPO) (Schulman et al., 2017) as the policy learning algorithm, and Fictitious Play (FP) (Brown, 1951) as the game-theoretic tool.

To deal with history-dependent partial observations and sparse rewards, both the attacker and the detector are equipped with policy nets with Transformer encoders. The Transformer encoder is mainly composed of scaled dot-product attention and multi-head self-attentions. It can effectively integrate information from long time horizons and large-scale data while not suffering from vanishing or exploding gradients in recurrent neural networks (RNNs) (Parisotto et al., 2020).

The attacker and the detector optimize their policies by the PPO algorithm to effectively learn a policy in the Markov game. Although independent reinforcement learning, where all agents are updating their policies simultaneously, is notoriously known for the instability issue in training (Tan, 1993), if we only train one agent at a time and keep others stationary, then other agents can be taken as a part of the environment, and PPO can effectively optimize the policy for higher cumulative rewards. Iterated Best Response PPO (IBR-PPO) (Figure 5.3 Left) is the most naive way of implementing the above idea. It alternates the training of the attacker and detector so that they learn the best policy against the most recent opponent. However, it may fall into the cyclic policy learning and never converge to any Nash Equilibrium (Roughgarden, 2010).

As a widely accepted method in MARL, creating a diverse pool of opponents and learning the best response to a mixture of them can alleviate the cyclic issue and help with generalization. Similar to fictitious play in game theory, we create a pool for each agent and add their historical policies to the pool. Concretely, for each iteration τ, we denote the set of policies explored until τ of agent i by our method as Π_τ^i, the opponents' joint policy set as Π_τ^{-i}. Then we learn the best response (BR), $\pi_*^i(\mathbb{U}(\Pi_\tau^{-i}))$, to the uniform mixture of the opponents' policy pool using a *best response learner* (*e.g.* PPO), and add the best response to the policy pool. Mathematically, for each iteration

$$\forall i : \Pi_{\tau+1}^i \leftarrow \Pi_\tau^i \cup \{\pi_*^i(\mathbb{U}(\Pi_\tau^{-i}))\} \tag{5.1}$$

where $-i$ represents all players except for player i, and $\mathbb{U}$ is the uniform distribution.

There are more advanced meta game frameworks like Double Oracle (McMahan et al., 2003) and Policy Space Response Oracle (Lanctot et al., 2017), which measure the meta game payoff matrix among different explored policies and solves the matrix for the best opponent mixture. In our case, since the environment contains some randomness, it is inefficient to precisely estimate the payoff matrix. We thus leave exploring more advanced game frameworks as future work.

The above components constitute our approach (Algorithm 3 in Appendix B.7). MACTA alternates the training of attacker and detector every E epochs and adds one *deterministic* policy checkpoint of the learning agent to the agent's policy pool every N epochs. During one agent's training, the agent faces a *uniform* mixture of all opponents' past deterministic policy checkpoints. Note that we create such a mixture by uniformly sampling policies from the opponent's policy pool at each action step.

Implementation Details. [1] Specifically, we start with empty policy pools for both agents, first train the attacker for 50 epochs (each epoch contains 3000 training steps) to gain the basic skills of obtaining information from the victim program, and add one policy to the attacker's policy pool every 10 epochs. Then we stop the attacker's training and switch to train the detector against the pool of the first 5 attacker policies for 50 epochs. Similarly, the detector will have 5 policies by the end of this training iteration (50 epochs). The above process is repeated until the target training iterations (1800 epochs). We adopt an Actor-Critic implementation of PPO for both the attacker and the detector, and both the policy net and the value net are 1-layer 8-head Transformer encoders with different output heads. We leverage the RLMeta (Yang et al., 2022) learning framework for the PPO implementation, which is an asynchronous version of PPO with sampling and learning in parallel, and construct our multi-agent learning framework on top of it. For stabilizing the self-play

[1] Our implementation is available at https://github.com/facebookresearch/macta.

process, we also apply dual-clip PPO (Ye et al., 2020). Refer to Appendix B.7 for a more detailed description of training and environment hyper-parameters.

5.4 Experiments

This section introduces our experiment designs to explore the following research questions:

Q1 Can MACTA generate both strong detectors and diverse attackers? (MACTA produces a wide range of attackers that effectively camouflage themselves among benign programs, while MACTA detectors demonstrate strong generalization to previously unseen threats.)

Q2 How resilient are the MACTA detectors to exploitation? (They significantly hinder the learning of new attackers and reduce the worst-case number of successful attacks by 20%.)

Q3 Which neural architecture performs best in the CTA game: MLP, LSTM, or Transformers? (Transformers.)

5.4.1 Evaluation Setup and Metrics

To evaluate the proposed MARL method, we compare with a few attacker and detector baselines. For attackers, we consider a textbook attack Prime+Probe (Algorithm 4), an RL-based attacker (AutoCAT) (Luo et al., 2023), and the PPO with Iterated Best Response Oracle (IBR-PPO) attacker. For detectors, we include our implementation of CC-Hunter (Chen and Venkataramani, 2014) and Cyclone (Harris et al., 2019) (Appendix B.8), and IBR-PPO Detector.

In this work, we employ episode return and intuitive metrics including ***Attack Correct Rate***, ***Attacks per Episode***, ***Detection Rate***, ***Episode Length***, and ***False Alarm Rate***. Details are listed in Table 5.1.

Table 5.1: Evaluation metrics.

Metrics	Object	Description
Attack Correct Rate	Attacker	Measures the ability of an attacker to infer a secret correctly (attack successfully). It is the percentage of correct guesses among all guesses aggregated over episodes.
Attacks per Episode	Attacker, Detector	Measures the speed of an attacker or the attacker's ability to bypass detection or the detector's ability to prevent attacks. It is the average number of correct guesses per episode.
Detection Rate	Attacker, Detector	Detection rate is the percentage of DAV episodes alarmed by the detector within the time limit in the evaluated DAV episodes.
Episode Length	Attacker, Detector	Measures how fast the detector can find out the existence of the attacker.
False Alarm Rate	Detector	Measures the false positive (terminate episode before time limit) rate of a detector given all benign agents.

5.4.2 Benign Dataset

We use the Standard Performance Evaluation Corporation (SPEC) 2017 benchmark suite (Bucek et al., 2018) to represent benign programs, and obtain their memory access traces using the gem5 simulator (Binkert et al., 2011). We then generate benign traces by combining the memory accesses from two programs based on the simulation timestamps. We introduce the details of the Train/Val/Test dataset in Appendix B.3.

5.4.3 Results

All the experiment results below are reported on an 8set-1way L1 cache. The attacker's memory address range is 8-15 and the victim's secret address is randomly chosen between 0-7. The episode length is 64 steps. To evaluate different methods, we report the statistics based on three independent training instances for each learning-based method and control the final policies from different instances of a method undergoing the same number of optimization steps.

5.4.3.1 Attacker Performance

Table 5.2: Attacker performance. Evaluation of the attacker's correct rate and number of attacks in 64-step episodes without detectors. Statistics are reported on three independent evaluations of 10,000 episodes.

Metrics \ Attackers	Prime+Probe	AutoCAT	IBR-PPO Attacker	MACTA Attacker
Attack Correct Rate (%) ↑	100.0 ± 0.0	100.0 ± 0.1	99.9 ± 0.1	100.0 ± 0.1
Attacks per Episode ↑	3.0 ± 0	5.2 ± 0.1	5.2 ± 0.1	4.3 ± 0.3

We first evaluate the attacker agent's performance in terms of attack correct rate and the number of attacks in an episode, to validate that the attacker agent is conducting effective attacks. Table 5.2 shows that every attacker evaluated can achieve a decent attack correct rate, indicating the agent acquires effective attack policies. In addition, the MACTA attacker has the smallest number of attacks per episode among the learning-based methods, because it learns to obfuscate itself as a benign program. Example attack sequences demonstrating the strategic attack behaviors can be found in Appendix B.4.

5.4.3.2 Head-to-Head Evaluations

In this head-to-head evaluation, we have an attacker play against a detector from *different* training instances for 10,000 episodes and report the mean detection rate and the mean episode length for all attacker and detector pairs. The head-to-head evaluation results can be found in Table 5.3 and Table 5.4. We also report the mean false alarm rate and the mean episode length of the detectors on *unseen* Benign agents in the last column of the table.

We find that the heuristic detector CC-Hunter cannot effectively discriminate the RL attackers from benign agents since the episodes are too noisy and too short to compute meaningful auto-correlations. Tuning the auto-correlation threshold only returns either a high false alarm rate or a low detection rate. The anomaly detector, Cyclone (One-Class SVM), is more effective at detecting high-bandwidth attackers such as AutoCAT and IBR-PPO attackers, yet it struggles with detecting

Table 5.3: Mean detection rate (%). Head-to-head evaluations with unseen opponents from different training instances. The higher the better for detectors when the opponent is an attacker, and the lower the better when the opponents are benign programs. '()' as Cyclone (SVM) is trained on Prime+Probe.

Detectors \ Opponents	Prime+Probe ↑	AutoCAT ↑	IBR-PPO Attacker ↑	MACTA Attacker ↑	Benign ↓
CC-Hunter (thold=0.45)	37.7 ± 0.6	13.7 ± 1.3	12.1± 0.4	16.4 ± 2.3	27.6 ± 0.9
Cyclone (One-Class SVM)	0.0 ± 0.0	55.8 ± 4.3	33.6 ± 12.8	9.0 ± 5.3	19.3 ± 0.9
Cyclone (SVM)	(**99.5** ± 0.1)	0.0 ± 0.0	0.0 ± 0.0	0.1 ± 0.1	1.4 ± 0.2
IBR-PPO Detector	0.9 ± 0.7	7.3 ± 20.5	6.4 ± 15.6	8.4 ± 21.9	**0.4** ± 0.5
MACTA Detector	**97.8** ± 0.9	**99.9** ± 0.2	**99.6** ± 0.4	**31.2** ± 18.5	1.1 ± 0.2

Table 5.4: Mean episode length (steps). Head-to-head evaluations with unseen opponents from different training instances. The lower the better for detectors when the opponent is an attacker, and the higher the better when the opponents are benign programs. Cyclone and CC-Hunter both require a fixed episode length of 64 steps.

Detectors \ Opponents	Prime+Probe ↓	AutoCAT ↓	IBR-PPO Attacker ↓	MACTA Attacker ↓	Benign ↑
IBR-PPO Detector	63.4 ± 0.4	59.6 ± 12.4	60.1 ± 9.5	58.9 ± 12.2	**63.7** ± 0.3
MACTA Detector	**16.4** ± 1.1	**12.0** ± 2.8	**12.5** ± 2.2	**50.5** ± 8.7	63.4 ± 0.1

low-bandwidth attackers like Prime+Probe and MACTA attacker and has a high false alarm rate. The SVM detector with Cyclone features is able to perform well (99.5% detection rate) on the heuristic attack (Prime+Probe) that it is trained on, but has low detection rate on RL attackers. Another drawback of these previous methods is that they require fixed-length observation that is longer than the steps needed to complete attacks (usually 12 steps in this cache configuration). IBR-PPO falls into the cyclic policy learning issue; the detector is able to react well (98.3% detection rate) to the attacker that it is trained against but fails to respond well to other attackers.

MACTA, however, is able to generalize to unseen attacks such as Prime+Probe and the IBR-PPO attacker. At the same time, MACTA also has a low false positive rate and fast detection speed which prevents further information leakage. We hypothesize that MACTA can abstract the general pattern of the attackers from interacting with diverse attacker strategies during training.

On the other hand, since the detector is trained to block all the previous

attack policies, the attacker had to explore a new policy space to evade detection. The MACTA attackers are able to evade a variety of unseen detectors. The above findings highlight the benefits of using MARL solution concepts in learning the detectors.

5.4.3.3 Exploitability Evaluations

We measure how a detector can be exploited by adaptive attackers, by fixing a detector strategy and training an RL exploiter (*i.e.*, an attacker) against the detector by dual-clip PPO from scratch. The training curve of the exploiters of MACTA detectors can be found in Figure 5.4. As the training time of the MACTA detectors increases, it becomes more difficult for an RL attacker to bypass the detectors. Specifically, it will take the RL exploiter attacker longer to find a meaningful attack strategy. And even though the RL exploiter attacker can learn to attack eventually, the number of attacks per episode decreases from around 5.0 attacks per episode (No Detector) to about 4.0 attacks per episode (MACTA-18th), leading to about 20% reduction in a learning attacker's bandwidth. The decrease in the number of attacks can come from the slower attack speed (to reduce the chance of detection) or faster detection speed so fewer attacks can be performed.

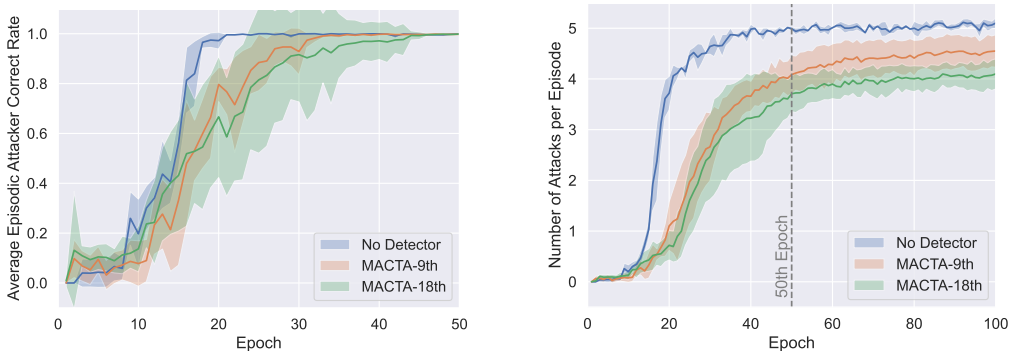


Figure 5.4: Exploitability evaluation. We fix the detector policies (No Detector, detector of 9th and 18th fictitious play iterations in MACTA (MACTA-9th, MACTA-18th)) and train an RL attacker against the detectors from scratch. **Left:** Average Episodic Attacker Correct Rate. **Right:** Attacker's Number of Attacks per episode.

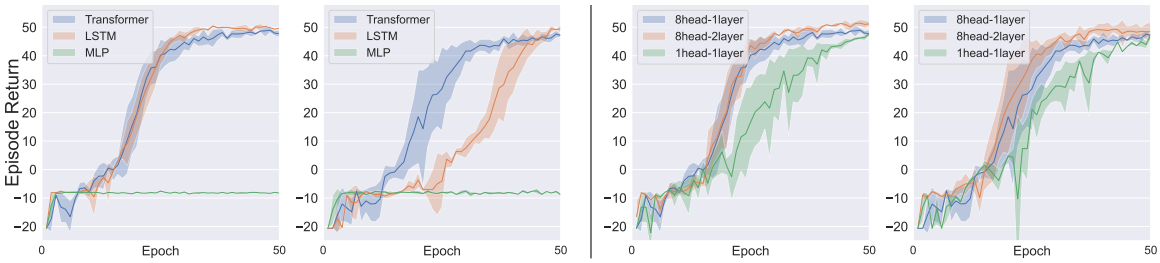


Figure 5.5: A study on neural architectures. We use a Transformer with 8-head attention and one Transformer encoder layer in MACTA experiments. **Left two:** Train attacker-only tasks using different neural architectures on two machines. **Right two:** Train attackers with different Transformer configurations on two machines.

5.4.4 Ablation Study on Neural Architecture

Our CTA task is an example where neural architecture plays a critical role in learning a meaningful policy. We train attacker-only tasks using different network architectures on different machines (details in Appendix B.6) as shown in Figure 5.5. For PPO attackers, MLP with residual connections (He et al., 2016b;a) fails to achieve a high episode return, while the Transformer and LSTM (Hochreiter and Schmidhuber, 1997) networks succeed. For Transformers, our study shows that increasing the number of encoder layers in the Transformer can slightly improve the return but is less efficient in wall time. On the other hand, reducing the number of heads slows down learning. The above evidence suggests that the sequence modeling structure is critical for CTA attack policy learning. Our hypothesis is that a successful attack is composed of a series of events, which may contain history-dependent relations among events, and Transformers can effectively model such relations. While the prior work (Luo et al., 2023) also shows that Transformers can be used for RL CTA attacker, we provide more in-depth studies on different model architectures in this work.

5.5 Related Work

This section reviews prior work on three fronts relevant to our study: (1) the design of detectors for cache timing attacks, (2) game-theoretic approaches to model-

ing security problems, and (3) population-based methods in multi-agent reinforcement learning. We highlight how our method, MACTA, differs from existing approaches by focusing on the automated discovery of novel attack and defense strategies in a realistic and dynamic environment.

Detectors for Cache Timing Attacks CC-Hunter (Chen and Venkataramani, 2014) proposes to detect cache-timing attacks using recurrent patterns generated during cache contention between attack and victim processes. More specifically, it uses autocorrelation to detect periodic interleaving between the two event trains. ReplayConfusion (Yan et al., 2016) records and deterministically replays a program's memory traces, changing the mapping of cache addresses but retaining the cadences. Executing the traces in different memory addresses can expose abnormal access patterns observed between an attacker and a victim, which do not exist in benign traces. Cyclone (Harris et al., 2019) uses cyclic interference from cache contention during an attack. This detector assigns domain tags to processes, then uses performance counters to enumerate abnormal cache contention behaviors triggered by each domain tag. PerSpectron (Mirbagher-Ajorpaz et al., 2020) trains a neural network classifier using the memory and latency event logs generated from attack examples. The follow-up work EVAX (Mirbagher-Ajorpaz et al., 2022) improves the classifier accuracy using generative adversarial networks (GAN). Existing detectors based on known attacks cannot deal with evolving attackers. Our study shows that the RL attacker can learn novel strategies to bypass existing static detectors. MACTA solves this problem by enabling auto-discovery of attacker policies.

Game Theory in Security Games Game theory provides a framework for decision-making and strategy, modeling how selfish agents interact and affect system outcomes. In Stackelberg games, a defender must first commit limited resources to protect disparate locations and an attacker that subsequently targets locations, potentially having seen the configuration of defenses (e.g., (Bier et al., 2007)). Such games have

masked systems from probes (Schlenker et al., 2018), defended systems against varied attack types (Thakoor et al., 2020), and assigned human analysts to automated system alerts (Schlenker et al., 2017). Whereas Stackelberg requires the defender to move first, we consider how the defender's policy should respond to the attacker's evolving policy. Game theory inspires GAN for security (Zolbayar et al., 2021; Baimukan and Zhu, 2021). Unlike prior works that explore adversarial samples in the neighborhood of a heuristic attack policy, our RL approach explores a broader, unknown space of attack policies with a well-defined objective. RL is an instance of stochastic games, often modeled by a Markov Decision Process. Representative studies of such games for distributed systems include threat detection and resource allocation (Krishnamurthy et al., 2007; Fan et al., 2019). To the best of our knowledge, we are the first to formulate a stochastic game for realistic, practical hardware timing attacks.

Population-based Multi-agent Reinforcement Learning Independent Reinforcement Learning in multi-agent environments suffers from the non-stationary opponent issue (Tan, 1993). While Iterated Best Response methods alleviate the above problem by learning from stationary opponents; they tend to over-fit to other players' policies and cause cycles in policy learning (Vinyals et al., 2019). Interacting with diverse opponent policies or heterogeneous agents is one effective way to avoid such cycles. Population-based MARL is thus proposed to solve large-scale extensive form games by creating a diverse pool of agents. Related work includes population-based reinforcement learning (Parker-Holder et al., 2020b), Neural Fictitious Self-Play (Heinrich and Silver, 2016), Fictitious Co-Play (Strouse et al., 2021), prioritized self-play (Vinyals et al., 2019), Double Oracle (DO) (McMahan et al., 2003) and its generalization Policy Space Response Oracle (PSRO) (Lanctot et al., 2017). The most closely related applications of population-based MARL to security games, such as those of Eghtesad et al. (2020) and Wang et al. (2019), use variants of Double Oracle, but they deal with different and less stochastic domains than ours.

5.6 Summary, Limitations, and Future Work

In conclusion, this chapter explores how game-theoretic reinforcement learning could be implemented to enable zero-shot generalization (**Dimension B of the key research question** in Chapter 1) and robustness against adaptive exploiters in the cache timing attack and detection domain. We first introduce the environment MA-AutoCAT that allows learning for both attackers and detectors, and their complex interactions with caches. Then we propose to combine the game-theoretic concept of Fictitious Play and Proximal Policy Optimization to train both agents (MACTA). Empirically, we found that the detector generated by MACTA can capture the general pattern of attacks and generalize to unseen attacks. The exploitability study of the detector also indicates the detectors can impede the learning process of adaptive attackers and slow down the attacks. On the other hand, the MACTA attacker is able to explore new policy space and mimic the benign agents to bypass the detectors. Finally, the neural architecture study demonstrates the strong representability of Transformers.

Beyond the empirical findings and their implications for practical deployment, we also identify several theoretical and algorithmic limitations that warrant further investigation.

Deployment in Real Systems. We use a cache simulator to study CTA, but we believe the trained attacker and detector can be applied to real hardware with sufficient engineering efforts. For attackers, Luo et al. (2023) demonstrates that an attack pattern learned in a cache simulator can be applied to multiple Intel processors. Similarly, we also show that the attack sequences from a MACTA attack can work on commercial processors in Appendix B.5. For the detectors, with hardware changes, the neural network model can be deployed inside a processor with a reasonable area and power overhead, as demonstrated by Mirbagher-Ajorpaz et al. (2020). Future work could study how the attacker and detector could be deployed in the real

production environment.

Convergence of the Policies. In MACTA, we adopt the Transformer-based PPO algorithm as the policy learning oracle, so there is no guarantee that the algorithm will return the best response to the opponents in limited optimization steps. Meanwhile, little previous work discusses the convergence of Fictitious Play when it is used as a game-theoretic meta solver in the general-sum MARL setting. As training continues, we observe that the detector's ability to generalize slightly diminishes, indicating that it is forgetting some past attacks. We hypothesize it can relate to the convergence of one player's policy, which causes low policy diversity in the pool.

The next chapter will introduce a method for multi-agent policy generalization by generating diverse training partners.

Chapter 6

Generalizing to Cooperative Teammates

Chapter 5 introduced a method for generating diverse **adversarial** opponents to promote generalization to previously unseen opponents. In this chapter, we shift our focus to **cooperative** scenarios. The foundation of this chapter lies in the research area of *Ad Hoc Teamwork* (AHT), where an agent must quickly adapt to cooperate with previously unseen teammates (hereafter referred to as "*unseen*" or "*novel*" teammates, or simply "teammates" when unambiguous).

This chapter explores the principles underlying the ad hoc teamwork problem. We begin by observing that a core challenge of AHT can be addressed by enabling an agent to emulate the set of all best-response policies that cover a wide range of possible teammate behaviors—referred to as the **coverage set** (Section 6.1). We then introduce the notion of a **minimum coverage set (MCS)**: the smallest set of policies such that for any possible teammate policy, there exists a best response within the MCS. We further show how this set can be approximated (Section 6.2).

Building on the concept of MCS, we propose the **L-BRDiv** algorithm, which jointly estimates the MCS of an environment and uses it to generate a diverse set of teammates for AHT training via a constrained optimization formulation (Section 6.3). The policies generated by L-BRDiv are designed to induce robust adaptation by encouraging the AHT agent to emulate responses from the MCS. We provide experimental evidence showing that L-BRDiv produces more robust AHT agents than existing

state-of-the-art teammate generation methods, while requiring fewer hyperparameters (Section 6.4).

This chapter presents a population-based, constrained-optimization learning framework that generates a diverse set of teammates that do not share the same best response. It contributes to **Dimension B: multi-agent policy generalization** of the core research question explored in this dissertation.

This work was published in the thirty-eighth *Proceedings of the AAAI Conference on Artificial Intelligence* (AAAI 2024). The author partially contributed to the theoretical derivation of the MCS concept and the design of the proposed L-BRDiv method. Muhammad Arrasy Rahman led the theoretical development and method implementation and conducted the experiments. Peter Stone provided valuable feedback for this work.

6.1 The Ad Hoc Teamwork Problem

The interaction between agents in an AHT environment can be modeled as a decentralized partially observable Markov decision process (Dec-POMDP) (the reward-sharing version of POSG in Section 2.2).

Existing AHT methods learn policies for a robust AHT agent by interacting with teammate policies from the training teammate policy set, $\Pi^{\text{train}} = \{\pi^{-1}, \pi^{-2}, \ldots, \pi^{-K}\}$. The AHT agent then optimizes its policy to maximize its returns in interactions with policies from Π^{train}. The objective of these existing AHT methods can be formalized as:

$$\pi^{*,i}(\Pi^{\text{train}}) = \underset{\pi^i}{\text{argmax}} \; \mathbb{E}_{\substack{\pi^{-i} \sim \mathbb{U}(\Pi^{\text{train}}), \\ a_t^i \sim \pi^i, \\ a_t^{-i} \sim \boldsymbol{\pi}^{-i}, P, O}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right], \tag{6.1}$$

with $\mathbb{U}(X)$ denoting a uniform distribution over set X. The learned AHT agent policy, $\pi^{*,i}(\Pi^{\text{train}})$, is then evaluated for its robustness. Given an evaluated $\pi^{*,i}(\Pi^{\text{train}})$, this robustness measure, $M_{\Pi^{\text{eval}}}\left(\pi^{*,i}(\Pi^{\text{train}})\right)$, evaluates the expected returns when the

AHT agent deals with teammates uniformly sampled from a previously unseen set of teammate policies, Π^{eval}. We formally define $M_{\Pi^{\text{eval}}}\left(\pi^{*,i}(\Pi^{\text{train}})\right)$ as the following expression:

$$\mathbb{E}_{\substack{\boldsymbol{\pi}^{-i}\sim\mathbb{U}(\Pi^{\text{eval}}), a^i_t\sim\pi^{*,i}(\Pi^{\text{train}}),\\ a^{-i}_t\sim\boldsymbol{\pi}^{-i},\, P,O}}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)\right], \tag{6.2}$$

The dependence of $\pi^{*,i}(\Pi^{\text{train}})$ on Π^{train} then implies that Equation 6.2 is also determined by Π^{train}.

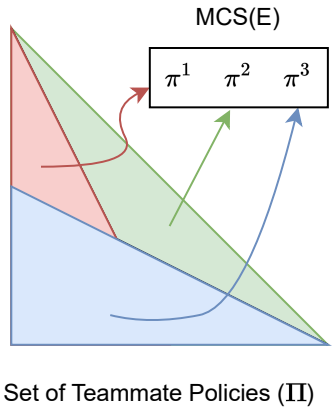
The goal of an AHT teammate generation process is to find Π^{train} producing an AHT agent policy that maximizes Equation 6.2 amid unknown Π^{eval}. Given the objective of AHT training from Equation 6.1 and the definition of the robustness measure from Equation 6.2, the objective of an AHT teammate generation process is to find the optimal set of training teammate policies, $\Pi^{*,\text{train}}$, formalized as:

$$\underset{\Pi^{\text{train}}}{\text{argmax}}\ \mathbb{E}_{\Pi^{\text{eval}}\sim\mathbb{U}(\Pi)}\left[M_{\Pi^{\text{eval}}}\left(\pi^{*,i}(\Pi^{\text{train}})\right)\right], \tag{6.3}$$

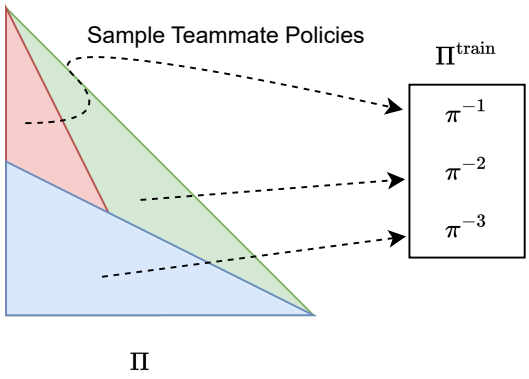
While uniformly sampling Π^{train} from Π may appear to be a reasonable solution to produce $\Pi^{*,\text{train}}$, training an AHT agent using Π^{train} may produce low returns if we only sample a limited number of policies from Π. When Π contains many possible teammate policies, the exact policies included in Π^{train} becomes important to ensure that the AHT agent is robust when collaborating with any teammate policy in Π.

6.2 Minimum Coverage Sets

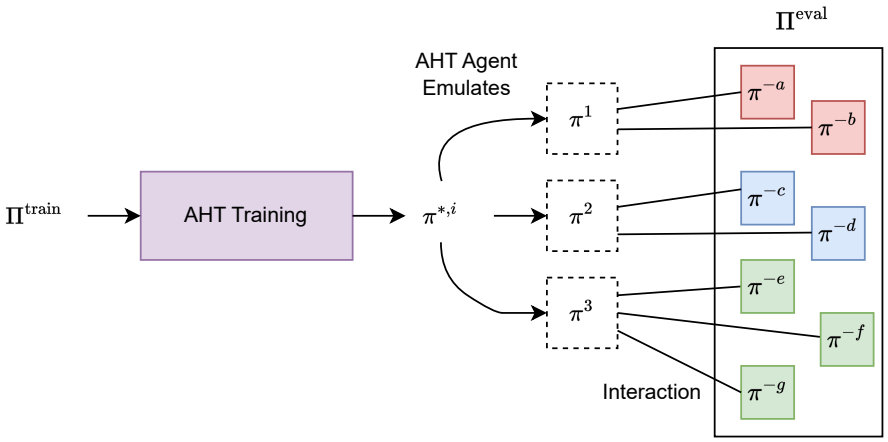
Assuming knowledge of Π^{eval}, the robustness of an AHT agent as defined by Equation 6.2 can be optimized by using Π^{eval} as teammate policies for AHT training. Given a teammate modeling component that accurately infers an unknown teammate's policy from Π^{eval} and an action selection component that can emulate any policy in the set of best-response policies to policies in Π^{eval}, $\text{BR}(\Pi^{\text{eval}})$, an AHT agent's robustness is maximized by following the best-response policy to the inferred



(a) Best-response policies to each $\pi^{-i} \in \Pi$.



(b) Generating Π^{train} based on identified best-response policies.



(c) AHT training against Π^{train} and the expected results when dealing with previously unseen teammate policies.

Figure 6.1: Leveraging MCS(E) for generating robust AHT agents. Figure 6.1a visualizes how teammate policies (points in the large triangle) can be grouped based on their best-response policies. The rectangle then shows an example MCS(E). From each subset of Π sharing the same best-response policy (colored small triangles), Figure 6.1b visualizes how one policy is sampled from each subset to create Π^{train} for AHT training. As visualized in Figure 6.1c, using our generated Π^{train} for AHT training should encourage agents that emulate the best-response policy (dashed squares) to any $\pi^{-i} \in \Pi$ when dealing with teammates from Π^{eval} (squares whose color represents its best-response policy).

teammate policy. Unfortunately, Π^{eval} being unknown makes this ideal training process impossible.

Improving an AHT agent's robustness without knowing Π^{eval} is still possible by identifying the ***coverage set*** of an environment. Denoting an environment characterized by a Dec-POMDP as E, any set containing at least one best-response policy to each teammate policy in Π is a coverage set of an environment, CS(E). CS(E) is formally characterized as:

$$\begin{aligned} &\forall \pi^{-i} \in \Pi, \forall H_t, \exists \pi^* \in \text{CS(E)} : \\ &\mathbb{E}_{s_0 \sim p_0}\left[\mathbf{R}_{*,-i}(H_t)\right] = \max_{\pi^i \in \Pi} \mathbb{E}_{s_0 \sim p_0}\left[\mathbf{R}_{,i,-i}(H_t)\right], \end{aligned} \tag{6.4}$$

where $\mathbf{R}_{i,-i}(H)$ denotes the following expression:

$$\mathbf{R}_{i,-i}(H) = \mathbb{E}_{\substack{a_T^i \sim \pi^i(.|H_T), \\ a_T^{-i} \sim \pi^{-i}(.|H_T), \\ P,O}} \left[\sum_{T=t}^{\infty} R_T(s_T, a_T) \middle| H_t = H \right]. \tag{6.5}$$

Given this definition, a CS(E) remains a coverage set when policies are added. Thus, Π itself is trivially a coverage set.

Irrespective of Π^{eval}, CS(E) will contain at least a single best-response policy to any $\pi^{-i} \in \Pi^{\text{eval}}$ since $\Pi^{\text{eval}} \subseteq \Pi$. An AHT agent capable of emulating any policy from CS(E) consequently can follow any policy from $\text{BR}(\Pi^{\text{eval}})$ for any Π^{eval}. Therefore, training an AHT agent to emulate any policy from CS(E) gives us a solution to design robust AHT agents even when Π^{eval} is unknown.

Considering CS(E) may contain policies that are not a best-response policy to any member of Π, we ideally only train AHT agents to emulate a subset of CS(E) that consists of policies that are the best-response to some $\pi^{-i} \in \Pi$. Based on this idea, we define the *minimum coverage set* of an environment, MCS(E) $\subseteq \Pi$, that is a coverage set ceasing to be a coverage set if any of its elements are removed. This characteristic of MCS(E) is formalized as:

$$\forall \pi^i \in \text{MCS(E)} : \text{MCS(E)} - \{\pi^i\} \text{ is not a coverage set.} \tag{6.6}$$

In the example provided in Figure 6.1a, $\text{MCS(E)} = \{\pi^1, \pi^2, \pi^3\}$ is an MCS since the elimination of any policy, π, from it cause a subset of Π to not have their best-response policy in $\text{MCS(E)} - \{\pi\}$.

Our work aims to design AHT agents capable of emulating any policies from MCS(E) by constructing Π^{train} in a specific way. If Π^{train} is constructed for each $\pi^i \in \text{MCS(E)}$ to have a $\pi^{-i} \in \Pi^{\text{train}}$ such that $\pi^i \in \text{BR}(\{\pi^{-i}\})$, using Π^{train} while optimizing Equation 6.1 enables us to achieve this goal. The role of MCS(E) in our teammate generation process is visualized in Figure 6.1b and Figure 6.1c.

6.3 L-BRDiv: Generating Teammate Policies By Approximating Minimum Coverage Sets

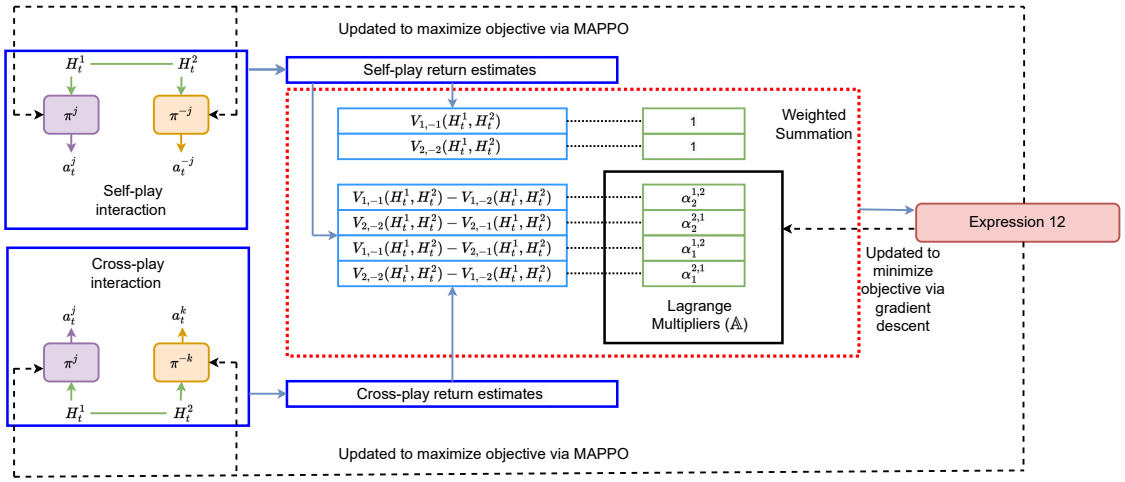


Figure 6.2: Lagrangian Best Response Diversity (L-BRDiv). The L-BRDiv algorithm trains a collection of policy networks (purple and orange boxes) and Lagrange multipliers (green cells inside the black rectangle). The purple boxes represent a policy from $\{\pi^i\}_{i=1}^K \subseteq \Pi$ while the policies visualized as an orange box is from $\{\pi^{-i}\}_{i=1}^K \subseteq \Pi$. Estimated returns between any possible pairs of policy, $(\pi^j, \pi^{-k}) \in (\{\pi^i | \pi^i \in \Pi\}_{i=1}^K \times \{\pi^{-i} | \pi^{-i} \in \Pi\}_{i=1}^K)$, and their associated Lagrange multipliers are used to compute the optimized term in the Lagrangian dual form (right red box) via a weighted summation operation (black dotted lines connect weights and multiplied terms). The policy networks are then trained via MAPPO (Yu et al., 2022) to maximize this optimized term, while the Lagrange multipliers are trained to minimize the term via stochastic gradient descent.

This section introduces our proposed teammate generation method based on estimating MCS(E). Section 6.3.1 details a constrained objective we use to estimate MCS(E). Finally, Section 6.3.2 provides a method that solves the constrained objec-

tive to jointly estimate MCS(E) while generating Π^{train}.

6.3.1 Jointly Approximating MCS(E) and Generating Training Partners

Discovering MCS(E) by enumerating the AHT agent's best-response policy to each teammate policy is intractable given the infinite policies in Π. Instead, we can estimate MCS(E) by eliminating policies from a finite CS(E) to generate MCS(E). Given a finite CS(E), an AHT agent policy is not a member of MCS(E) if it is not the best response to any teammate policy.

We check if $\pi^i \in \text{CS(E)}$ is the best-response policy of at least one policy from Π by solving the *feasibility problem*, which is the following constrained optimization problem:

$$\max_{\pi^{-i} \in \Pi} \mathbb{E}_{s_0 \sim p_0} \left[\mathbf{R}_{i,-i}(H_t) \right], \tag{6.7}$$

with the following constraints:

$$\begin{aligned} &\forall \pi^j \in (\text{CS(E)} - \{\pi^i\}) : \\ &\mathbb{E}_{s_0 \sim p_0} \left[\mathbf{R}_{j,-i}(H_t) \right] \leq \mathbb{E}_{s_0 \sim p_0} \left[\mathbf{R}_{i,-i}(H_t) \right]. \end{aligned} \tag{6.8}$$

Any CS(E) member that violates the above constraint for all $\pi^{-i} \in \Pi$ is not a member of MCS(E). While this approach relies on knowing a finite CS(E), note that knowledge of a finite CS(E) is sometimes available. For instance, the set of all deterministic policies is a finite CS(E) for environments with a finite action space and state space.

Applying the above procedure to find MCS(E) can still be impossible for two reasons. First, a finite CS(E) can be unknown. Second, the size of CS(E) may be prohibitively large, which prevents solving the feasibility problem for all $\pi^i \in \text{CS(E)}$. Amid these challenging problems, we resort to estimating MCS(E) by only discovering its subset with K policies, $\text{MCS}^{\text{est}}(\text{E}) = \{\pi^i\}_{i=1}^K$.

We now describe an alternative constrained optimization objective that jointly finds $\text{MCS}^{\text{est}}(\text{E})$ while generating a set of teammate policies for AHT training, $\Pi^{\text{train}} = \{\pi^{-i}\}_{i=1}^K$, according to the method illustrated in Figure 6.1. Two characteristics are

desired when finding $\text{MCS}^{\text{est}}(\text{E})$. First, we require each AHT agent policy from $\text{MCS}^{\text{est}}(\text{E})$ to only be the best-response policy to one teammate policy from Π^{train}, π^i. The second characteristic prioritizes the discovery of MCS(E) members that enables the AHT agent to produce high returns with a designated teammate policy, $\pi^{-i} \in \Pi$. These two requirements are formulated as the following constrained optimization problem:

$$\max_{\substack{\{\pi^i\}_{i=1}^K \subseteq \Pi, \\ \{\pi^{-i}\}_{i=1}^K \subseteq \Pi}} \sum_{i \in \{1,2,\ldots,K\}} \mathbb{E}_{s \sim p_0} \left[\mathbf{R}_{i,-i}(H_t) \right], \tag{6.9}$$

with the following constraints that must be fulfilled for all $i, j \in \{1, 2, \ldots, K\}$ and $i \neq j$:

$$\mathbb{E}_{s \sim p_0} \left[\mathbf{R}_{j,-i}(H_t) \right] + \tau \leq \mathbb{E}_{s \sim p_0} \left[\mathbf{R}_{i,-i}(H_t) \right], \tag{6.10}$$

$$\mathbb{E}_{s \sim p_0} \left[\mathbf{R}_{i,-j}(H_t) \right] + \tau \leq \mathbb{E}_{s \sim p_0} \left[\mathbf{R}_{i,-i}(H_t) \right]. \tag{6.11}$$

Note that a near-zero positive threshold ($\tau > 0$) is introduced in the constraints to prevent discovering duplicates of the same π^i and π^{-i}, which turns Constraints Equation 6.10 & Equation 6.11 into equality when $\tau = 0$.

6.3.2 Lagrangian BRDiv (L-BRDiv)

We present the **L**agrangian **B**est **R**esponse **Div**ersity (L-BRDiv) algorithm to generate Π^{train} that encourages an AHT agent to emulate $\text{MCS}^{\text{est}}(\text{E})$. L-BRDiv generates Π^{train} by solving the Lagrange dual of the optimization problem specified by Equation 6.9 and Equation 6.11, which is an unconstrained objective with the same optimal solution.

The Lagrange dual for our optimization problem is defined as:

$$\min_{\substack{\mathbb{A}\subseteq\mathbb{R}_{\geq 0}^{K(K-1)} \\ \times\mathbb{R}_{\geq 0}^{K(K-1)}}} \max_{\substack{\{\pi^i\}_{i=1}^K\subseteq\Pi, \\ \{\pi^{-i}\}_{i=1}^K\subseteq\Pi}} \Bigg(\sum_{i\in\{1,\ldots,K\}} \mathbb{E}_{s_0\sim p_0}\left[\mathbf{R}_{i,-i}(H_t)\right] + \sum_{\substack{i,j\in\{1,\ldots,K\} \\ i\neq j}} \alpha_1^{i,j}\left(\mathbb{E}_{s_0\sim p_0}\left[\mathbf{R}_{i,-i}(H_t) - \tau - \mathbf{R}_{j,-i}(H_t)\right]\right) + \sum_{\substack{i,j\in\{1,\ldots,K\} \\ i\neq j}} \alpha_2^{i,j}\left(\mathbb{E}_{s_0\sim p_0}\left[\mathbf{R}_{i,-i}(H_t) - \tau - \mathbf{R}_{i,-j}(H_t)\right]\right)\Bigg), \tag{6.12}$$

with $\mathbb{A} = \{(\alpha_1^{i,j}, \alpha_2^{i,j}) | \alpha_1^{i,j} \geq 0, \alpha_2^{i,j} \geq 0\}_{i,j\in\{1,2,\ldots,K\},i\neq j}$ denoting the set of optimizable Lagrange multipliers.

L-BRDiv learns to assign different values to Lagrange multipliers in $\mathbb{A}$ of (Equation 6.12). Optimizing Lagrange multipliers gives L-BRDiv two advantages over previous methods, which treat these hyperparameters as constants. First, we demonstrate in Section 6.4 that L-BRDiv creates better Π^{train} by identifying more members of MCS(E). Second, it does not require hyperparameter tuning on appropriate weights associated with cross-play return, which in previous methods require careful tuning to discover members of MCS(E) (Rahman et al., 2023) and prevent the generation of incompetent policies not achieving high returns against any AHT agent policy (Charakorn et al., 2023).

We provide details of the teammate generation process undergone in L-BRDiv in Algorithm 1. L-BRDiv implements the policies optimized in the Lagrange dual as neural networks trained with MAPPO (Yu et al., 2022) to maximize the weighted advantage function (Equation 6.14), whose weights correspond to the total weight associated with each expected return term in (Equation 6.12). At the same time, L-BRDiv trains a critic network to bootstrap the evaluation of (Equation 6.12) instead of a Monte Carlo approach, which can be expensive since it requires all generated policy pairs to initially follow the observation-action history, H_t. Meanwhile, the Lagrange multipliers are trained in Lines 12-13 to minimize (Equation 6.12) while ensuring it is non-negative.

Algorithm 1 Lagrangian Best Response Diversity

Require:

Cardinality of $\text{MCS}^{\text{est}}(\text{E})$ and Π^{train}, K.

Randomly initialized policy networks in $\text{MCS}^{\text{est}}(\text{E})$ & Π^{train}, denoted by $\{\pi^{i}_{\theta_i}\}^{K}_{i=1}$ & $\{\pi^{-i}_{\theta_{-i}}\}^{K}_{i=1}$ respectively.

Randomly initialized critic network $V^{j,-i}_{\theta_c}$, target $V^{j,-i}_{\theta'_c}$.

Initial values for the Lagrange multipliers, $\mathbb{A}$.

for $t_{\text{update}} = 1, 2, \ldots, N_{\text{updates}}$ **do**

  $(i, j) \sim \mathbb{U}(\{1, 2, \ldots, K\}^2)$

  $D \leftarrow \text{AgentInteraction}(\pi^{j}_{\theta_j}, \pi^{-i}_{\theta_{-i}})$

  for $(H_t, a_t, r_t, H_{t+1}) \in D$ **do**

    // Critic Optimization Step

    Update θ_c with SGD & a target critic to minimize

$$\left(V^{j,-i}_{\theta_c}(H_t) - r_t - \gamma V^{j,-i}_{\theta'_c}(H_{t+1})\right)^2 \tag{6.13}$$

    // Policy Optimization Step

    $$w^{i,j}(\mathbb{A}) \leftarrow \begin{cases} 1 + \sum_{k \neq j} \left(\alpha^{i,k}_1 + \alpha^{i,k}_2\right) & , i = j \\ -\left(\alpha^{i,j}_1 + \alpha^{j,i}_2\right) & , i \neq j \end{cases}$$

    Update θ_j and θ_{-j} with MAPPO to maximize:

$$w^{i,j}(\mathbb{A})\left(r_t + \gamma V^{j,-i}_{\theta_c}(H_{t+1}) - V^{j,-i}_{\theta_c}(H_t)\right) \tag{6.14}$$

    if $t_{\text{update}} \bmod T_{\text{lagrange}} = 0$ **then**

      // Lagrange Multiplier Optimization Step

      Update $\mathbb{A}$ using SGD to minimize Equation 6.12 where $\forall i, j \in \{1, 2, \ldots, K\}$:

$$\mathbb{E}_{s_0 \sim p_0}\left[\mathbf{R}_{j,-i}(H_t)\right] \approx V^{j,-i}_{\theta_c}(H_t) \tag{6.15}$$

      $\mathbb{A} \leftarrow \{\max(\alpha, 0) \mid \alpha \in \mathbb{A}\}$

    end if

  end for

end for

Return $\{\pi^{-i}_{\theta_{-i}}\}^{K}_{i=1}$

6.4 Experiments

In this section, we describe the environments and baseline algorithms in Section 6.4.1 and Section 6.4.2. Section 6.4.3 then details the experiment setups for evaluating the robustness of AHT agents in L-BRDiv and baseline methods via their generated training teammate policies. Finally, we present the AHT experiment results and an analysis of MCS^{est}(E) policies identified by L-BRDiv in Section 6.4.4 and Section 6.4.5.

This chapter aim to investigate and verify the following research questions:

Q1 Is L-BRDiv effective in generating meaningfully diverse teammates? (Yes.)

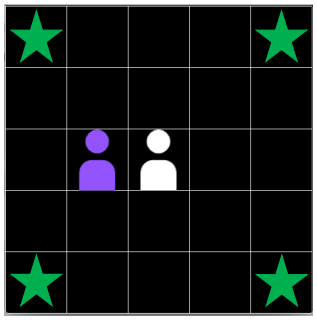
Q2 Does the diverse teammates generated by L-BRDiv facilitate fast adaptation of the AHT agents? (Yes.)

6.4.1 Environments

We run our experiments in four two-player cooperative environments. The first environment is a repeated matrix game where agents have three actions, whose reward function is provided in Figure 6.3a. Since eliminating self-sabotaging behaviour (Cui et al., 2023a) is not the focus of our work, we remove teammate-related information and actions from an agent's observation such that self-sabotaging behaviour is not a member of possibly discovered teammate behaviours, Π. We also do experiments in the Cooperative Reaching environment (Rahman et al., 2023) where two agents can move across the four cardinal directions in a two-dimensional grid world. Both agents are given a reward of 1 once they simultaneously arrive at the same corner grid. The third environment is Weighted Cooperative Reaching, which is similar to Cooperative Reaching except for a modified reward function (Figure 6.3c) that provides lower rewards if both agents arrive at different corner cells. The last environment is Level-based Foraging (LBF) (Christianos et al., 2020), where both agents must move along the four cardinal directions to a cell next to the same object and retrieve it

10	0	4
0	6	4
4	4	6

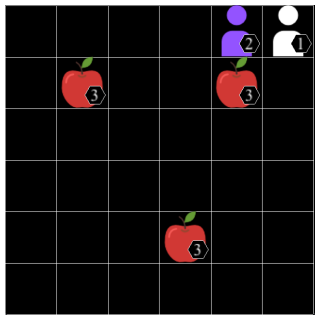
(a) Repeated Matrix Game.



(b) Coop Reaching.

	A	B	C	D
A	10	0	6	6
B	0	10	6	6
C	6	6	8	0
D	6	6	0	8

(c) Weighted Coop Reaching.



(d) LBF.

Figure 6.3: Environments for AHT experiments. We provide experiments in a repeated matrix game whose reward function is displayed in Figure 6.3a. Figure 6.3b displays an example state of the Cooperative Reaching environment where the green stars represent corner cells that provide agents rewards once they simultaneously reach it. If we start from the top-left corner cell in Figure 6.3b and assign IDs (A-D) to corner cells in a clockwise manner, Figure 6.3c shows the reward function of the Weighted Cooperative Reaching environment where agents' rewards depend on which pair of destination cells the two agents arrive at. Finally, Figure 6.3d shows a sample state of Level-based Foraging (LBF) where the apples represent the collected objects.

by simultaneously selecting actions for collecting objects. Successful object collection gives both agents a reward of 0.33.

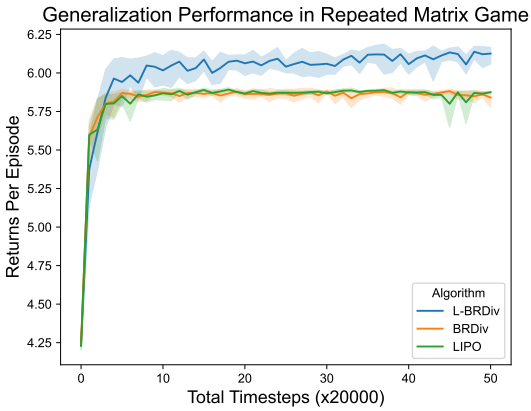
6.4.2 Baseline Methods

Our experiments compare L-BRDiv against BRDiv (Rahman et al., 2023) and LIPO (Charakorn et al., 2023). Comparing L-BRDiv and BRDiv helps investigate

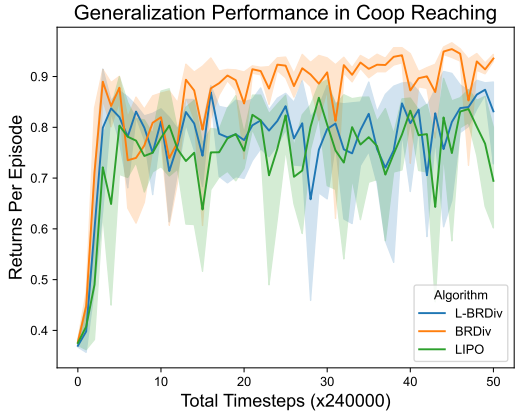
the detrimental effect of using fixed uniform weights instead of L-BRDiv's optimized Lagrange multipliers ($\mathbb{A}$). Meanwhile, including LIPO as a baseline enables us to investigate the advantage of L-BRDiv and BRDiv's use of weights with a larger magnitude for self-play maximization (i.e. $w^{i,i}(\mathbb{A})$ in Equation 6.14) compared to the weights for cross-play minimization (i.e. $w^{i,j}(\mathbb{A})$ in Equation 6.14). As justified in Section 6.5, these two policies are more appropriate baselines for L-BRDiv than any other teammate generation algorithms that we are aware of.

6.4.3 Experiment Setup

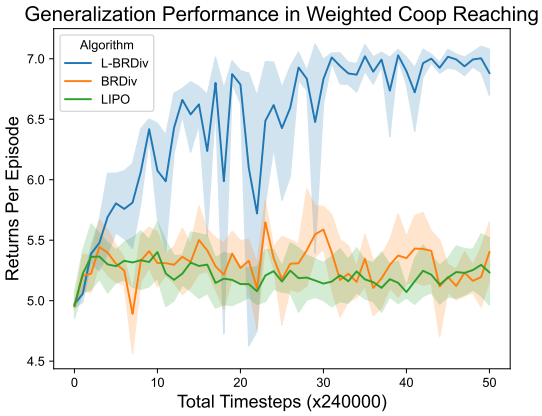
We start our experiments for each environment by generating K training teammate policies using the compared methods. We ensure fairness in our experiments by using RL^2 algorithm (Duan et al., 2016) to find an optimal AHT agent policy defined in Equation 6.1 based on Π^{train} generated by each teammate generation algorithm. Since our partially observable environments provide no useful information to infer teammate policies except for rewards obtained at the end of each interaction episode, we choose RL^2 since it can use reward information to create agent representations maintained and updated across multiple episodes. For each of the compared algorithms, the teammate generation and AHT training process are repeated under four seeds to allow for a statistically sound comparison between each method's performance. As a measure of robustness, we then evaluate the average returns of the AHT agent trained from each experiment seed when collaborating with policies sampled from Π^{eval}. We construct Π^{eval} for each environment by creating heuristic-based agents, whose behaviour we describe in Appendix C.1. Finally, we compute the mean and 95% confidence interval of the recorded returns across four seeds and report it in Figure 6.4.



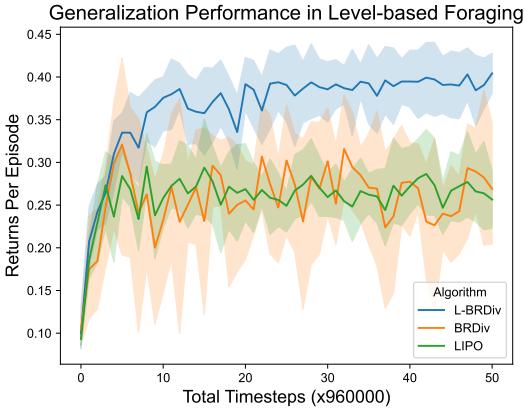
(a) Repeated Matrix Game.



(b) Coop Reaching.



(c) Weighted Coop Reaching.



(d) LBF.

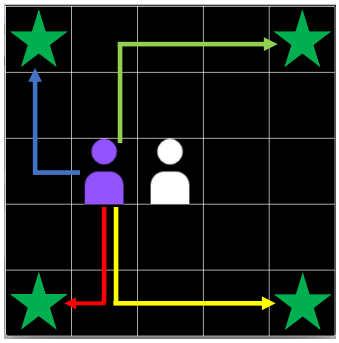
Figure 6.4: Generalization performance against previously unseen teammate types. Figure 6.4a, Figure 6.4c, and Figure 6.4d show that L-BRDiv produced significantly higher episodic returns when dealing with unknown teammate policies in all environment except for Cooperative Reaching. Figure 6.4b also show that L-BRDiv obtained episodic returns close to BRDiv's when evaluated in the Cooperative Reaching environment.

6.4.4 Ad Hoc Teamwork Experiment Results

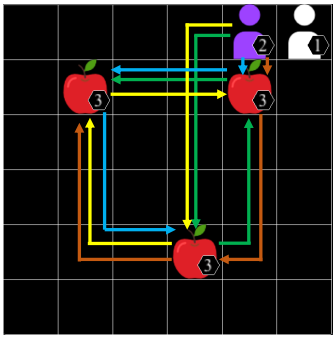
Figure 6.4 shows the results of the AHT experiments. We find that L-BRDiv significantly outperforms other compared methods in the repeated matrix game, Weighted Cooperative Reaching, and LBF. While BRDiv slightly outperforms L-BRDiv in Cooperative Reaching, overlapping confidence intervals among the last few checkpoints suggest that the difference is only marginally significant.

	π(A)	π(B)	π(C)
1	1	0	0
2	0	1	0
3	0	0	1

(a) AHT agent action selection probability for policies in MCS^{est}(E) in the Repeated Matrix Game.



(b) MCS^{est}(E) in Coop Reaching & Weighted Coop Reaching.



(c) AHT agent policies in the MCS^{est}(E) discovered for LBF.

Figure 6.5: MCS^{est}(E) yielded by L-BRDiv. L-BRDiv is capable of estimating all members of MCS(E) in all environments except LBF. Meanwhile in LBF, it discovers at least four conventions, which is still more than what LIPO and BRDiv discovered. The discovery of more MCS(E) results in L-BRDiv producing more robust AHT agents.

L-BRDiv outperforms the compared baselines in all environments except Cooperative Reaching since these environments all have reward functions that cause some members of the MCS, $\pi^i \in$ MCS(E), to yield high expected returns in cross-play interactions against a generated teammate policy, $\pi^{-j} \in \Pi^{\text{train}}$, that is not its intended partner, $\pi^{-i} \in \Pi^{\text{train}}$. Meanwhile, all $\pi^i \in$MCS(E) for Cooperative Reaching have equally low (i.e. zero) returns against the intended partner of other MCS(E) members. The large cross-play returns disincentivize BRDiv and LIPO's optimized objective from discovering π^i and π^{-i} during teammate generation. The inability to discover $\pi^i \in$ MCS(E) and π^{-i} will then lead towards diminished robustness since the trained AHT agent will yield lower returns against teammates whose best-response policy is π^i. In contrast, Cooperative Reaching's reward structure makes MCS(E) (i.e. the set of four policies moving towards each distinct corner cell) consist of policies

yielding equally low cross-play returns of zero among each other.

Although both BRDiv and LIPO are equipped with a hyperparameter, $\alpha > 0$, that can change weights associated with self-play returns maximization and cross-play returns minimization in their learning objective, it is possible to find simple scenarios where no feasible α facilitates the discovery of a desirable Π^{train} to maximize an AHT agent's robustness. Such a desirable Π^{train} is characterized by all AHT agent policies in MCS(E) having at least one teammate policy in $\in \Pi^{\text{train}}$ whom it is the best-response policy to. Appendix C.2 shows that the Repeated Matrix Game and Weighted Cooperative Reaching environment are examples of such scenarios. Even in environments like LBF where there may exist an α enabling both BRDiv and LIPO to discover a desirable Π^{train} by optimizing their learning objectives, finding an appropriate α is costly if we factor in the computational resources required to run a single teammate generation process. Unlike BRDiv and LIPO, L-BRDiv's inclusion of Lagrange multipliers as learned parameters enables it to discover desirable Π^{train} in a wider range of environments while reducing the number of hyperparameters that must be tuned.

Note that L-BRDiv and the baseline methods all successfully discover MCS(E) in Cooperative Reaching. However, each teammate policy generated by L-BRDiv and LIPO which has one of the MCS(E) members as its best-response policy ends up being less optimal than their BRDiv-generated counterparts. These suboptimal policies require more steps to complete an episode by occasionally moving away from their destination corner cell. Learning from these suboptimal agents made the AHT agent less decisive when selecting which corner cell to move towards and finally ends up producing agents with slightly lower returns.

6.4.5 Behaviour Analysis

The AHT agent policies that L-BRDiv discovers as members of MCS^{est} in all environments are provided in Figure 6.5a–Figure 6.5c. Unlike the compared baseline

methods that only discover two members of MCS(E), results from the Repeated Matrix Game show L-BRDiv is capable of consistently finding all three deterministic policies that are members of MCS(E). While all compared methods successfully discover AHT policies in the MCS(E) of Cooperative Reaching, L-BRDiv is the only method capable of finding all four members of MCS(E) corresponding to movement towards each corner grid in Weighted Cooperative Reaching. As we show in Appendix C.2, BRDiv and LIPO's failure to discover all members of MCS(E) in the Repeated Matrix Game and Weighted Cooperative Reaching is because discovering MCS(E) does not optimize their optimized objective for any constant and uniform α. In the LBF environment, none of the methods perfectly discover MCS(E) consisting of all six possible permutations of collecting objects in the environment. However, L-BRDiv is closer to estimating MCS(E) than the baseline algorithms by discovering four MCS(E) members in one seed and five MCS(E) members in the remaining seeds. Compared to the baseline methods, L-BRDiv's ability to discover more MCS(E) members eventually enables it to create more robust AHT agents that can emulate the best-response policy to a wider range of teammate policies.

6.5 Related Work

This chapter revisits Ad Hoc Teamwork (AHT) through the lens of best response diversity, a concept related to, but distinct from, adversarial diversity. We propose L-BRDiv, a novel method for generating diverse teammate policies in cooperative multi-agent systems. Here we provide an overview of the Ad Hoc Teamwork problem, Diversity in Multi-agent Learning, and Adversarial Diversity.

Ad Hoc Teamwork Existing AHT methods equip an agent with two components to achieve near-optimal performance when interacting with any unknown teammate policy (Mirsky et al., 2022). The first is a *teammate modeling component* that infers an unknown teammate's policy via observations gathered from limited interactions

with the unknown teammate. The second is an *action selection component* that estimates the best-response policy to the inferred teammate policy, which selects actions that maximize the AHT agent's returns when collaborating with an unknown teammate. PLASTIC-Policy (Barrett et al., 2016) is an early example AHT method that defines an AHT agent policy based on the aforementioned components. Recent works (Rahman et al., 2021; Zintgraf et al., 2021; Papoudakis et al., 2021; Gu et al., 2021) implement these two components as neural network models which are trained to optimize the AHT agent's returns when dealing with a set of teammate policies seen during training.

Diversity in Multi-agent Learning Introducing diversity in training partners' policies is one way to generate robust response policies in multi-agent systems. A popular line of methods leverages population-based training and frequent checkpointing (Strouse et al., 2021; Vinyals et al., 2019; Cui et al., 2023c; Bakhtin et al., 2022b). These methods rely on random seeds to find diverse policies, resulting in no guarantee that the generated policies are sufficiently diverse. Other studies optimize various types of diversity metrics directly into reinforcement learning objectives or as constraints. Xing et al. (2021) introduce a target-entropy regularization to Q-learning to generate information-theoretically different teammates. MAVEN (Mahajan et al., 2019) maximizes the mutual information between the trajectories and latent variables to learn diverse policies for exploration. Lupu et al. (2021) propose generating policies with different trajectory distributions. Trajectory diversity, however, is not necessarily meaningful for diversifying teammate policies (Charakorn et al., 2023; Rahman et al., 2023), so we do not consider these methods as baselines in our work. Other work in single-agent settings introduces Quality Diversity (Mouret and Clune, 2015; Pugh et al., 2016) or Behavior Diversity (Wu et al., 2023), which rely on domain-specific heuristics, while our method is domain-independent.

Adversarial Diversity Our research is related to work on *Adversarial Diversity* (Cui et al., 2023a; Charakorn et al., 2023; Rahman et al., 2023). These approaches generate diverse agents by maximizing *self-play* scores while minimizing *cross-play* scores in a policy pool. ***Self-play*** refers to an interaction with a designated teammate, and ***cross-play*** means playing with teammates other than the designated teammate. These approaches impose strong penalties for high cross-play returns. As a result, they may not discover teammate policies that produce high cross-play returns with other policies' best-response policies. Instead of discovering meaningfully diverse conventions, they also encourage agents to *self-sabotage* by deliberately undermining their collaboration with any policies other than the policy encountered during self-play training, as identified from their observed behaviour. Although our method resembles prior work in the optimization objective, we formulate the problem as a constrained optimization problem that allows us to generate a better set of teammate policies for AHT training. We compare our method against BRDiv (Rahman et al., 2023) and LIPO (Charakorn et al., 2023). However, we do not include ADVERSITY (Cui et al., 2023a) as a baseline since it shares the same objective as LIPO while adding methods to eliminate self-sabotaging behaviour in Hanabi (Bard et al., 2020), which we do not focus on since self-sabotaging behaviour can be desirable in other environments where a teammate's slightest deviation from a utilized convention yields low rewards.

6.6 Summary, Limitations, and Future Work

In this chapter, we propose that a suitable set of teammate policies for Ad Hoc Teamwork (AHT) training should enable agents to emulate all policies within MCS(E)—the minimum set of best-response policies needed to collaborate with any teammate policy in Π. To generate such teammate policies, we introduce and evaluate L-BRDiv, a method that formulates a constrained optimization problem and applies the Lagrangian multiplier technique to jointly approximate MCS(E) and construct

diverse teammate policies for AHT training.

Our experimental results demonstrate that L-BRDiv produces more robust AHT agents than existing teammate generation methods by identifying a broader subset of MCS(E) while eliminating the need for tuning key hyperparameters. These contributions directly support **Dimension B: Multi-Agent Policy Generalization** of the core dissertation question.

Despite these promising results, our work has several limitations and opens up important avenues for future research:

Scaling to More Agents. Currently, L-BRDiv is implemented and evaluated in two-agent grid-world settings. A valuable extension would be to scale the approach to more complex environments involving more than two agents, where coordination requires diverse joint policies and best responses.

Alternative Prioritization Criteria. The prioritization mechanism in our current approach optimizes for high self-play rewards. Future work could explore alternative objectives beyond Equation 6.9, particularly those that discourage the discovery of self-sabotaging or brittle policies (Cui et al., 2023a), enabling a more nuanced and effective approximation of MCS(E) under limited policy budgets.

Extension to General-Sum Settings. While this work focuses on fully cooperative scenarios, the notion of minimum coverage sets naturally extends to general-sum games. Expanding L-BRDiv to such settings could lead to more adaptable and robust agents. Preliminary results suggest that simply activating constraint Equation 6.11 helps uncover diverse policies even in general-sum environments, underscoring the method's broader applicability.

We now turn to the next topic: how multiple agents can collaborate and coordinate effectively with human proxies to address a real-world challenge at scale.

Part IV

Learning with Human Proxies

Chapter 7

Collaborating with Human Proxies

Collaborating effectively with humans is a critical component of AI for social good. While Chapter 4 explored how autonomous agents can learn to understand and generate *human language*, this chapter focuses on scenarios in which humans and agents coexist in the same environment, and examines how agents can learn to coordinate with *human behaviors*. In particular, we study the challenge of **mixed autonomy** in autonomous driving—where both human-driven and autonomous vehicles (AVs) share the road—and investigate how AVs can collaborate with human drivers to improve overall traffic efficiency and safety at a system level.

A well-known phenomenon in human driving is the emergence of suboptimal behaviors such as *stop-and-go waves*, which can lead to phantom congestion and reduced overall efficiency. Prior work by Vinitsky et al. (2018) demonstrated that introducing a small number of autonomous vehicles (AVs), controlled by reinforcement learning policies under centralized coordination, can dampen these oscillations and improve traffic flow. These studies modeled human drivers using proxy agents and showed that, under specific conditions, AVs have the potential to alleviate congestion and enhance system performance.

However, these earlier approaches assume a static set of agents and do not generalize well to ***open collaboration systems***, where agents dynamically enter and

leave the network. In such settings, prior methods can fail to sustain collaboration and are even vulnerable to manipulation by autonomous agents.

In this chapter, we address these limitations by developing a framework that enables AVs to collaborate with **human proxies** in open traffic systems. We propose a distributed training scheme along with a cooperative-competitive reward design tailored for an open traffic environment (Section 7.2). To study the scalability of the method in larger-scale systems, we experiment with a modular learning method that generalizes to larger and more complex traffic scenarios (Section 7.2.3). Our experimental results demonstrate that the learned policy consistently outperforms human driving baselines, illustrating the potential for scalable and robust AV-human collaboration in realistic transportation systems (Section 7.3).

This work contributes to **Dimension C: Collaborating with Human-Like Agents** by showcasing how AVs, through reinforcement learning and principled policy design, can adapt to human behavior and enhance traffic efficiency in mixed-autonomy settings.

The research presented in this chapter was published in the Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021). The author led the design and implementation of the experiments in close collaboration with William Macke and Aastha Goyal, who also contributed to the experimental work. Harel Yedidsion, Daniel Urieli, and Peter Stone were involved throughout the project, providing valuable feedback and collaborative insights.

7.1 Problem Formulation

We start by defining the traffic congestion reduction problem, its MDP (Section 2.1) formulation, and the traffic simulation environment we use.

Traffic congestion reduction problem definition. Given an open road network (as defined in the introduction) with mixed autonomy traffic consisting of both

human-driven vehicles and AVs, maximize the network's traffic efficiency by controlling the AV accelerations. Traffic Efficiency is measured in terms of **outflow** – the number of vehicles per hour exiting the network. A solution to the congestion reduction problem is a multiagent driving policy which maps the AV states to acceleration actions. We make the following assumptions: (i) Agents (AVs) are altruistic and have a common goal of reducing system congestion and (ii) Human drivers are self-interested and try to improve their own travel time.

MDP Formulation The congestion reduction problems we address in this chapter can be modeled as a discrete-time, finite-horizon Markov Decision Process (MDP) (Section 2.1). A *driving policy* is a probability distribution $\pi_\theta : \mathcal{S} \times \mathcal{A} \to [0, 1]$ parameterized by θ that stochastically maps states to driving actions.

To find a solution policy, we train an RL agent to optimize a driving policy to maximize the expected return $E_\tau \left[\sum_{t=0}^{T} R\left(S_t, A_t\right)\right]$, where $\tau := (S_0, A_0, S_1, A_1 \dots)$ denotes a trajectory, $S_0 \sim \rho_0$, $A_t \sim \pi_\theta\left(\cdot|S_t\right)$, $S_{t+1} \sim P\left(\cdot|S_t, A_t\right)$. In this chapter, $\mathcal{S}$ is a set of AV observations, $\mathcal{A}$ is a set of acceleration actions, P is computed by the simulator, and R denotes the reward function. We discuss several implementations for the reward function in Section 7.2.1

Simulation Environment We interface to the SUMO traffic simulator (Krajzewicz et al., 2012) using UC Berkeley's Flow software (Kheterpal et al., 2018). Flow provides OpenAI Gym (Brockman et al., 2016b) environments as wrappers around SUMO for easy interaction with various RL algorithm implementations. The simulator takes in maps of road structures, and simulates vehicle movements using accepted human driving models (Treiber and Kesting, 2017) and definitions of *inflows*, i.e., the location and rate of vehicles entering the network. The simulated vehicles follow safety and acceleration limits enforced by the simulator. A vehicle's *leader* and *follower* are the closest vehicles in front of and behind it (if they exist). We note that the actual inflow rate frequently differs from the requested one, for instance, in cases where vehicles

cannot enter the road network due to congestion. This mechanism opens an option for a vehicle to moderate the inflow by slowing down intentionally immediately after joining the network. The inability to guarantee exact inflows is the reason that the average-speed-based metric is not a valid congestion measure in open networks, as discussed in Section 7.2.1.

7.2 Methodology

In this section, we describe the evaluation metrics we use and the structure of the centralized and distributed multi-agent driving policies we train using RL. We discuss in detail the considerations that go into choosing appropriate *Metrics* and *Rewards*. Metrics are used for measuring the performance of a given policy, but are not always effective as RL rewards. In such cases, rewards different from the metric may be used as a performance measure for the RL agents.

7.2.1 Evaluation Metrics

In the Flow benchmark (Vinitsky et al., 2018), the performance of the system is evaluated using *time-average sample-average speed* over the episode, defined by Equation 7.1

$$\text{Time-Average Sample-Average Speed} \triangleq \frac{\sum_{t=1}^{T} \sum_{i=1}^{n_t} v_{i,t}/n_t}{T} \tag{7.1}$$

where n_t is a time-dependent variable representing the number of vehicles in the traffic network at time t, $v_{i,t}$ is the instantaneous speed of vehicle i at time t, and T is the episode length. In an ideal scenario with constant inflows, there are multiple metrics that would all lead to the same ordering of policies: maximizing average speed, maximizing network outflow, and minimizing average time delay (Dresner, 2008). In open road networks, a good policy should optimize the network outflow by maximizing the number of vehicles that pass through the network in a fixed time interval, however policies that achieve high average speed might do so through manipulations that

reduce inflows and outflows. For instance, one way to manipulate the average-speed metric is to block the incoming vehicles from entering the network until there is enough space for existing vehicles to accelerate to the maximum speed, thus maximizing the average-speed metric by compromising inflow and outflow. The average-speed metric is vulnerable because it ignores the (unmeasured) speeds of vehicles that haven't entered the simulated road network. The outflow metric on the other hand is robust to this form of manipulation since delaying vehicles from entering the network is eventually penalized through reduced outflow. Therefore, we propose *Outflow* as a performance metric in open networks as defined in Equation 7.2

$$\text{Outflow} \triangleq \frac{\sum_t^T O_t}{T} \tag{7.2}$$

where T is the episode length and O_t represents the number of vehicles that leave the network during timestep t.

The reduction of inflows and outflows as a means of improving average speed is demonstrated in Table 7.1, which compares the results of using three reward functions — the original Flow reward, the average-speed reward, and the outflow reward — on Simple Merge defined in Section 7.4.1.

7.2.2 Centralized Multiagent Driving Policy

Our centralized driving policies are built on top of the ones used in previous work (Vinitsky et al., 2018; Kreidieh et al., 2018), where a centralized RL agent trained using the Proximal Policy Optimization (PPO) algorithm (Schulman et al., 2017), controls a predefined fixed number of agents, N_{AV} as illustrated in Figure 7.1. AVs are added to the list of controlled vehicles according to a FIFO rule based on when they entered the network. Below, we discuss the state space and reward signal used for the centralized approach.

State Similarly to past work (Kreidieh et al., 2018), at time-step t the centralized driving policy accepts a state observation S_t which is a concatenation of 5-tuples

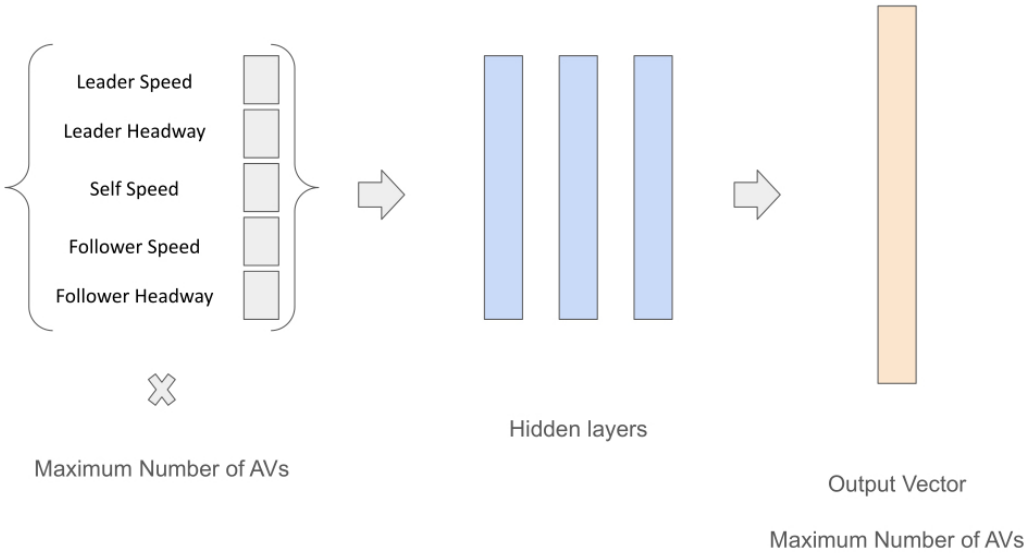


Figure 7.1: Centralized neural network policy, where local states for vehicles are concatenated to form a global state. The state is passed through a series of hidden layers, resulting in an output vector of accelerations of controlled AVs.

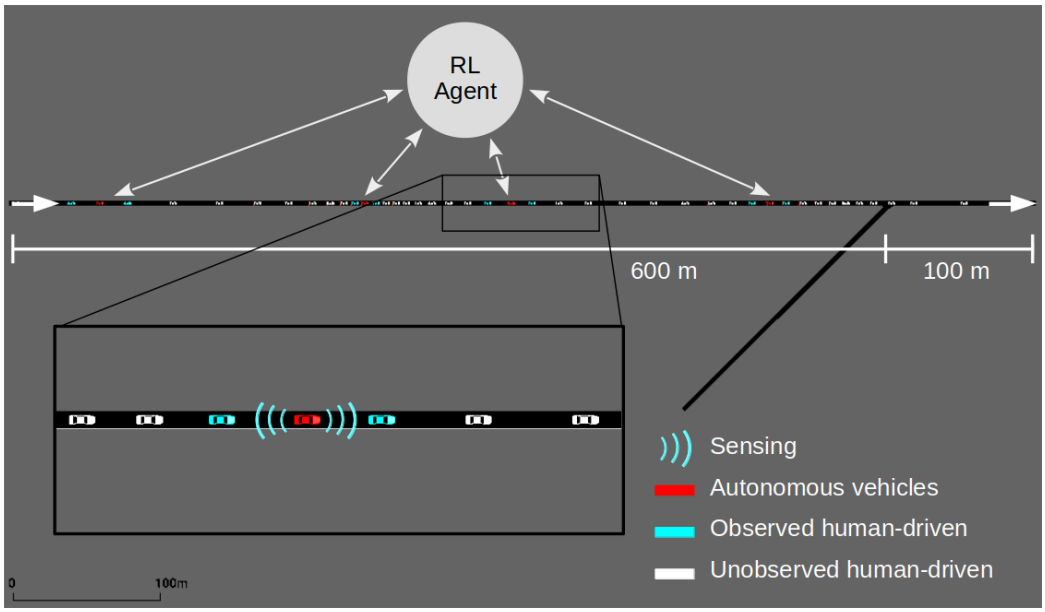


Figure 7.2: Simple Merge network of length 700 m and Inflow rate 2000 veh/hr with an on-ramp of inflow rate 200 veh/h. Perturbations caused by merging vehicles lead to stop-go waves congestion (Kreidieh et al., 2018).

representing local AV states with the following values:

1. Normalized speed of the AV_i, v_i
2. Normalized speed of the leader of AV_i, v_i^L
3. Normalized headway between AV_i and its leader, h_i^L
4. Normalized speed of the follower of AV_i, v_i^F
5. Normalized headway between AV_i and its follower, h_i^F

The speed values are normalized by the max possible speed V_{max}, and the headway values are normalized by a constant representing the maximum possible headway, h_{max}. Suppose the maximum number of AVs controlled by the centralized policy is N_{AV}, then the state feature S_t is a vector of length $5N_{AV}$, and is padded with zeros when the number of AVs in the network is smaller than N_{AV}. Formally, the state of AV_i at time t, $S_{i,t}$ is defined in Equation 7.3, and the concatenated state of all the AVs at time t, S_t is defined in Equation 7.4.

$$S_{i,t} = [\frac{v_{i,t}}{V_{max}}, \frac{v_{i,t}^L}{V_{max}}, \frac{h_{i,t}^L}{h_{max}}, \frac{v_{i,t}^F}{V_{max}}, \frac{h_{i,t}^F}{h_{max}}] \tag{7.3}$$

$$S_t = [S_1, S_2, ...S_{N_{AV}}] \tag{7.4}$$

Reward There are several possible objectives to optimize for in open networks, such as maximizing network outflow, or minimizing the maximum time delay of any vehicle to prevent starvation. In this chapter, we focus on maximizing the efficiency of a network in the form of average outflows. There are three reward functions considered in our experiments.

Original Flow Reward (Vinitsky et al., 2018) The reward in the Flow benchmark is composed of ℓ_2-norm distance to a desired velocity and a small-headway penalization term. This reward encourages every vehicle to travel as close as it can

to the desired speed every time step while maintaining a large headway.

$$r_t = \max(\|V_d \mathbb{1}^n\|_2 - \|V_d - v\|_2, 0)/\|V_d \mathbb{1}^n\|_2 - \alpha \sum_{i \in AV} \max(h_{max} - h_i, 0) \tag{7.5}$$

where v is a speed vector of all the vehicles in the network, V_d is the desired speed scalar, $\mathbb{1}^n$ is a $\mathbb{1}$ vector with n elements, where n is the total number of vehicles in the network, α is an adjustable constant, h_i is the headway between the ith AV and its leader, and h_{max} is a constant of expected headway.

Average Speed Reward We define an instantaneous average speed reward as

$$r_t = \frac{\sum_{i=1}^{n} v_i}{n V_{max}} \tag{7.6}$$

where n is the current number of all vehicles in the traffic network, and V_{max} is the maximum speed allowed on every lane. This reward is provided every time step. Summing it over the entire episode and then dividing the sum by the episode's horizon T gives the value of average speed (Equation 7.1) of the episode.

Outflow Reward The reward for instantaneous outflow is

$$r_t = O_t \tag{7.7}$$

where O_t is the number of vehicles that leave the observed area of the traffic network through any lane during the tth time step. We note that the sum of this reward over the simulation will always be proportional to the Outflow metric (Equation 7.2) by a factor of $1/T$, assuming the simulation occurs over a fixed period of time. Thus optimizing this reward is equivalent to optimizing the Outflow metric.

7.2.3 Modular Transfer Learning Approach

Scaling up to the I-696 Merge scenario results in an order of magnitude more vehicles (hundreds instead of tens). Training RL agents in this scenario is challenging

for at least three reasons. First, the state and action space grow exponentially with the number of controlled vehicles when using a centralized approach. Specifically, the combined action space of the system is, $|A| = |A_i|^{N_{AV}}$ where $|A_i|$ is the size of the action space of a single AV, and the size of the combined state space is $|S| = |S_i|^{N_{AV}}$ where $|S_i|$ is the size of the state space of a single AV. Second, while the centralized agent's most important actions are those that are near the merge point, the congestion-related rewards (Section 7.2.2) are calculated based on all vehicles in the network, most of which are more impacted by their own actions than by the centralized controller, so that the agent's reward is very noisy. Third, there is a large delay in rewards due to the delay in the effect of an AV action on the system's average speed and outflow. Therefore, in the I-696 scenario we use transfer reinforcement learning and a modular approach.

Method We create a window surrounding the junction so that the length of each road segment is comparable to a corresponding segment in a smaller network we trained on. We then take a policy that was trained in the small network, and apply it to the AVs inside the window, while outside of the window AVs act like human drivers. We refer to this approach as the *Zero-Shot Transfer* approach, and compare it to training from scratch within this same window, referred to here as *Train from scratch (Window)*.

State and Reward The states and rewards employed in the modular approach are the same as in the centralized method.

7.2.4 Distributed Multiagent Driving Policy

In our distributed setting, autonomous agents share the same policy which is executed locally as Figure 7.3 shows. Each agent only has access to its local observations, and acts independently from other agents. In the training process, each agent

receives its own reward, and the experiences of all agents are used to train the same policy.

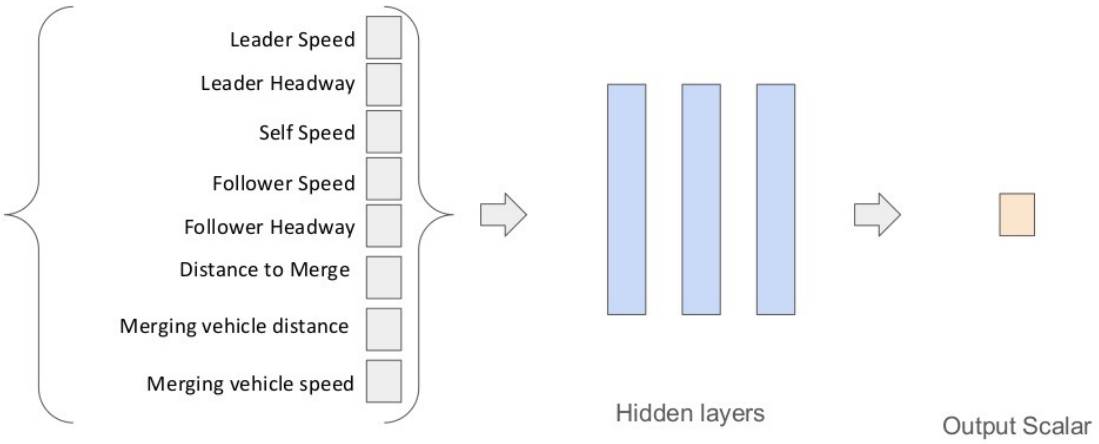


Figure 7.3: Decentralized policy, where each vehicle only has access to local observations. The local observation is passed through hidden layers, resulting in the final scalar output of the AV acceleration. This same policy is applied to every AV in the network, each with its own local observations.

State In the distributed setting, AVs rely only on their own sensed information and lack the information of the entire network that the central policy has. To mitigate this lack of information, we include both the original state features for a single AV of the centralized method as well as several additional features which can be obtained using the AV's sensors, including: distance from agent to the next merging point; the speed of the next merging vehicle and its distance to the merge junction. With this added information the state becomes:

$$S_{i,t} = [\frac{v_{i,t}}{V_{max}}, \frac{v_{i,t}^{L}}{V_{max}}, \frac{h_{i,t}^{L}}{h_{max}}, \frac{v_{i,t}^{F}}{V_{max}}, \frac{h_{i,t}^{F}}{h_{max}}, \frac{d_{next}}{d_{max}}, \frac{v_{merge}}{V_{max}}, \frac{d_{merge,next}}{d_{max}}] \tag{7.8}$$

where d measures the length along the predefined route in m, v denotes speed in m/s, h denotes headway in m, V_{max} is the max possible speed, d_{max} is a constant

that evaluates the length from the network entry to the merging junction, and h_{max} is a constant representing the max possible headway. In particular, v_i, v_i^L, v_i^F denotes speed of the ith AV at time step t, its nearest leader's speed, and its nearest follower's speed respectively; h_i^L and h_i^F denote headway between the ith AV and its leader, and the headway between the ith AV and its follower; d_{next} represents the distance between the ith AV and the next junction on its route, and $d_{merge,next}$ is the minimal distance of all vehicles on a different edges to the junction. Since the policy is shared among all agents in the traffic network, the number of AVs can vary among different environments, and the theoretical maximum value that i can take is the maximal number of vehicles allowed in the observed traffic network.

Reward The reward design in the distributed setting is different from that in the centralized setting, since the agent only gets rewards while it is in the simulation. The Outflow reward is only affected by the AV's actions after the AV had exited the simulation, so the agent does not get to observe its own rewards. The Average Speed reward does not encourage the agents to exit the simulation, since higher rewards (speeds) result in spending less time in the simulation and therefore lower cumulative rewards. As a result the agents may reduce their speed without incurring a substantial reduction in their return. A possible alternative is to penalize an agent for every time step it stays in the network (i.e. for agent i, the reward at time t would be $r_{i,t} = -0.1$). We refer to this as *selfish* reward. We found in preliminary experiments that if agents are only rewarded according to individual time-delay, their learned policy is inferior to a policy trained using a combination of selfish and collaborative reward in distributed shared policy training.

We use η_1 to denote the weighting of the individual time-delay penalty (selfish component), and η_2 to denote the weighting of the system average speed (collaborative component), where $\eta_1 + \eta_2 = 1, \eta_1 \geq 0, \eta_2 \geq 0$.

The effectiveness of mixing reward in this way is consistent with previous work on multiagent reward mixing (Durugkar et al., 2020). Our preliminary experiments

also show that a bonus for each agent upon leaving the traffic network helps, so the final reward used in our distributed approach is defined as:

$$r_{i,t} = (1 - \mathbb{1}done)(-\eta_1 + \eta_2 \times \frac{\sum_{j=1}^{n} v_j}{nV_{max}}) + \mathbb{1}done \cdot Bonus \tag{7.9}$$

Where $\mathbb{1}done$ is an indicator function that takes value 1 at the time the agent vehicle leaves the network, and 0 when the vehicle is still in the network. We perform sensitivity analysis on the values of coefficients of the reward function in Table 7.4

7.3 Experiment Setup

In this section we describe the properties of the two types of road networks that we use in our empirical evaluation, the Simple Merge, and the I-696 highway. We also describe the characteristics of the two types of vehicles in the network, human driven vehicles, and AVs.

We aim to answer the following research questions through the empirical evaluations:

Q1 What is the best evaluation metric for the traffic congestion reduction in the open road network? (Outflow.)

Q2 Does the distributed learning method reduce traffic congestion more effectively with better state representations? (Yes.)

Q3 Does the modular transfer approach scale to the larger scenario? (Yes.)

Q4 What is the best mixed reward for the distributed learning agents? (A mixture of a system reward and individual reward can be found through grid search.)

7.3.1 Traffic Scenario 1 - The Simple Merge

Our Simple Merge experiments are based on the Flow benchmark (Vinitsky et al., 2018). The road network consists of a main highway of length 600m before

the merge and a merging lane of 200m. After merging, the vehicles still need to travel an additional 100m. The junction controller is a "priority" controller where both incoming edges have equal priority. This controller is the same as in previous benchmarks (Vinitsky et al., 2018). If two vehicles arrive at the junction at the same time with equal priority, the one with the lower speed will yield to the vehicle with the higher speed. The main highway has an inflow of 2000 vehicles per hour consisting of 90% humans and 10% AVs. The merging lane has an inflow of 200 vehicles per hour, made up entirely of human drivers. This setup is compatible with real highway capacity levels of 2250 vehicles/hour/lane (Laufer, 2007). For both inflows, vehicles enter the traffic network according to the predefined inflow rate with some small stochastic variance in the arrival times. In the mixed autonomy traffic flow, the AVs are equally spaced among human vehicles. In the centralized policy the maximum number of controlled AVs, N_{AV}, is 5. In the distributed policy there is no limit on the number of controlled AVs.

7.3.2 Traffic Scenario 2 - The I-696 Merge

The I-696 network has the same shape as the real-world Interstate 696 highway in the US, which is a much larger network than the simple merge. In our experiments, we simplified the I-696 network to have a single rather than multiple lanes, a main road, and a single merging road as highlighted in Figure 7.4. We refer to this part of the network as the I-696 Merge. The I-696 Merge is much longer than the Simple Merge, which makes it challenging for existing methods to learn effective driving policies. The highway length before the merge is 3131m, the merging edge length is 1878.56m, and after merging the vehicle still needs to travel 5077.7m. The defined traffic inflows are the same as in the Simple Merge.

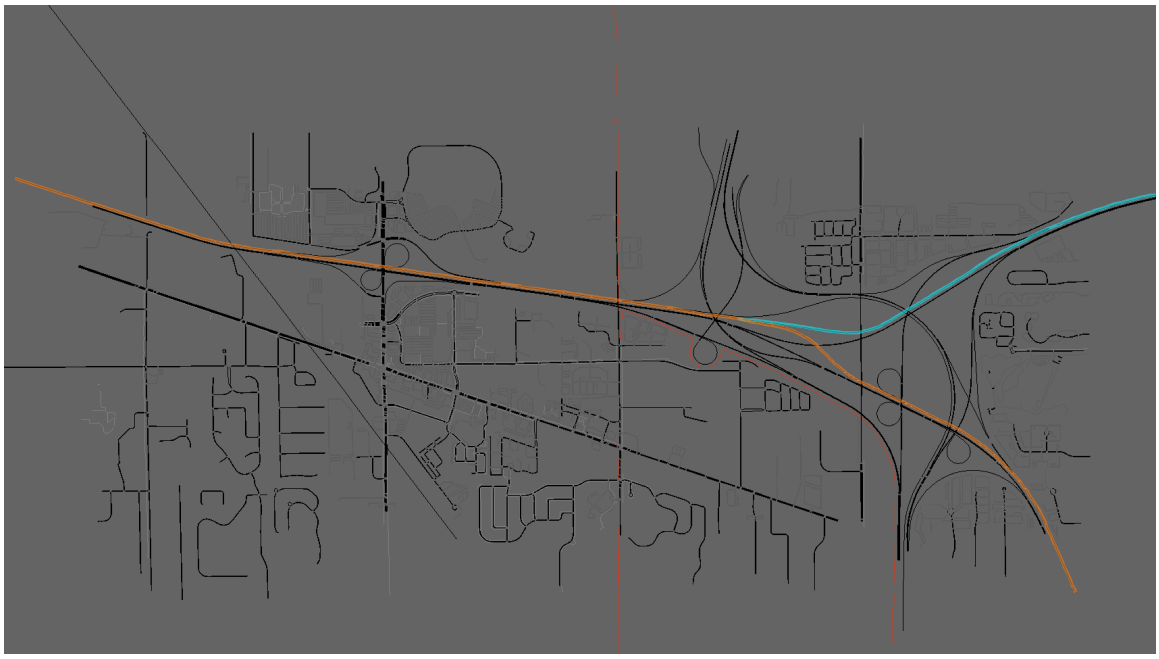


Figure 7.4: I-696 Network

7.3.3 Human-Proxy Vehicles

In all scenarios, the behavior of human-driven vehicles is modeled using the *Intelligent Driver Model* (IDM) (Treiber et al., 2000), which seeks to maintain a target speed while ensuring a minimum time gap of one second from the leading vehicle. This model has been widely adopted to simulate human driving behavior in traffic studies.

7.3.4 Autonomous Vehicles (AV)

Autonomous vehicles are only included in the main highway inflow with a 10% penetration rate and equal spacing. There are at most 5 controlled AVs in centralized Simple Merge, 30 in the centralized I-696 Merge, and any number in the distributed Simple Merge.

7.3.5 Training Details

All experiments are trained with the same set of parameters using the Proximal Policy Optimization (PPO) algorithm. Both tasks were trained in an episodic manner with a horizon of 2000 time steps of length 0.5 seconds. All results are obtained from SUMO 1.6.0 and Ray 1.0.1. [1]

7.4 Empirical Results

In all our experiments, for each configuration we execute three policy learning runs, select the learned policy with the highest return, and evaluate its performance in 100 simulations using a fixed set of 100 random seeds (which affect the arrival times of the vehicles entering the network). We report the mean values of relevant metrics accompanied with their 95% confidence interval error bounds. [2]

7.4.1 Comparison of Reward Functions

Table 7.1 shows the results of training the centralized policy described in Section 7.2.2 in the Simple Merge scenario described in Section 7.3.1, using each of the three reward functions described in Section 7.2.2 (along with human driven traffic as a baseline).

Table 7.1: Performance of different reward functions on Simple Merge

Reward	**Average Outflow** (vehs/hr)	**Average Inflow** (vehs/hr)	**Average Speed** (m/s)
Human	1558.12±2.99	1725.48±2.89	7.27± 0.15
Original Flow Reward	1724.55±6.98	1769.36±6.60	18.95±0.19
Average Speed Reward	1379.45±2.99	1408.46±3.28	**19.34**±0.02
Outflow Reward	**1804.21**±7.17	**1864.55**±7.24	16.21±0.08

[1] Our code base is publicly available here: https://github.com/cuijiaxun/MITC-Project
[2] Videos can be found here: https://www.cs.utexas.edu/~aim/flow.html

All reward functions — the original Flow reward, the average-speed reward, and the outflow reward — result in improved average speed over the human baseline, where the average speed reward results in the highest average speed in the network. However, we see that this improvement comes at the cost of overall reduced network throughput, even when compared with the human baseline. The Average Speed reward produced network inflows and outflows that were significantly lower than the human baseline (an independent T-Test yields p-values < 0.001 for both metrics). By contrast, both the Flow reward function and the outflow reward function are able to increase all 3 metrics compared to the human baseline; however, the Outflow reward function still outperforms the Flow reward in terms of outflow and inflow by a statistically significant margin. An independent T-Test yields p-values < 0.001 for both inflows and outflows.

7.4.2 Modular Transfer Learning

In this section we compare the performance of three RL approaches and human-driven traffic on the I-696 Merge scenario (Section 7.3.2):

- Zero-Shot Transfer approach (Section 7.2.3)
- Train from scratch (Window) approach (Section 7.2.3)
- Train from scratch (N_{AV}=30) approach — trained on the entire I-696 Merge with a maximal number of controlled AV, $N_{AV} = 30$, and applied to up to 30 AVs in I-696 Merge.

All three approaches were trained using two reward functions: the Flow reward, and the Outflow reward. Table 7.2 demonstrates that the Zero-Shot Transfer approach integrated with the outflow reward produces the best outflow results: significantly better than human performance. The next best approach is the Train From Scratch in a window approach, in combination with the Outflow reward. However the difference between these top two approaches is not statistically significant with p-value=0.141

in an independent T-Test. The top two approaches also have better average speed than the human baseline and comparable inflows. The Train from scratch (N_{AV}=30) approach performs worse than the human baseline under both reward functions, as it was unable to address the three challenges described in Section 7.2.3. The original

Table 7.2: Evaluation results of transferring a policy from Simple Merge to I-696 Merge

Experiment	**Reward**	**Average Outflow** (vehs/hr)	**Average Inflow** (vehs/hr)	**Average Speed** (m/s)
Human	None	936.90±5.96	2184.91±0.29	16.27±0.12
Train From Scratch (N_{AV}=30)	Outflow	366.98±1.91	561.60±3.12	19.57±0.27
	Flow reward	638.06±10.99	1165.06±10.22	14.95±0.12
Train From Scratch (Window)	Outflow	**1012.64**±9.23	2178.76±2.81	16.99±0.13
	Flow reward	923.29±5.79	2181.17±1.92	15.98±0.10
Zero-Shot Transfer (Window)	Outflow	**1017.32**±10.49	2170.55±4.61	17.05±0.16
	Flow reward	928.00±6.06	2181.53±1.67	16.09± 0.11

Flow reward never beats the human baseline in terms of outflows, in any of the training approaches.

Note that outflows in I-696 are approximately half of those in Simple Merge due to the length of I-696, since vehicles take a long time to reach the end of the simulated highway. Since the simulation on I-696 is much slower than on Simple Merge, training on I-696 takes approximately 5 times longer for the same number of iterations than training on Simple Merge.

7.4.3 Distributed Setting

In this section we perform feature selection on the distributed approach's state representation, and conduct sensitivity analysis on the hyper-parameters of the distributed reward function.

Distributed State Feature Augmentation The centralized agent receives local observations sent from all autonomous vehicles, which can provide indirect information indicating the traffic situation at different locations over the network. For

Table 7.3: Evaluation results of distributed method using different features

Augmentation	**Episodic Return**	**Average Outflow** (vehs/hr)	**Average Inflow** (vehs/hr)	**Average Speed** (m/s)
Human	Not Applicable	1558.12±2.99	1725.48±2.89	7.27± 0.15
No Augmentation	458.09±9.65	1610.68±10.56	1658.45±10.64	16.02±0.17
Full Augmentation	**476.81**±14.47	**1791.07**±6.60	1850.72±6.76	15.91±0.05
Dist	447.64±13.26	1663.49±10.95	1725.55±9.94	14.57±0.31
Dist+MergeInfo	444.95±13.33	1674.72±9.72	1741.90±7.76	14.40±0.22
MergeInfo	434.43±8.21	1600.67±9.28	1657.01±9.34	15.04±0.14
Congestion+MergeInfo	456.05±13.18	1666.55±14.97	1726.24±14.99	14.92±0.22
Congestion+Dist	425.87±4.17	1686.24±6.49	1755.50±7.39	13.53±0.06

example, the speed of first AV may represent the congestion level ahead of the second AV, so the second AV can adjust its behavior according to this. In our distributed setting, however, no information is communicated between AVs, so the agents have to make decisions solely based on local information.

One intuition is that AVs can make better decisions if they are provided with system-level information. The following environmental information is hypothesized to be useful for the distributed agents to learn a policy that can improve traffic efficiency. For simplicity, we will use the abbreviations in parentheses for each feature for future reference.

1. Average speed of vehicles between the AV and the next junction (Congestion)
2. Distance from the AV to the next junction (Dist)
3. Distance from the first vehicle that is going to merge to the junction and the speed of this merging vehicle (MergeInfo)

We show in Table 7.3 that if the agents totally rely on the speed and headway information of itself, its leader, and its follower without any extra information (referred to as "No Augmentation"), they can achieve slightly better results than

the human baseline, but additional state information further improves the performance. The experiments were conducted using a "0.1-Collaborative reward" (the reward defined in Equation 7.9 with $\eta_1 = 0.9$, $\eta_2 = 0.1$) and a bonus for completion ($r_{i,done} = +20$). We noticed that when including the congestion feature, the learning process became less stable, with average return decreasing later in the training process. We conjecture that this happens because the observed values of this feature are highly dependent on the AVs' current policy. For instance, during early training, the agents may experience mainly high congestion values, while once the policy improves, it sees mainly low congestion values. Therefore including this feature adds an additional non-stationary element to the learning process. Table 7.3 shows how state augmentation affects the training process. The model with the full set of state features (Dist, MergeInfo, and Congestion) performed the best.

Table 7.4: Experiment results of different reward function parameters of the distributed method on Simple Merge

$\eta_1, \eta_2, r_{i,done}$	**Average Outflow** (vehs/hr)	**Average Inflow** (vehs/hr)	**Average Speed** (m/s)
Human	1558.12±2.99	1725.48±2.89	7.27± 0.15
$\eta_1 = 1, \eta_2 = 0, +0$	1749.78±7.78	1807.99±7.93	16.01±0.06
$\eta_1 = 1, \eta_2 = 0, +20$	1771.74±5.63	1831.75±5.98	15.91±0.04
$\eta_1 = 0.9, \eta_2 = 0.1, +20$	**1791.07**±6.60	1850.72±6.76	15.91±0.05
$\eta_1 = 0.9, \eta_2 = 0.1, +0$	1622.34±6.74	1685.02±6.86	13.99±0.06
$\eta_1 = 0.8, \eta_2 = 0.2, +20$	**1796.76**±6.78	1856.70±7.07	16.03±0.05
$\eta_1 = 0.7, \eta_2 = 0.3, +20$	1740.64±5.14	1801.58±5.30	15.38±0.04
$\eta_1 = 0.5, \eta_2 = 0.5, +20$	1750.46±6.51	1809.83±6.72	15.59±0.23
$\eta_1 = 0, \eta_2 = 1, +20$	1744.96±6.69	1808.28±7.07	13.11±1.09
$\eta_1 = 0, \eta_2 = 1, +0$	271.44±6.29	566.64±3.96	1.54±0.02

Distributed Reward - Parameter tuning Next, we compare the resulting performance when learning policies with the distributed reward defined in Equation 7.9,

parameterized with different values of η_1 (selfish), η_2 (collaborative) and $Bonus$ (completion).

In Table 7.4 we can see that when the reward is purely collaborative ($\eta_1 = 0$, $\eta_2 = 1$,+0), it does not encourage agents to leave the system, since the outflow of 271.44 is about 17% of the human baseline outflow. In fact, the longer an agent stays in the system, the more reward it can get at the early training stage. As a result, the policy optimization can become trapped at a local optimum. Visualization of the resulting policy shows that at some point an AV stops and lets the merging vehicles travel at full speed so as to gain more speed-based reward.

We see that by either moving to a fully selfish reward, or adding an exit bonus reward $r_{i,done} = 20$ when the ith agent has exited the simulation, the outflow improves by about 12% compared with the human baseline (an outflow of 1749.78 or 1744.96), and by using both fully selfish reward and the exit bonus the outflow improves by additional 1.5% (an outflow of 1771.74). An additional improvement of $1.3\% - 1.6\%$ is achieved by mixing a small fraction of global reward with a large fraction of selfish reward, as well as an exit bonus (an outflow of 1791.07 and 1796.76 for the 0.2- and 0.1-collaborative policies). We note, however, that due to the high variance in performance during training, the RL policies didn't always achieve these results during the training process. Generally, when the collaboration weight η_2 is less than 0.5 and there is an exit bonus, the trained policies can all increase the average speed and outflow with respect to the human baseline, without significantly lowering the inflow.

The distributed policy outperforms the human baseline, but achieves slightly lower outflows than the centralized one. The difference between the top performing distributed policy and the centralized one is not statistically significant.

7.5 Related Work

Traffic congestion has long been an active research area (Downs, 2000). A common form of traffic jam in freeways is *stop-and-go waves*, which were shown in field experiments to emerge when density exceeds a critical value, even with no apparent bottleneck (Sugiyama et al., 2008). In recent field experiments, hand-designed controllers dissipated such waves and improved traffic flow (Stern et al., 2018).

RL for Mixed Autonomy. The recent industry-wide development of autonomous and automated vehicles (AVs) has led to a surge of interest in harnessing AVs to reduce traffic congestion. On the theoretical side, there have been efforts to formalize and analyze the foundations for AVs impacting traffic systems (Wu et al., 2018). On the applied side, large-scale traffic simulators have been adopted into a newly developed experimental framework called Flow (Wu et al., 2017; Vinitsky et al., 2018), which we use in this chapter. Using Flow, past research showed that Reinforcement Learning (RL) (Sutton and Barto, 2018) can learn an effective centralized multiagent driving policy, which simultaneously senses and controls all AVs, and improves the average traffic speed over human-driven traffic, implemented with accepted human driving models (Treiber and Kesting, 2017). However, we show that the average-speed metric is manipulable by an RL agent and might not accurately reflect the network's traffic efficiency. Instead, we propose using an alternative metric.

Transfer Learning in RL. Since using RL to learn controllers in realistic simulated or real-world setups could be impractically slow, some research looked at using transfer learning (Taylor and Stone, 2009) to expedite learning, by transferring from a simulated ring to a simulated simple merge scenario (Kreidieh et al., 2018), and from a simulated to a scaled city (Jang et al., 2019). Our transfer learning approach is different in the modular way it reuses state representation, which makes it more scalable. In the ring-to-merge transfer, the authors handled the different source and

target scenario structure and size by assuming a maximum number of AVs in the road network, duplicating the ring state representation by this number, and using 0-padding if the actual number of AVs was smaller. In contrast, in our transfer approach the policy does not directly control more AVs than it was trained for. Instead, it is deployed only in a specific key location in the scenario, and its state representation remains the same even though the scenario has a different geometry and a larger number of participants. In addition, the policy transferred from ring to merge did not surpass the performance of a policy trained from scratch, while using our approach the transferred policy did.

7.6 Summary, Limitations, and Future Work

In this chapter, we explore how autonomous agents can effectively coordinate with human proxies in open traffic scenarios. We demonstrate that the commonly used metric—average speed—is insufficient for evaluating traffic efficiency in open networks, as it can be artificially inflated by agents manipulating vehicle inflow. To address this limitation, we propose using network outflows as a more robust and manipulation-resistant metric. Using this metric as a reward function, our RL algorithm produces a driving policy that outperforms policies trained with prior reward functions, achieving higher throughput and average speed in small open network scenarios.

Building on this insight, we address the challenge of improving traffic flow in a large-scale network that contains a chain of merging intersections (specifically, Michigan's I-696 highway). We develop a modular transfer learning framework that transfers policies trained in smaller networks to local windows around junctions in the larger network. This modular approach yields superior outflows compared to both human-driven baselines and policies trained from scratch on the full network, while reducing training time by up to 80%.

Finally, we show that a distributed multi-agent RL policy—relying only on

local observations from onboard vehicle sensors—can improve traffic efficiency in a small open network. This distributed setting offers a more practical alternative to centralized methods that require global coordination among all autonomous vehicles. Empirical experiments in this chapter contribute to the **Dimension C: Collaborating with Human-Like Agents**.

Several limitations of this work point to promising directions that emerge from this work.

Human Proxies vs. Real Human Behavior. In this work, human driving behavior is modeled using proxy agents governed by the Intelligent Driver Model (IDM). While IDM offers a widely accepted approximation, it does not fully capture the variability, unpredictability, and strategic nuances of real human drivers. As a result, the learned policies may not generalize well to environments populated by actual human drivers. A promising future direction is to incorporate real-world human driving data into the simulation loop—either by learning behavior models from datasets such as NGSIM or Waymo Open Dataset, or by directly integrating human demonstrations. Doing so could improve the fidelity of the simulation and provide a more reliable benchmark for evaluating the robustness and adaptability of autonomous driving policies.

Sim-to-Real Transfer. All experiments in this study were conducted in simulated environments. While simulation provides a safe and controlled platform for developing and evaluating RL policies, deploying these policies in real-world traffic systems poses significant challenges. These include sensor noise, real-time inference constraints, and unexpected behaviors from human drivers. Future work could explore sim-to-real transfer methods, such as domain randomization, policy distillation, or fine-tuning with real-world data, to bridge the gap between simulation and deployment. Three years after the publication of this work, Lichtlé et al. (2024) conducted field experiments using the same simulator and demonstrated that optimizing human

proxies with real human driving data could improve real-world performance of the learned policies.

Broader Scenario Coverage. This work focuses on a small number of open-network traffic scenarios, including a scaled version of the I-696 highway. While these setups provide valuable insights, they do not capture the full diversity of real-world road structures and traffic patterns. Future research could expand the evaluation to include a wider variety of road topologies, such as urban intersections, arterial roads, roundabouts, and mixed-use streets and reward designs. Additionally, varying the density of human and autonomous vehicles, simulating different weather and visibility conditions, or introducing multi-modal traffic (e.g., pedestrians and bicycles) could further stress-test the robustness of the proposed methods.

This chapter presents the final part of the dissertation's technical contributions. Next, we will summarize the work, situate it within the broader context of related research, and discuss ongoing and future extensions of the contributions presented.

Part V

Related and Future Work

Chapter 8

Related Work

This chapter presents a comprehensive review of related work that underpins the contributions of this dissertation. While individual chapters provide contextual related work relevant to their specific contributions, this chapter situates the overall contributions within the broader research landscape and highlights their connections to subsequent developments in the field.

The remainder of this chapter reviews prior research in multi-agent policy generalization, inter-agent communication, and learning in mixed-autonomy settings. We focus on how these topics support our contributions in multi-agent learning and coordination, and we highlight representative works that shaped the field. Specifically, we discuss methods for **multi-agent policy generalization** (including diverse policy generation, population-based training, Empirical Game-Theoretic Analysis (EGTA), and Ad Hoc Teamwork (AHT)), advances in **multi-agent communication** (ranging from vehicle-to-vehicle signaling to natural language dialog), and approaches to **mixed-autonomy policy learning** (where AI agents learn to collaborate with or amidst human-controlled agents). We then survey two **application domains** relevant to this dissertation: **(a)** the cache timing attack problem (a security domain illustrating adversarial multi-agent interactions) and **(b)** large language model (LLM) based agents for autonomous driving (illustrating cutting-edge multi-agent coordination with AI-driven vehicles).

8.1 Multi-Agent Policy Generalization

Multi-agent policy generalization research addresses how an agent can learn strategies that are robust across diverse partners, opponents, and scenarios. A core challenge is that policies trained against a fixed set of co-players often overfit and fail against novel behaviors. To mitigate this issue, researchers have explored diverse policy generation and population-based training. In such approaches, instead of training a single set of policies in isolation, a population of agents co-evolves, exposing each agent to a wide variety of behaviors. This idea underpinned successes like AlphaStar's league training in StarCraft (Vinyals et al., 2019) and the Policy-Space Response Oracles (**PSRO**) framework (Lanctot et al., 2017). By maintaining a growing pool of policies and iteratively adding best responses, PSRO offers a principled way to explore strategy space and approximate game-theoretic solutions. Population-based training (**PBT**) yields a broader range of experiences by encouraging diversity in policies, which in turn improves generalization of the learned policies (Parker-Holder et al., 2020a). Overall, these methods produce agents that can handle unseen strategies more effectively by simulating the richness of the multi-agent environment during training. Our contributions **MACTA** and **L-BRDiv** are population-based training methods and can be formulated by the PSRO framework.

8.1.1 Empirical Game Theory Analysis

Empirical Game-Theoretic Analysis (**EGTA**) (Wellman, 2006) is a methodology that bridges multi-agent learning and classic game theory. When the game complexity scales, classic game theory analysis will not achieve a solution concept analytically (Wellman et al., 2025). Instead of relying on a fully specified analytic game, EGTA uses simulations of agents (which may be learned reinforcement learning policies or heuristics) to empirically estimate payoffs for strategy profiles, which yields an empirical game model capturing the strategic landscape. Researchers can then analyze this game model for equilibria or other solution concepts, guiding the

discovery of new strategies. Notably, automated strategy generation techniques like **PSRO** (Lanctot et al., 2017) have become popular EGTA approaches for iteratively expanding the strategy set. This dissertation instantiates variants of EGTA in analytically complex domains such as cache timing attacks and autonomous driving. Overall, EGTA provides a practical framework for analyzing multi-agent interactions through simulation-based game models and offers a promising approach for studying policy generalization in these settings.

8.1.2 Ad Hoc Teamwork

Ad Hoc Teamwork (**AHT**) refers to the problem of designing agents that can collaborate with previously unseen teammates without prior coordination or shared training. This challenge was formally introduced by Stone et al. (2010), who defined it as the problem of creating *"an autonomous agent that is able to efficiently and robustly collaborate with previously unknown teammates on tasks to which they are all individually capable of contributing."* Unlike self-play settings where agents co-train, an AHT agent must generalize to partners exhibiting arbitrary behaviors or strategies. Prior work in AHT typically assumes a fixed set of heuristic or reinforcement learning-based teammates and focuses on modeling and responding effectively to them (Mirsky et al., 2022; Barrett et al., 2016; Rahman et al., 2021; Zintgraf et al., 2021; Papoudakis et al., 2021; Gu et al., 2021). Recognizing the importance of teammate diversity in training, methods such as Fictitious Co-Play (Strouse et al., 2021) and Trajectory Diversity (Lupu et al., 2021) aim to generate a broad range of teammates by randomness. L-BRDiv was proposed to further this line of research by generating "meaningfully diverse" teammates — those that elicit distinct best responses — through optimizing best response diversity or adversarial diversity (Cui et al., 2023a; Charakorn et al., 2023; Rahman et al., 2023). Subsequent work has built upon this foundation to propose AHT evaluation metrics (Wang et al., 2024) and develop open-ended training frameworks for AHT (Wang et al., 2025).

8.2 Multi-Agent Communication

Communication is a key enabler for coordination in multi-agent systems. By sharing information, agents can synchronize plans, warn each other of dangers, or negotiate task allocation. Research on multi-agent communication can be divided into **domain-specific signaling** (using engineered messages in contexts like vehicle coordination) and **natural language communication** (where agents use or learn human-like language to communicate). Below, we discuss two sub-areas particularly relevant to this dissertation: Vehicle-to-Vehicle (V2V) communication in autonomous driving scenarios, and natural language communications in agent interactions.

8.2.1 Vehicle-to-Vehicle Communication

Network connectivity offers a great potential for improving the safety and reliability of self-driving cars. Vehicles can now share surrounding information via Vehicle-to-vehicle (**V2V**) and vehicle-to-infrastructure (**V2X**) channels using wireless technologies, such as Dedicated Short Range Communication (DSRC) (Kenney, 2011) and cellular-assisted V2X (C-V2X) (Gallo and Harri, 2013; Qualcomm, 2019). These V2V/V2X communication devices are increasingly deployed in current and upcoming vehicle models (Thompson, 2016; Plungis, 2018)). The academic community has built city-scale wireless research platforms (COSMOS (Yu et al., 2019)) and large connected vehicle testbeds (*e.g.,* MCity (Bezzina et al., 2023), DRIVE C2X (Stahlmann et al., 2011)), to explore the feasibility of cooperative vehicles and applications. Prior work (Qiu et al., 2018; Chen et al., 2019) proposed cooperative perception systems that broaden the vehicle's visual horizon by sharing raw visual information with other nearby vehicles. Such systems can be scaled up to dense traffic scenarios leveraging edge servers (Zhang et al., 2021) or in an ad-hoc fashion (Qiu et al., 2021). Wang et al. (2020); Li et al. (2021b); Xu et al. (2022) proposed multi-agent perception models to process sensor information and share compact representations within a local traffic network. In contrast, COOPERNAUT focus on cooperative driving of

networked vehicles with onboard LiDAR data under realistic networking bandwidth, advancing towards real-world V2V settings. Following COOPERNAUT, there has been advances in V2V modeling and emerging datasets for V2V cooperative perception (Xu et al., 2022).

8.2.2 Learning to Communicate in Natural Language

Recent advances in multi-agent systems have explored how agents can learn to communicate using natural language, enabling them to collaborate or compete in human-interpretable ways. Early work by Lewis et al. (2017) demonstrated that agents trained on human negotiation dialogues and fine-tuned via self-play reinforcement learning can develop strategic communication behaviors, such as feigning interest, while maintaining grammatical fluency. To avoid drifting into private, uninterpretable languages, the authors combined RL with supervised anchoring. Similarly, Lazaridou et al. (2020) proposed combining a pre-trained language model with a reinforcement-learned communication policy in referential games, showing that structural priors from natural language, when aligned with functional RL rewards, produce messages that are both effective and interpretable. This combination of structural priors and task rewards remains central to language emergence in agent interactions.

More recent efforts scale these ideas to complex social and strategic games. CICERO (Bakhtin et al., 2022a) achieved human-level performance in the strategy game Diplomacy by integrating a planning module trained with RL and a natural language model fine-tuned on human gameplay data. It grounded dialogue in strategic intent and filtered outputs for consistency, enabling fluent negotiation with human players. In parallel, social deduction games such as Werewolf have emerged as testbeds for emergent communication. Xu et al. (2023a) trained agents to select among LLM-generated messages using RL, allowing them to develop persuasive and deceptive dialogue strategies. Brandizzi et al. (2022) pushed further by training both speaking and listening agents from scratch via self-play RL, showing that meaningful communication can emerge even without human supervision. LLM+DEBRIEF

differs from prior work by enabling embodied agents to use natural language for coordinating **physical tasks** and introducing **centralized reflection**, where agents discuss and refine strategies post-execution to improve future performance. Such reflective learning has only become feasible with recent advances in large language models. Collectively, these works demonstrate that grounded reward structures, strategic alignment, and language priors are key to enabling agents to learn and use natural language effectively in multi-agent environments.

8.3 Multi-Agent Policy Learning with Mixed Autonomy

Mixed-autonomy scenarios, such as road traffic with both autonomous and human-driven vehicles, pose unique challenges: autonomous agents must learn policies that account for human behavior and even influence it to optimize system-level performance. A representative line of work comes from Vinitsky et al. (2018), who developed the Flow framework for mixed-autonomy traffic control. Their benchmarks, like the single AV in a ring road or multiple AVs at intersections, demonstrated that model-free deep reinforcement learning can eliminate stop-and-go waves and improve throughput beyond human baselines, as shown by Wu et al. (2018); Wu et al. (2017); Vinitsky et al. (2018) and Stern et al. (2018).

Later works (Lichtlé et al., 2024) conducted field experiments using the same simulator and demonstrated that optimizing human proxies with real human driving data could improve real-world performance of the learned policies. These findings generalize beyond driving: mixed autonomy is also observed in robotics and multi-agent games, where AI must operate safely and effectively amid unpredictable human actions. Techniques such as predictive human modeling and safe RL have emerged to address this uncertainty. Collectively, these works establish mixed autonomy as a crucial frontier for human-AI collaboration, offering foundational tools and paradigms that inform this dissertation's contributions.

8.4 Application Domains

Finally, we review related work in two specific application domains that intersect with our research: (1) the cache timing attack problem in cybersecurity, and (2) LLM-based agents for autonomous driving. The former showcases multi-agent dynamics in an adversarial context (attacker vs defender), while the latter represents a cutting-edge development in multi-agent cooperation with advanced AI models.

8.4.1 The Cache Timing Attack Problem

With increasingly sensitive data and tasks, security in modern computer systems has become one of the *14 Grand Challenges for Engineering* (National Academy of Engineering, 2007). A prominent example is cache-timing attacks (**CTA**), which exploit shared processor caches to leak sensitive information such as encryption keys (Yarom and Falkner, 2014; Liu et al., 2015), violate isolation boundaries (Kocher et al., 2019), escalate privileges (Lipp et al., 2018), and even bypass hardware security features in modern processors (Ravichandran et al., 2022). In these attacks, adversaries infer private information from victims through shared cache access patterns.

Over the years, both attack and defense strategies in CTA have been manually crafted by computer architecture experts. To automate detection, various statistical and machine learning approaches have been proposed. For instance, CC-Hunter (Chen and Venkataramani, 2014) uses autocorrelation to detect recurrent patterns in cache contention, while Cyclone (Harris et al., 2019) applies SVM classifiers based on cyclic interference patterns. ReplayConfusion (Yan et al., 2016) replays program traces with altered cache mappings to expose abnormal patterns, and PerSpectron (Mirbagher-Ajorpaz et al., 2020) trains a neural classifier on memory and latency logs. EVAX (Mirbagher-Ajorpaz et al., 2022) further enhances detection with GAN-based augmentation. However, these detectors often rely on fixed patterns and struggle to handle evolving attack strategies (Xiong and Szefer, 2020; Briongos et al., 2020; Saileshwar et al., 2021; Guo et al., 2022b;a).

Recognizing the arms race nature of security, researchers have adopted game-theoretic approaches to model attacker-defender interactions (Anwar et al., 2018; Elderman et al., 2017; Eghtesad et al., 2020). While insightful, these works typically simplify real-world dynamics and operate within limited strategy spaces. In contrast, real-world CTA involves high-dimensional state-action spaces, sparse rewards, and long horizons, making exhaustive analysis infeasible. MACTA, a reinforcement learning framework that enables automatic discovery of attacker strategies, was proposed in this dissertation to address this challenge.

8.4.2 LLM Agents for Autonomous Driving

LLM agents have shown potential to address various autonomous driving tasks. In particular, they are promising in tackling corner cases (Wen et al., 2023b) due to their reasoning ability and the common-sense knowledge embedded, yielding a more generalizable autonomous driving stack. Recent studies have explored various approaches to tailor state-of-the-art LLMs for driving (Wen et al., 2023a; Hu et al., 2024). However, a foundational challenge lies in grounding LLM agents in the real world—they need to perceive and understand the traffic scenarios. A straightforward approach is to obtain the observations from oracle perception models (Mao et al., 2023b) and convert them to textual descriptions (Mao et al., 2023a; Sha et al., 2023; Jin et al., 2023; Cui et al., 2023b). Some other studies tackled this challenge by introducing Visual Language Models (VLMs), which are adapted to driving domains through in-context instruction tuning (Ma et al., 2023) or fine-tuning (Wayve, 2023; Xu et al., 2023b; Ding et al., 2023; Yang et al., 2023). To enhance LLM agents' reasoning ability, prior works have investigated incorporating handcrafted guidance and examples in the prompts (Sha et al., 2023; Jin et al., 2023; Cui et al., 2023b), structuring the reasoning procedure (Mao et al., 2023b; Sima et al., 2023), and fine-tuning the models on driving datasets. Notably, fine-tuning LLMs and VLMs requires an extensive amount of driving data with language labels. Several approaches have attempted to adapt existing language-driving datasets for LLM fine-tuning (Ding

et al., 2023; Xu et al., 2023b; Ma et al., 2023) or augment large-scale multimodal driving datasets (Caesar et al., 2020; Sun et al., 2020; Mao et al., 2021) with language labels (Qian et al., 2023; Shao et al., 2023; Sima et al., 2023; Nie et al., 2023). In contrast, our work generates scalable driving data through agent self-play. Note that existing models were predominantly evaluated in an *open-loop* fashion. In contrast, similar to some prior work (Shao et al., 2023; Sha et al., 2023; Jin et al., 2023), we conduct closed-loop evaluation of the proposed method and baseline methods in CARLA (Dosovitskiy et al., 2017). More importantly, none of the existing work has explored optimizing LLM agents in a multi-agent setting with natural language vehicle-to-vehicle communication like LLM+Debrief. Concurrent efforts such as AgentsCoDriver (Hu et al., 2024) and LangCoop (Gao et al., 2025) explore emergent communication capabilities of LLMs but do not optimize communication strategies or policies explicitly.

Chapter 9

Future Work

While this dissertation advances several key aspects of multi-agent learning, many promising directions remain open for future exploration. The future work sections in each chapter have outlined concrete next steps specific to individual contributions. This chapter provides a broader perspective by highlighting follow-up work the author has undertaken that builds upon the techniques introduced in this dissertation, as well as prospective research directions inspired by the dissertation as a whole.

9.1 Towards Super-Human Pokémon AI

Competitive environments, such as Pokémon Video Game Championships (VGC) battles, offer a rich and challenging setting for studying multi-agent policy generalization. The domain is characterized by descriptive natural language rules and a vast configuration space of team compositions, stochastic environment dynamics, and partial observability—making it an ideal testbed for evaluating generalization and robustness in multi-agent decision-making.

The author worked with Angliss et al. (2025) to introduce a benchmark, **VGCBench**, for the Pokémon VGC domain and involved MACTA as one of the competitive baselines (referred to as FP in that work). The study demonstrated that initializing FP with a behavior-cloned policy trained on large-scale human gameplay

data significantly enhanced its robustness and performance in this complex setting. The resulting policy achieved a skill level comparable to players who qualify for the VGC championship.

Future research may extend this line of work by applying empirical game-theoretic analysis (EGTA) to better understand the strategic landscape of the game, or by integrating large language models to interpret natural language rule descriptions and Pokémon capabilities, potentially enabling generalization across arbitrary team configurations.

9.2 Open-Ended Training for Ad Hoc Teamwork

The author worked with Wang et al. (2025) to transform the Ad Hoc (AHT) Teamwork problem into a two-player zero-sum game between an incompatible-teammate generator and the Ad Hoc Teamwork agent that aims to approximate the best response to all possible teammates. To address this formulation, the study introduces a regret-based open-ended training algorithm, **ROTATE**, which jointly generates challenging teammates and trains robust AHT agents. ROTATE builds upon the core idea from L-BRDiv: namely, that a useful teammate policy is one whose best response differs significantly from BR policies that the current AHT agent could emulate, thus encouraging diversity and generalization in learning. Future work could extend the study of this framework to more complex settings involving N-agent interactions or general-sum games, broadening its applicability to richer and more realistic ad hoc teamwork scenarios.

9.3 General-Sum N-Agent L-BRDiv

Although the concept of Minimum Coverage Sets (MCS) can be extended to settings with more than two agents, L-BRDiv was originally designed for fully cooperative games and evaluated empirically only in two-player settings.

An ongoing line of work by the author aims to extend L-BRDiv to general-sum games — including fully adversarial, mixed-motive, and cooperative scenarios — and to environments with more than two agents. This extension involves two key modifications: (1) enabling support for general-sum games by activating only constraint Equation 6.11, and (2) handling multi-agent settings by treating all teammates as a joint policy. Empirical results suggest that this generalized version of L-BRDiv is effective in uncovering diverse policies even in complex general-sum environments, demonstrating the broader applicability of the approach.

9.4 Ad Hoc Teamwork Benchmark

A standardized benchmark for Ad Hoc Teamwork agents is critical yet currently lacking. Evaluation in AHT is often influenced by the choice of teammate policies, which are typically hand-crafted and not normalized across works, making it difficult to fairly compare methods. As the complexity of environments grows, designing diverse evaluation teammates through heuristics becomes infeasible, limiting our ability to assess an AHT agent's generalization. Moreover, the compositional nature of multi-agent setups hinders the development of a unified implementation and interaction API.

An ongoing effort initiated by the author is to build an AHT benchmark that addresses this gap by providing both a codebase for agent interaction and an evaluation framework. The evaluation set must consist of distinct teammates sampled uniformly or weighted through methods such as cross-play analysis or similarity-based selection. To probe collaborative weaknesses, we propose using diagnostic tools like NashConv or regret-based exploitability metrics that reveal how well an agent collaborates under worst-case teammate configurations without falling into self-sabotage—a scenario where an agent undermines team performance despite having cooperative potential. Existing benchmarks like MeltingPot (Agapiou et al., 2022) fall short in this regard: they often involve too many agents, contain mixed-motive or non-cooperative

elements, and lack the evaluative structure and diagnostics necessary to systematically advance AHT research.

9.5 Natural Language Communication and Collaboration for Embodied Agents

As embodied agents become more prevalent in real-world applications, ranging from autonomous driving to household robotics, natural language emerges as a crucial medium for collaboration. Unlike structured protocols, language enables flexible, human-compatible communication that can convey intentions, constraints, and situational updates. However, enabling embodied agents to effectively use natural language for coordination poses several challenges: grounding language in perception and action, maintaining coherence in multi-turn interactions, and handling ambiguity in real-time decision-making. These challenges are exacerbated in multi-agent settings, where agents must communicate with both humans and other agents, sometimes under zero-shot or ad hoc conditions.

While this dissertation takes a step toward this direction through the development of the LLM+Debrief framework, future work could significantly enhance its capabilities along two key dimensions: (1) enabling language-conditioned and compositional embodied motion, and (2) developing physical understanding and reasoning in natural language. Moreover, methods such as MACTA and L-BRDiv could contribute to generating diverse strategy profiles, thereby enriching the spectrum of behaviors in competitive or collaborative embodied agent interactions.

9.6 Multi-Agent Strategic Reasoning for Large Language Models

Large Language Models (LLMs) have recently demonstrated impressive capabilities in single-agent reasoning tasks, including arithmetic problem-solving, code generation, and chain-of-thought prompting. However, multi-agent environments

pose a fundamentally different set of challenges that require reasoning beyond an isolated perspective. In such settings, agents must account not only for the underlying physical-world dynamics but also for the diverse intentions, beliefs, and potential actions of other agents—whether they are collaborators, competitors, or fall somewhere in between. Strategic reasoning in these contexts involves anticipating others' moves, coordinating joint plans, negotiating trade-offs, and adapting dynamically to the observed behaviors of teammates or opponents.

Current LLMs are not yet capable of performing such multi-agent strategic reasoning at a level comparable to human experts, especially in complex, high-stakes domains such as Go or Pokémon VGC. These domains demand long-term planning, recursive opponent modeling, and nuanced understanding of team-level synergies—all of which challenge current capabilities. However, techniques developed in this dissertation can offer a path forward. For example, MACTA provides promising implications not only for improving robustness and generalization but also for studying safety alignment: by training LLMs against a curriculum of diverse training partners or attacker models, we may be able to better understand their vulnerabilities and encourage the emergence of safer, strategically aligned behaviors.

9.7 Multi-Agent Collaboration Safety

The communication-based learning methods proposed in this dissertation, such as LLM+Debrief and Cooper naut, operate under the assumption that all participating agents have cooperative intentions. While this assumption is valid in controlled environments, real-world multi-agent systems must be designed to handle a wider range of scenarios. In practice, agents may encounter adversarial behaviors, faulty communication, or misleading messages that compromise the safety and effectiveness of collaboration. Relying solely on honest communication can leave systems vulnerable to manipulation or coordination failure.

To ensure collaboration safety in open and potentially adversarial settings, fu-

ture work should investigate more robust communication mechanisms. For example, incorporating redundancy in information sharing, developing verification or consistency checks, and exploring cryptographic techniques to authenticate the source and integrity of messages. Agents should also be trained to identify and adapt to inconsistencies or signs of deception in communication. Extending the frameworks introduced in this dissertation to handle such challenges is a critical step toward building resilient and secure multi-agent systems that can operate safely in the real world.

Chapter 10

Conclusion

The growing integration of artificial intelligence into physical systems calls for agents that can operate robustly in dynamic, uncertain, and human-populated environments. This dissertation addresses a fundamental question: **How can autonomous agents be trained to effectively communicate and collaborate with diverse, previously unseen partners in open, complex environments?** The investigation was structured around three main dimensions of the key research question:

(A) Communication-Supporting Representations

(B) Multi-agent Policy Generalization

(C) Collaborate with Human-Like Agents.

This work introduces a series of technical contributions across multi-agent reinforcement learning, teammate policy generation, and mixed-autonomy traffic coordination, each addressing key challenges within these dimensions.

10.1 Contributions

This dissertation makes the following contributions to the multi-agent learning literature:

1(a) This dissertation introduces and evaluates **COOPERNAUT** (Chapter 3), an end-to-end driving framework that learns transmittable representations of local point-cloud observations through imitation learning. These learned representations support safer driving with significantly fewer collisions compared to disconnected agents in accident-prone scenarios, without compromising traffic efficiency. This contribution addresses **Dimension A**, focusing on what information to communicate under bandwidth constraints in cooperative autonomous driving.

1(b) This dissertation presents **LLM+DEBRIEF** (Chapter 4), a self-play learning framework that enables embodied large language model (LLM) agents to communicate and coordinate via natural language in driving scenarios. Agents are trained to articulate intentions, share observations, and negotiate driving plans. Experimental results demonstrate that agents can collaborate effectively using human-comprehensible language, a critical step toward human-AI teaming. This contribution addresses **Dimensions A and C** by enabling natural language-based joint decision-making among agents in dynamic environments.

2(a) This dissertation introduces **MACTA** (Chapter 5), a training framework that combines Proximal Policy Optimization (Schulman et al., 2017) with Fictitious Play (Brown, 1951) to develop robust policies in adversarial settings. The resulting policy demonstrates strong generalization to unseen and adaptive opponents, as shown in a simulated cache-timing attack scenario. This contribution addresses **Dimension B** by enhancing policy generalization through game-theoretic reinforcement learning.

2(b) This dissertation addresses the challenge of generalization in cooperative multi-agent settings by proposing that an agent can emulate a coverage set of the teammate policy space through exposure to a diverse range of training partners (Chapter 6). To this end, it introduces **L-BRDIV**, a teammate generation method that approximates a diverse subset of policies requiring distinct best responses. Results show that L-BRDIV produces qualitatively diverse teammates and enables ad

hoc agents to achieve state-of-the-art performance on standard ad hoc teamwork benchmarks. This contribution directly supports **Dimension B**, offering a practical approach to diversity-aware training in cooperative multi-agent environments.

3 This dissertation conducts an empirical study (Chapter 7) on the application of decentralized multi-agent reinforcement learning in **mixed-autonomy traffic coordination**. In this study, reinforcement learning agents trained without explicit communication successfully interact with both human drivers and AI agents. Findings reveal that a small number of RL-based autonomous vehicles can collaborate to influence human behavior and significantly improve traffic efficiency in open settings. This contribution addresses **Dimension C**, highlighting how decentralized agents can integrate and coordinate with human participants in real-world tasks.

10.2 Broader Impact

This dissertation contributes to the design of ad hoc and open-world AI systems that are not only intelligent but also socially compatible with diverse agents, including humans. Through empirical case studies in autonomous driving, mixed-autonomy traffic systems, and strategic reasoning, this work demonstrates that AI agents can be trained to:

1. Coordinate with previously unseen and diverse partners
2. Communicate effectively in natural language
3. Influence and collaborate with humans in dynamic, open environments

These findings offer concrete insights into the development of collaborative and generalizable multi-agent AI systems, advancing the goal of deploying adaptive and socially aligned agents in the real world.

Appendix

Appendix A

Additional Details on LLM+DEBRIEF

This appendix provides supplementary information for Chapter 4, including implementation details of the proposed method, environment design, example prompts, and representative learned cooperative strategies and knowledge.

A.1 Method

Algorithm 2 implements LLM+DEBRIEF, a multi-LLM-agent learning framework that leverages communication and centralized reflection using large language models (LLMs) to enhance coordination between agents in a simulated environment.

A.1.1 Implementation Details

We utilize `gpt-4o-mini` with a temperature of 0.2 for the agent policy, making decisions and collecting experiences every 0.5 seconds (10 simulation frames). The message dialogs received are maintained within a 2-second window according to the age of the message during each episode. The debriefing process is conducted after each episode for a total of 60 episodes, comprising a $N = 1$ round of discussion among agents followed by a final round of individual reflection to summarize and consolidate the discussion results. To enable stronger reasoning and summarization capabilities, `gpt-4o` is used for the debriefing sessions and reflection. The transition data are

Algorithm 2 Multi-Agent Centralized Debrief Reflection with Communication

Input: Multi-agent Simulation Environment `env`, LLM agents$\{\pi_{i\in\mathcal{I}}\}$, Debriefing round R.
Initialize: Knowledge $\{K_{i\in\mathcal{I}}\}$, Replay Buffer *ReplayBuffer*
for j=1, 2, 3... // Training epoch **do**
  $\{\text{obs}_i\}$ = `env.reset()`
  while $t < T$ // Time step **do**
    for $i = 1, ..., N$ //Per agent, but execute in parallel **do**
      // Get CoT reasoning for each agent based on observation and knowledge
      reasoning$_i$ ← `agents.reason`(obs$_i$, K$_i$)
      // Get decisions for each agent based on observation and knowledge
      message$_i$, control$_i$ ← `agents.act`(obs, K_i, reasoning$_i$)
    end for

    // Step the environment with actions
    $\{\text{next\_obs}_i\}$ ← `env.step`($\{\text{message}_i, \text{control}_i\}$)
    // Store experience to the replay buffer
    ReplayBuffer`.add`(obs, next_obs, reasonings, messages)
    // Message Dialog becomes part of the observation
    $\{\text{obs}_i\} \leftarrow \{\text{next\_obs}_i\} \cup \{\text{message}_i\}$
  end while

  // Get episode feedback from the environment
  feedback ← `env.evaluate()`

  // Lable all the transition data in hindsight
  `data_post_processing(`*`ReplayBuffer`*`)`
  // Debriefing and learning from feedback, update knowledge
  // Randomly decide debrief order
  for $r = 1, ..., R$ **do**
    if strategy=None **then**
      cooperation_stategy = `agent`$_r$`.propose()`
    else
      cooperation_stategy = `agent`$_r$`.revise()`
    end if
  end for
  //Summarize the dialogue and use it for future learning
  $\{K_i\}$ ← `agent.reflect()`($\{K_i\}$)
end for
last $\{\pi_{i,j}\}$ during the last iteration of self-play

sampled from the trajectory with a batch size of 4.

The **Batch Context Sampling** follows the following heuristic rule to assign probability mass on each transition data point and sample according to the normalized probability mass:

$$\begin{aligned}\text{Weight}_i = 1 &+ 2 \times \mathbb{1}\{\text{exists other agents}\} \\ &+ 5 \times \max(2 - \text{time to collision}, 0) \\ &+ 10 \times \mathbb{1}\{\text{actions contribute to collision}\} \\ &+ 0.1 \times \mathbb{1}\{\text{stagnation}\} \times \{\text{timestep}\} \\ &+ 2 \times \mathbb{1}\{\text{actions contribute to stagnation}\}\end{aligned} \tag{A.1}$$

where $\mathbb{1}$ represents the indicator function that takes the value 1 when the event happens, and 0 otherwise.

A.1.2 Inference Latencies

Table A.1 summarizes the average latencies and message sizes for each scenario in the communication setting, evaluated using `gpt-4o-mini` on a machine with Intel Gen 10 CPUs. The metrics include partial observable captioner latency (in seconds), reasoning latency (in seconds), decision latency (in seconds, excluding reasoning latency), and message size (in Mb). Data are aggregated over 10 episodes at each LLM decision step. Scenarios without communication exhibit slightly lower reasoning and decision latencies compared to those with communication within the same order of magnitude. We observe that the reasoning step is the main bottleneck of policy inference. However, the reasoning step is inevitable for LLM agents to make reasonable decisions without finetuning.

A.2 Environment

The TalkingVehiclesGym environment (Figure A.1) adopts the Gymnasium and PettingZoo APIs, assuming a parallel-acting setup to support efficient par-

Table A.1: Captioning, reasoning, decision latency, message size using `gpt-4o-mini` LLM Policy

Latencies \ Scenario	Overtake	Left Turn	Red Light	Overtake	Highway Merge	Highway Exit
Captioner Latency (s)	0.022	0.023	0.025	0.022	0.017	0.016
Reasoning Latency (s)	10.32	10.89	9.93	9.57	12.10	10.55
Decision Latency (s)	1.06	1.25	1.37	0.86	1.05	1.27
Message Size (Mb)	0.0016	0.0013	0.0014	0.0014	0.0005	0.0005

allel language model inference. We significantly modified the CARLA scenario runner to enable multi-agent communication and heterogeneous agent configurations within CARLA. TalkingVehiclesGym interfaces with both the CARLA server and client, establishing ***agents*** as a bridge between the simulator and ***learnable policies*** that rely on experience buffers for optimization. Language-communicating agents utilize an MQTT-based transceiver we implemented, enabling direct inter-agent communication without routing through the server.

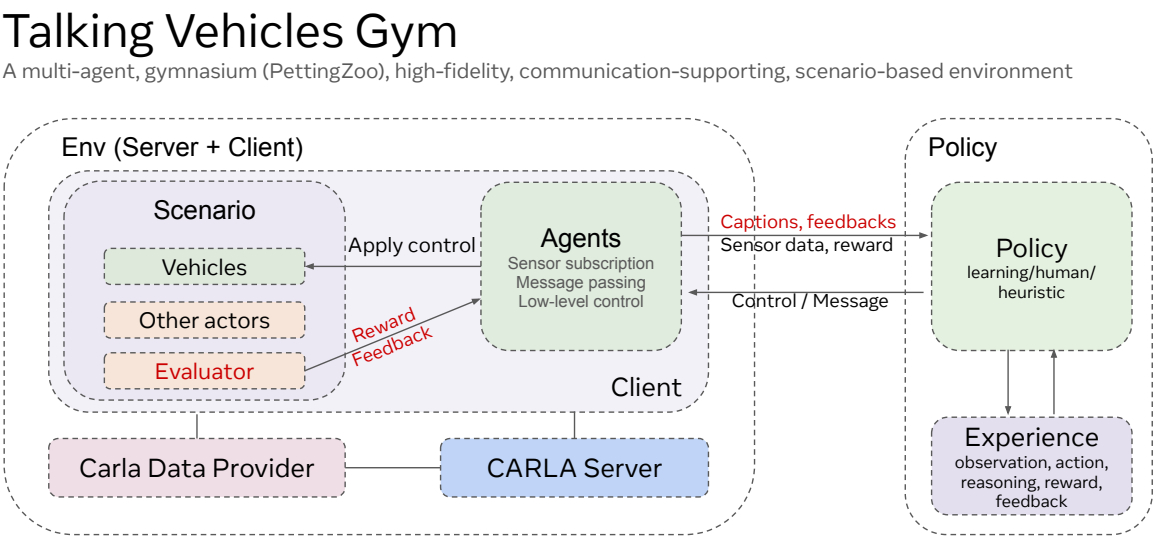


Figure A.1: TalkingVehiclesGym **simulation framework**. An agent is defined within the scenario and has a specific sensor registration and action space. A policy takes observations from an agent, computes actions, and learn from the experience replay buffer.

A.3 Example Agent Prompting Flow

Figure A.2 serves as a demonstration of the prompts; Please refer to our code for the actual prompts for the policy and the reflection process.

Figure A.2: Example agent prompting flow.

A.4 Example Learned Knowledge and Cooperative Strategies

The following knowledge and cooperation strategy examples are fully generated by LLMs without human modification.

A.4.1 Overtake (Perception)

Car1 (Overtaking Car)

knowledge: To efficiently and safely overtake a stationary vehicle in lane 1 by moving into lane -1, I will integrate active sensor use with continuous communication. The stationary vehicle will monitor lane -1 and provide real-time alerts on traffic status. Simultaneously, I will use my sensors to independently verify lane status. If both sources confirm lane -1 is clear, I will promptly proceed with the overtaking maneuver to avoid stagnation. If the stationary vehicle detects traffic, it will immediately advise me to hold my position. I will remain ready to accelerate smoothly once the lane is confirmed clear. This dual-verification approach, with proactive sensor use and real-time communication, will prevent collisions and ensure a timely and safe overtaking maneuver.

cooperative strategy: In our refined cooperative strategy, the stationary vehicle in lane 1 will continuously monitor lane -1 for any approaching traffic and provide real-time alerts. Meanwhile, the overtaking vehicle will actively use its sensors to independently verify the status of lane -1. If both the stationary vehicle's observations and the overtaking vehicle's sensors confirm that lane -1 is clear, the overtaking vehicle will promptly proceed with the maneuver to avoid stagnation. If the stationary vehicle detects any traffic in lane -1, it will immediately communicate this, advising the overtaking vehicle to hold its position. The overtaking vehicle will remain ready to accelerate smoothly once the lane is confirmed clear. This dual-verification approach, combining sensor data and real-time communication, ensures a timely and safe overtaking maneuver, preventing collisions and minimizing stagnation.

Truck (Stopped Truck in Lane 1)

knowledge: As the stationary truck in lane 1, my role is to continuously monitor lane -1 for any visible traffic and provide immediate alerts if I detect any vehicles, advising the overtaking vehicle to hold its position. The overtaking vehicle should actively use its sensors to independently verify lane status. If both my observations and the overtaking vehicle's sensors confirm that lane -1 is clear, the overtaking vehicle should promptly proceed with the overtaking maneuver to avoid stagnation. This dual-verification approach ensures a safe and efficient overtaking maneuver, preventing collisions and minimizing stagnation.

cooperative strategy: In our refined cooperative strategy, as the stationary truck in lane 1, I will continuously monitor lane -1 for any approaching traffic and provide immediate alerts if I detect any vehicles, advising the overtaking vehicle to hold its position. Meanwhile, the overtaking vehicle will actively use its sensors to independently verify the status of lane -1. If both my observations and the overtaking vehicle's sensors confirm that lane -1 is clear, the overtaking vehicle should promptly proceed with the overtaking maneuver to avoid stagnation. This dual-verification approach, combining sensor data and real-time alerts, ensures a timely and safe overtaking maneuver, preventing collisions and minimizing stagnation.

A.4.2 Red Light (Perception)

Car1 (Proceeding Forward with Green Lights)

knowledge: As a vehicle with a green light at an intersection, my primary task is to pass through safely and timely. I should communicate my intention to proceed to nearby cooperative vehicles and request confirmation that the intersection is clear. I must remain vigilant for warnings from vehicles with a clear view of the intersection about potential threats, such as vehicles approaching against a red light. If a warning is received, I should remain stationary until the monitoring vehicle confirms the intersection is clear. Continuous communication and responsiveness to

updates are key to preventing collisions and avoiding unnecessary delays, allowing me to proceed promptly when it is safe.

cooperative strategy: In our cooperative strategy, the vehicle with the best view of the intersection, typically positioned to turn left, will lead in monitoring and communicating updates about potential threats. This vehicle will provide real-time information about any approaching vehicles against a red light, including their distance and speed, and confirm when they have stopped or are no longer a threat. If a threat is detected, the vehicle with the green light, ready to proceed straight through the intersection, should remain stationary until the intersection is confirmed clear. The monitoring vehicle will send updates every few seconds to keep all vehicles informed. Once the threat is resolved, the monitoring vehicle will confirm the intersection is clear, allowing the vehicle with the green light to proceed promptly and safely. This plan ensures safety by preventing collisions and minimizes unnecessary delays by enabling vehicles to pass through the intersection efficiently when it is safe to do so.

Truck (Stopped at Intersection)

knowledge: As the stationary truck with a clear view of the intersection, my primary role is to monitor traffic and provide real-time updates to ensure the safe and efficient passage of vehicles. If I detect any vehicle approaching the intersection against a red light, I must immediately alert all nearby vehicles, providing specific information about the threat's distance and speed. The vehicle with the green light should remain stationary until I confirm that the threat has stopped or is no longer present. I will send updates every few seconds to keep all vehicles informed. Once the intersection is clear, I will confirm this, allowing the vehicle with the green light to proceed promptly and safely. This proactive communication ensures that all vehicles are aware of potential threats and can take appropriate actions to prevent collisions, while minimizing unnecessary delays by enabling vehicles to pass through the intersection efficiently when it is safe. My task is to facilitate this communication and

coordination to maintain a balance between safety and efficiency at the intersection.

cooperative strategy: In our cooperative strategy, the vehicle with the best view of the intersection, which is the stationary truck (myself), will lead in monitoring and communicating updates about potential threats. If a vehicle is detected approaching against a red light, I will immediately alert all vehicles, providing specific information about the threat's distance and speed. The vehicle with the green light, positioned to proceed straight through the intersection, should remain stationary until I confirm that the threat has stopped or is no longer present. I will send updates every few seconds to keep all vehicles informed. Once the intersection is clear, I will confirm this, allowing the vehicle with the green light to proceed promptly and safely. This plan ensures safety by preventing collisions and minimizes unnecessary delays by enabling vehicles to pass through the intersection efficiently when it is safe to do so.

A.4.3 Left Turn (Perception)

Car1 (Left-Turning)

knowledge: To execute a northbound left turn at the intersection while yielding to oncoming southbound traffic from lane -2, I should first slow down to assess the speed and distance of any oncoming vehicles. I will rely on the stationary vehicle at the intersection to continuously monitor traffic and provide real-time updates. This vehicle will send messages advising me when it is safe to proceed and instruct the oncoming vehicle to yield. I should only proceed with the turn once I receive confirmation that the oncoming vehicle has acknowledged the yield instruction. If the oncoming vehicle does not acknowledge or adjust its speed, I should stop and wait until it is safe to proceed. This approach ensures a safe and efficient left turn without causing collisions or unnecessary stagnation.

cooperative knowledge: In our revised cooperative strategy, the stationary vehicle at the intersection will continuously monitor the traffic flow and provide real-

time updates to both the left-turning vehicle (myself) and the oncoming vehicle. The stationary vehicle will send a message advising me to slow down and assess the situation, ensuring I only proceed with the left turn when I receive confirmation that the oncoming vehicle has acknowledged the yield instruction. Simultaneously, the stationary vehicle will instruct the oncoming vehicle to yield and adjust its speed to allow me to pass safely. If the oncoming vehicle does not acknowledge or adjust its speed, the stationary vehicle will alert me to stop and wait until it is safe to proceed. This plan ensures that all vehicles are aware of each other's intentions, allowing me to make the left turn safely and efficiently without causing collisions or unnecessary stagnation.

Truck (Stopped at Intersection)

knowledge: As a stationary vehicle with a clear view of the intersection, my primary task is to facilitate the safe and quick passage of the northbound left-turning vehicle by sharing critical traffic information. I must continuously monitor the intersection and assess the speed and distance of any oncoming vehicles. If an oncoming vehicle is approaching at a speed that could lead to a collision, I will send timely messages advising the left-turning vehicle to slow down and assess the situation, ensuring it only proceeds when the oncoming vehicle has acknowledged the yield instruction. I will also instruct the oncoming vehicle to yield and adjust its speed. If the oncoming vehicle does not acknowledge or adjust its speed, I will alert the left-turning vehicle to stop and wait until it is safe to proceed. This ensures clear communication and proactive monitoring, preventing collisions and avoiding unnecessary stagnation.

cooperative knowledge: In our revised cooperative strategy, as the stationary vehicle with a clear view of the intersection, I will continuously monitor the traffic flow and provide real-time updates to both the northbound left-turning vehicle and the oncoming vehicle. I will send a message to the left-turning vehicle advising it to slow down and assess the situation, ensuring it only proceeds when it receives

confirmation that the oncoming vehicle has acknowledged the yield instruction. Simultaneously, I will send a message to the oncoming vehicle instructing it to yield and adjust its speed to allow the left-turning vehicle to pass safely. If the oncoming vehicle does not acknowledge or adjust its speed, I will alert the left-turning vehicle to stop and wait until it is safe to proceed. This plan ensures that all vehicles are aware of each other's intentions, allowing the left-turning vehicle to pass the intersection safely and quickly without causing collisions or unnecessary stagnation.

A.4.4 Overtake (Negotiation)

Car1 (Overtaking Car)

knowledge: To successfully overtake the stopped broken truck in lane 1 by using lane -1, prioritize clear communication and adaptive speed management. Before attempting the maneuver, send a message to any oncoming vehicle in lane -1, indicating your intention to overtake and requesting a slight temporary speed reduction to create a safe gap. Wait for acknowledgment and ensure the gap is sufficient before proceeding. During the overtaking, minimize your time in lane -1 to reduce collision risk. Once safely back in lane 1, send a confirmation message to allow the oncoming vehicle to resume its speed. This approach ensures a coordinated and safe overtaking maneuver without causing unnecessary delays or collisions.

cooperative strategy: In our cooperative strategy, when I, as the vehicle intending to overtake a stationary truck in my lane, need to move into the opposite lane, I will first send a message to the oncoming vehicle in the opposite lane, indicating my intention to overtake and requesting a temporary speed reduction to create a safe gap. The oncoming vehicle should acknowledge this request and, if feasible, slightly slow down to create a safe gap, but avoid coming to a complete stop to prevent stagnation. Once the gap is sufficient, I will proceed with the overtaking maneuver and return to my original lane as quickly and safely as possible. After completing the maneuver, I will send a confirmation message, allowing the oncoming vehicle to

resume its target speed. This plan ensures that I minimize my time in the opposite lane while the other vehicle maintains its urgency, thus preventing collisions and avoiding stagnation. Effective communication and adaptive speed adjustments are key to ensuring both vehicles can complete their tasks safely and efficiently.

Car2 (Opposite Car)

knowledge: To effectively execute the task of going forward and keeping lane in lane -1 while in a hurry, prioritize maintaining speed and lane. If a vehicle in the opposite lane intends to overtake a stationary vehicle and needs to temporarily move into my lane, anticipate receiving a message indicating this intention. Upon receiving such a message, acknowledge it and, if feasible, slightly slow down to create a safe gap for the overtaking maneuver, but avoid coming to a complete stop to prevent stagnation. Ensure the gap is sufficient for safe passage. Once the overtaking vehicle has safely returned to its original lane, resume the target speed. This approach maintains urgency while facilitating safe and efficient traffic flow, preventing collisions and avoiding stagnation. Prioritize effective communication and adaptive speed adjustments.

cooperative strategy: In our cooperative strategy, when a vehicle in the opposite lane intends to overtake a stationary vehicle and temporarily move into my lane, it should first send a message indicating its intention and request a temporary speed adjustment. As the vehicle tasked with going forward and keeping lane, I should acknowledge this request and, if feasible, slightly slow down to create a safe gap, but avoid coming to a complete stop to prevent stagnation. The overtaking vehicle should proceed with the maneuver as quickly and safely as possible, minimizing its time in my lane. Once the overtaking vehicle has safely returned to its original lane, it should send a confirmation message, allowing me to resume my target speed. This plan ensures that the overtaking vehicle minimizes its time in the opposite lane while I maintain my urgency, thus preventing collisions and avoiding stagnation. Effective communication and adaptive speed adjustments are key to ensuring both vehicles can

complete their tasks safely and efficiently.

A.4.5 Highway Merge (Negotiation)

Car1 (Merging Vehicle)

knowledge: To effectively merge onto the highway when in a hurry, I should initiate communication by indicating my intention to merge and request the vehicle directly on the highway lane to my left to create a gap by slightly slowing down or temporarily changing lanes if feasible. I must observe the responses from vehicles already on the highway, particularly the one closest to the merge point, and adjust my speed to align with the newly created gap. I should not accelerate until the highway vehicle has stabilized its speed and distance. Continuous communication is crucial to ensure all vehicles are aware of each other's actions, allowing for coordinated speed and lane adjustments. If the gap is insufficient, I should be prepared to slow down significantly or stop to reassess the situation, ensuring a safe and efficient merge without causing collisions or stagnation.

cooperative strategy: In the cooperative strategy, as the merging vehicle, I will initiate communication by indicating my intention to merge onto the highway and requesting the vehicle directly to my left on the highway to create a gap by slightly slowing down or, if feasible, temporarily changing lanes. The highway vehicle should acknowledge this request and adjust its position accordingly, ensuring it maintains a safe distance. Meanwhile, I will adjust my speed to align with the newly created gap, ensuring I do not accelerate until the highway vehicle has stabilized its speed and distance. The vehicle behind the highway vehicle should maintain its speed or slightly slow down to prevent closing the gap prematurely. Continuous communication will be maintained, with updates on speed adjustments and intentions, to ensure all vehicles are aware of each other's actions. This approach will prevent collisions by ensuring a clear and sufficient gap for merging while avoiding stagnation by coordinating speed and lane adjustments effectively.

Car2 (Highway Vehicle)

knowledge: To execute the task of keeping on the original highway lane and going forward while in a hurry, prioritize maintaining a safe and efficient flow of traffic. When approaching a merge junction, be vigilant for merging vehicles and anticipate their need to enter the highway. If a merging vehicle communicates its intention, acknowledge the request and slightly reduce speed to create a sufficient gap, facilitating a safe merge. Ensure clear communication of actions to allow the merging vehicle to adjust its speed accordingly. Maintain your lane and continue moving forward, gradually accelerating to the desired speed once the merging vehicle has safely merged. Ensure the vehicle behind maintains its speed to prevent closing the gap prematurely. Continuous communication and dynamic speed adjustments are key to preventing collisions and avoiding stagnation, allowing the task to be fulfilled efficiently.

cooperative strategy: In the cooperative strategy, the merging vehicle should initiate communication by indicating its intention to merge onto the highway and requesting the highway vehicle directly to its left to create a gap by slightly slowing down. The highway vehicle, which is myself, should acknowledge this request and slightly reduce speed to create a safe merging space, while maintaining my lane and preparing to accelerate once the merge is complete. The merging vehicle should adjust its speed to align with the gap, ensuring it does not accelerate until I have stabilized my speed and distance. The vehicle behind me on the highway should maintain its speed or slightly slow down to prevent closing the gap prematurely. Continuous communication should be maintained, with updates on speed adjustments and intentions, to ensure all vehicles are aware of each other's actions. This approach will prevent collisions by ensuring a clear and sufficient gap for merging while avoiding stagnation by coordinating speed adjustments effectively.

A.4.6 Highway Exit (Negotiation)

Car1 (Exiting Highway)

knowledge: To exit the highway via the leftmost lane, initiate communication with the vehicle in the leftmost lane at least 100 meters before the exit junction, clearly indicating your intention to merge. Maintain a reasonable speed in the high-speed lane while seeking a safe gap to merge ahead of the traffic flow. If the vehicle in the leftmost lane is slightly ahead or side-by-side, it should decelerate slightly to create a gap. Adjust your speed dynamically to align with the gap being created, ensuring a smooth and safe transition into the left lane. Physically verify that the gap is sufficient for a safe merge before attempting the lane change. If the vehicle in the leftmost lane is stationary or unable to create a gap, communicate to confirm this status and seek alternative gaps or adjust your route if necessary. Prioritize visual confirmation over communication alone, and be prepared to adapt your strategy to the current traffic conditions to prevent collisions and avoid traffic stagnation.

cooperative knowledge: In our cooperative strategy, as the vehicle in the high-speed lane intending to exit, I will initiate communication with the vehicle in the leftmost lane at least 100 meters before the exit junction, clearly indicating my intention to merge. If the vehicle in the leftmost lane is slightly ahead or side-by-side, it should decelerate slightly to create a gap ahead, allowing me to merge smoothly without causing stagnation. I will maintain a speed that allows me to observe the gap being created and will only proceed with the lane change once I have a clear visual confirmation of a safe gap. If the vehicle in the leftmost lane is stationary or unable to create a gap due to traffic conditions, it should communicate this status immediately. In such cases, I will adjust my speed to maintain a safe distance and seek an alternative gap or prepare to slow down significantly if necessary. Both vehicles should actively communicate their speed adjustments and confirm when a safe gap is established, ensuring that the lane change is executed without collision or stagnation.

Car2 (Leader of the Left Flow Staying on the Highway)

knowledge: To effectively execute the task of staying in the leftmost lane and proceeding forward on the highway while prioritizing safety in a hurry, I should keep the following updated knowledge in mind: 1. **Proactive Communication:** Monitor for messages from adjacent vehicles intending to merge into my lane, ensuring communication is initiated at least 100 meters before exit junctions, and respond promptly to facilitate coordination. 2. **Adaptive Speed Management:** Decelerate slightly to create a sufficient gap when a merging vehicle is slightly ahead or side-by-side, allowing it to merge smoothly without causing stagnation. 3. **Enhanced Situational Awareness:** Continuously assess the speed and position of vehicles in adjacent lanes to anticipate merging actions and adjust my speed accordingly, ensuring a safe and efficient merge. 4. **Coordinated Communication:** Actively communicate and confirm speed adjustments with the merging vehicle to establish a safe gap, preventing collisions and maintaining traffic flow. 5. **Task Focus:** Maintain my position in the leftmost lane and proceed efficiently, ensuring cooperative actions support safety and fluid traffic flow, especially near exit junctions.

cooperative knowledge: In our cooperative strategy, when a vehicle in the adjacent lane intends to merge into the leftmost lane for a highway exit, it should initiate communication at least 100 meters before the exit junction, clearly indicating its intention to merge. As the vehicle currently in the leftmost lane, my responsibility is to promptly acknowledge this message and assess the traffic situation. If I am slightly ahead or side-by-side with the merging vehicle, I will decelerate slightly to create a gap ahead, allowing the merging vehicle to merge smoothly without causing stagnation. The merging vehicle should maintain a speed that allows it to observe the gap being created and only proceed with the lane change once it has a clear visual confirmation of a safe gap. If I am stationary or unable to create a gap due to traffic conditions, I will communicate this status immediately. In such cases, the merging vehicle should adjust its speed to maintain a safe distance and seek an alternative gap or prepare to slow down significantly if necessary. Both vehicles should actively communicate their speed adjustments and confirm when a safe gap is established,

ensuring that the lane change is executed without collision or stagnation."

A.4.7 Highway Merge (Negotiation) Silent Reflection

Car1 (Merging Vehicle) knowledge: Updated Knowledge for Merging onto the Highway:1. **Continuous Monitoring:** Always be aware of vehicles behind, directly in front, or to the side, as they pose immediate collision risks.2. **Early Gap Identification:** Identify potential merging gaps early and adjust speed in advance to align with these gaps, considering both vehicles ahead and behind.3. **Dynamic Speed Adjustment:** Accelerate only when a clear and safe gap is confirmed. Be ready to slow down if a vehicle behind is approaching quickly or if a vehicle ahead is close.4. **Maintain Safe Distance:** Prioritize keeping a safe distance from vehicles directly ahead and behind. If a vehicle is too close, adjust speed to increase the gap before merging.5. **Safety Over Speed:** Prioritize safe merging over speed. Avoid aggressive maneuvers that could lead to collisions, even if it means a slight delay.6. **Flexible Strategy:** Adapt strategies based on real-time traffic conditions. Reassess and choose a safer alternative if a planned action seems unsafe.7. **Immediate Threat Focus:** Pay special attention to vehicles approaching from behind in your intended merging path. Adjust your strategy to create a safe gap with these vehicles before merging.8. **Proximity Awareness:** If a vehicle is within a critical distance behind, prioritize adjusting speed to ensure a safe merging gap.9. **Cautious Acceleration:** When a vehicle is directly in front and moving slower, maintain speed or slow down to allow it to move further away before attempting to merge.By applying these strategies, you can merge onto the highway more effectively and safely, even when in a hurry.

Car2 (Highway Vehicle) knowledge: Updated Knowledge for Navigating Highway Merge Junctions 1. **Early Detection and Assessment**: Identify merging vehicles early, focusing on their speed and proximity. If they are close, prepare to adjust your speed promptly to facilitate safe merging.2. **Prioritize Safety Over Speed**: Always prioritize avoiding collisions over maintaining speed. Adjust your

speed to ensure safe distances from merging vehicles, even if it causes a slight delay.3. **Dynamic Speed Adjustment**: Be ready to slow down significantly if a merging vehicle is very close. Avoid abrupt speed increases that could reduce merging space and lead to collisions.4. **Continuous Monitoring**: Maintain awareness of the speed and position of nearby vehicles, especially those merging. Be vigilant of vehicles approaching from behind and to the side.5. **Proactive Space Creation**: Act promptly to create space for merging vehicles. Adjust your speed early to prevent conflicts and maintain a smooth flow.6. **Anticipate Merging Intentions**: If a vehicle is close and in a merging lane, anticipate its intention to merge and adjust your speed or position accordingly to prevent collisions.7. **Balance Urgency with Caution**: While in a hurry, balance the need for speed with safety. Ensure that any speed adjustments do not compromise the safety of merging vehicles.8. **Immediate Response to Close Proximity**: When a merging vehicle is extremely close, prioritize immediate action to slow down or create space, even if it means temporarily reducing speed significantly.9. **Evaluate Lane Change Options**: If safe and necessary, consider a temporary lane change to allow merging vehicles to enter your lane smoothly, while maintaining your original route.10. **Avoid Stagnation**: Ensure that actions do not cause stagnation or impede traffic flow. Adjust speed dynamically to maintain progress while ensuring safety.

A.4.8 Overtake (Perception) Communication Protocol by LLM+Debrief, seed 12, checkpoint-28

Car1 (Overtaking Car)

knowledge: To successfully overtake the stopped broken truck using lane -1, I will maintain a flexible speed and use my sensors to independently assess the speed and distance of oncoming traffic in lane -1. The stationary truck will continuously monitor lane -1 and provide real-time updates, sending a **"hold"** message if the gap is not safe. I will only proceed with the overtaking maneuver when both the stationary truck's updates and my sensor data confirm a safe gap, at which point the stationary

truck will send a **"go"** message. This ensures alignment and minimizes the risk of collision. If no safe gap is available, I will maintain my position and communicate my readiness to adapt speed as necessary, ensuring a coordinated and safe overtaking process.

cooperative strategy: In the revised cooperative strategy, the stationary truck in lane 1 will continuously monitor the traffic in lane -1 and provide real-time updates on the speed and distance of oncoming vehicles. If the gap is not safe for overtaking, the stationary truck will send a **"hold"** message to prevent premature maneuvers. As the overtaking vehicle, I will use my sensors to independently verify the traffic conditions in lane -1 and maintain a flexible speed, ready to adapt dynamically. I will only proceed with the overtaking maneuver when both the stationary truck's updates and my sensor data confirm a safe gap, at which point the stationary truck will send a **"go"** message. This dual verification approach ensures alignment in our actions, minimizing the risk of collision and avoiding stagnation by allowing me to adjust my speed based on real-time conditions. If no safe gap is available, I will maintain my position and communicate my readiness to adapt speed as necessary, ensuring a coordinated and safe overtaking process.

Truck (Stopped Truck in Lane 1)

knowledge: As the stationary truck in lane 1, my role is to assist the overtaking vehicle by continuously monitoring traffic in the opposite lane (-1) and providing real-time updates on the speed and distance of oncoming vehicles. If the gap is not safe for overtaking, I will send a **"hold"** message to prevent premature maneuvers. The overtaking vehicle should independently verify the traffic conditions using its sensors and maintain a flexible speed, ready to adapt dynamically. If both my updates and the overtaking vehicle's sensor data confirm a safe gap, I will send a **"go"** message, allowing the overtaking vehicle to proceed. This dual verification approach ensures alignment in decision-making, minimizes collision risk, and avoids stagnation by allowing the overtaking vehicle to adjust its speed based on real-time conditions.

cooperative strategy: In our revised cooperative strategy, as the stationary truck in lane 1, my role is to continuously monitor the traffic in the opposite lane (lane -1) and provide real-time updates on the speed and distance of oncoming vehicles. If the gap is not safe for overtaking, I will send a **"hold"** message to the bypassing vehicle to prevent premature maneuvers. The bypassing vehicle, meanwhile, should use its sensors to independently verify the traffic conditions in lane -1 and maintain a flexible speed, ready to adapt dynamically. If both my updates and the bypassing vehicle's sensor data confirm a safe gap, I will send a **"go"** message, and the bypassing vehicle should proceed with the overtaking maneuver. This dual verification approach ensures that both vehicles are aligned in their actions, minimizing the risk of collision by confirming safety from two perspectives and avoiding stagnation by allowing the bypassing vehicle to adjust its speed based on real-time conditions. If no safe gap is available, the bypassing vehicle should maintain its position and communicate readiness to adapt speed as necessary, ensuring a coordinated and safe overtaking process.

Appendix B

Additional Details on MACTA

This appendix provides supplementary details for Chapter 5. It includes the motivation for studying cache timing attacks, implementation specifics of the proposed method, environment configurations, and expanded analyses. These analyses encompass both quantitative and qualitative perspectives, evaluation on real hardware, and detailed comparisons with baseline methods. We also provide insights into the benign datasets used and implementation details of baseline attackers and detectors.

B.1 Why Study Cache Timing Attacks

CTA are stealthy but powerful. They do not violate any access control policies enforced by the operating system and low-level hardware and they are shown to pose serious security concerns in practice. For example, some implementations of security critical software such as encryption algorithms have a secret dependent access pattern, and an attacker can use CTA to obtain secret keys (Osvik et al., 2006; Liu et al., 2015). CTA also enables covert communication channels between two domains and breaks the existing security isolation mechanism, e.g., sandbox in javascript (Oren et al., 2015), isolation between processes (Kocher et al., 2019), and the system privilege levels (Lipp et al., 2018). CTA can also facilitate brute forcing hash values stealthily without triggering exceptions, which is shown to help break

the ARM pointer protection mechanisms (Ravichandran et al., 2022). One of the important defensive strategies is to detect unique characteristics of memory access patterns of attacker programs that are different from usual benign ones, as leveraged by the state-of-the-art cache-timing channel detectors (Yan et al., 2016; Chen and Venkataramani, 2014; Harris et al., 2019; Mirbagher-Ajorpaz et al., 2020). However, many new attacks (Briongos et al., 2020; Luo et al., 2023) avoid the characteristics that the detector uses and it is hard to adapt existing detectors to previously unseen attacks or access patterns.

B.2 Environment Configurations

Table B.1: Environment hyper-parameters.

Parameter Group	Parameter Name	Parameter Value
MA-AutoCAT	Max Episode Length	64 steps
MA-AutoCAT	Observation Window Size for the attacker and the detector	64 steps
MA-AutoCAT	Probability between Attack Scenario and Benign Scenario during Training	[50%, 50%]
MA-AutoCAT	Benign Program Logs (Train)	48 Million Steps
MA-AutoCAT	Benign Program Logs (Validation)	4 Million Steps
MA-AutoCAT	Benign Program Logs (Test)	40 Million Steps
MA-AutoCAT	Attacker Memory Address Range	8-15
MA-AutoCAT	Victim Memory Address Range	0-7
Cache Simulator	Cache Configuration	L1 Cache, 8 set 1 way
Cache Simulator	Replacement Policy	Least Recently Used (LRU)

Game Mechanism. In MA-AutoCAT, within a fixed-length episode, the attacker agent can guess the secret address of the victim as many times as possible and get a reward for every correct guess (*successful attack*). In the meantime, the detector agent can monitor the cache access history and interactions of two programs and decide whether to raise a flag/alarm to terminate the episode to prevent further information leakage.

Attacker's Reward Function. The attacker is punished by 0.01 for every time step, +10 if guess the victim's secret successfully, -10 if incorrectly. It will get a

one-time punishment of 20 if it reaches a timeout without any attack, a one-time punishment of -10 if identified by detector. The episode length to collect reward is affected by the detector.

Detector's Reward Function. The detector can raise a flag to terminate the episode. If the detector raises a flag in an attack scenario, then the detector receives a reward for remaining steps [max step - current step]; if the detector raises a flag in a benign scenario, then it receives a large penalty [5 $\times$ max step]. If the detector lets the episode going, for benign scenario there is no punishment; while the detector gets -10 every time the attacker attacks successfully.

Attacker's Action. For each time step, the attacker can choose an action $a^a \in \{a^a_X, a^a_v, a^a_{vr}, a^a_{gY}\}$, where a^a_X represents access address X, a^a_v represents letting the victim access a secret-related address, a^a_{vr} represents letting the victim access a random address and a^a_{gY} represents guessing the secret address to be Y.

Detector's Action. For each time step, the defender can choose $a^d \in \{a^d_{term}, a^d_{cont}\}$ where a_{term} means terminate the episode and a^d_{cont} means let the episode keep going.

Attacker's Observation. The attacker's observation space includes a history of attacker actions and memory access latency it receives from the cache simulator. For each time step, a new step observation $(s^a_{lat}, s^a_{vt}, s^a_{act}, s^a_{step})$ is appended to the observation window, where $s^a_{lat} \in \{s_{hit}, s_{miss}, s_{N.A.}\}$ represents the access latency, $s^a_{vt} \in \{s_t, s_{nt}\}$ represents whether to wait for the victim's action, s_{act} records the attacker's current action and s^a_{step} is the current time step.

Detector's Observation. The detector can observe a history window of the memory access actions of both programs. For each time step, the new observation is composed of $(s^d_{lat}, s^d_{program}, s^d_{set}, s^d_{step})$, where $s^d_{lat} \in \{s_{hit}, s_{miss}\}$ represents the access

latency (access latency of all programs are visible to the detector), $s^d_{program} \in \{s_a, s_b\}$ indicates the identity of the program, s^d_{set} represents the cache set being accessed at the current step, and s^d_{step} represents the current defender time step.

Benign Programs. The benign programs share the same action space with the victim and the attackers, and their domain id is randomized per episode. The actions of benign programs are sampled from pre-collected memory traces of the benign programs and we map the traces to the cache configuration (8 sets and 1 way) in this chapter. A detailed introduction to the benign datasets we use and the detectors' false alarm rate comparison is in Section B.3.

B.3 Benign Dataset

The memory traces of different benign programs are sampled from the Standard Performance Evaluation Corporation (SPEC) 2017 benchmark suite (Bucek et al., 2018). Specifically, we run each individual benchmark in the gem5 simulator (Binkert et al., 2011) and generate log files containing all memory accesses executed and their corresponding timestamps. To sample typical memory access patterns, we follow the standard practice in computer architecture studies to skip the first 2 million operations which represent the warm-up phase of programs, and sample the memory accesses in the middle of execution by skipping more steps. We collect memory access traces from the following 10 SPEC benchmarks, of which names are in the form of (program_id.program_name_speed/rate): 500.perlbench_r, 502.gcc_r, 505.mcf_r, 548.exchange2_r, 549.fotonik3d_r, 602.gcc_s, 607.cactuBSSN_s, 631.deepsjeng_s, 638.imagick_s, and 641.leela_s. Since our benign scenario consists of two benign programs, we mix two memory traces based on the timestamp in the simulation using either different benchmarks or identical benchmarks with different starting points. We prepare each benign trace to have 4 million memory accesses in total. Note that there can be combinatorically many different traces of two programs, we only select a

representative subset of them for the training and evaluation. Finally, we project the memory access traces onto the valid action space given the current cache configuration (8-set, 1-way, cache line size of 64 bytes).

We randomly select three programs (500.perlbench_r, 502.gcc_r, 505.mcf_r), and use different combinations (with replacement and with different skip steps) of them as training set. For example, the trace file name "500-2M_500-4M" means it contains two perlbench programs sharing the same cache. They start from different times (skipping the first 2 million memory accesses and skipping the first 4 million memory accesses, respectively) and continue until there are 4 million memory accesses by either program. We generate the validation dataset (549.fotonik3d_r, 607.cactuBSSN_s) and test dataset (548.exchange2_r, 631.deepsjeng_s, 638.imagick_s, and 641.leela_s) in the same way as the training set.

The ML models (MACTA, IBR-PPO, Cyclone) are trained on the same training set and tuned on the validation set. In the meantime, CC-Hunter's threshold is selected based on the validation set. Since the memory traces can exhibit different behaviors, we provide our per-trajectory false positive rate estimate in Figure B.1. All the machine learning models appear to perform better on the test set than the training set, which indicates a potential distribution shift between the training set and the test set. CC-Hunter's false positive rate is too high (ranging from $7.5 - 30\%$) to be plotted in the figure. In general, MACTA and Cyclone detectors have similar false positive rate on benign programs. The IBRPPO detector has lower false positive rates but it also has much lower detection rates. In addition, we closely inspect the benign traces that cause false positive. We find most of them are variations of the Prime+Probe attack on subset of cache sets. This is because even though the benign programs do not have malicious intentions, but they can still generate small pieces of memory access patterns that happen to be an attack pattern.

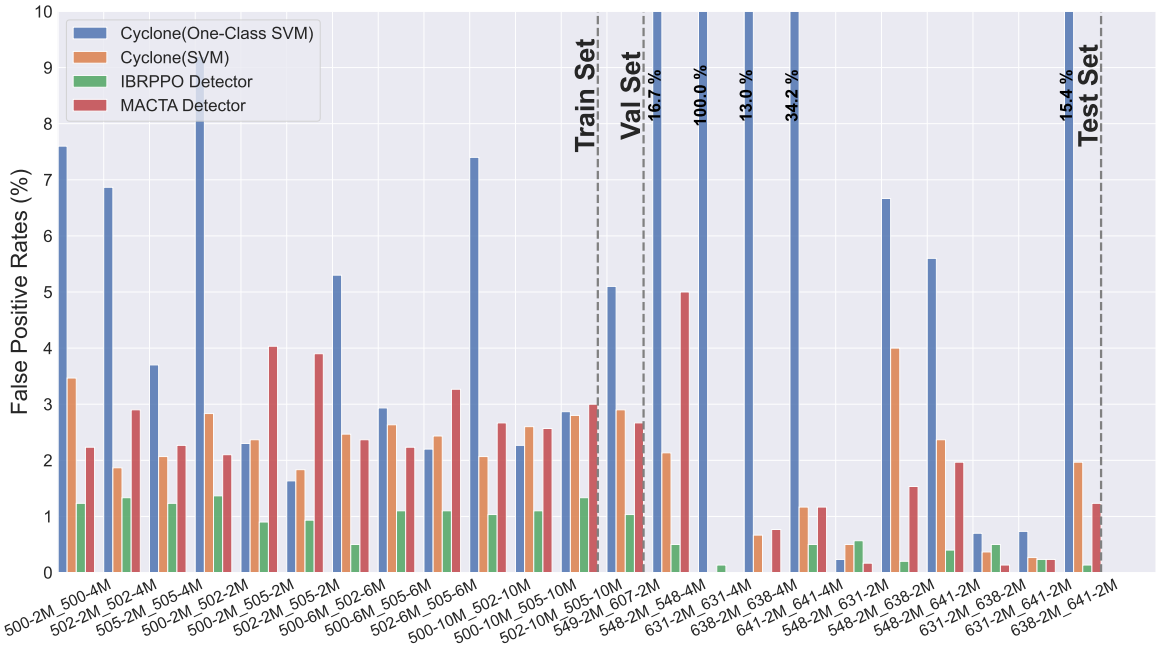


Figure B.1: False positive rates on different datasets. We report the per-dataset mean false positive rate for three models. CC-Hunter(threshold=0.45)'s false positive rates are too high to be included here.

B.4 Trajectory Analysis

Figure B.3 illustrates different attackers' attack sequences given a fixed secret bit. Here, we use victim secret=5 as an example. The top row shows a sampled pattern of benign programs. In that case, the two programs act independently and alter the access to the cache frequently. The Prime+Probe attacker causes cache contention by accessing the cache frequently, and only invokes the victim to access the cache when needed. Once contention with the victim in one cache set is observed (i.e., a cache miss after the same address is accessed by the attacker), the attacker will make a guess without more memory accesses. The IBR-PPO attacker takes a similar strategy as AutoCAT's, but it learns to insert some extra victim invocation steps to confuse the detector. The issue with the extra invocation strategy is that the victim only accesses its secret bit, which can be easily captured by the detector. The MACTA attacker, however, learns more advanced strategies. It learns to invoke

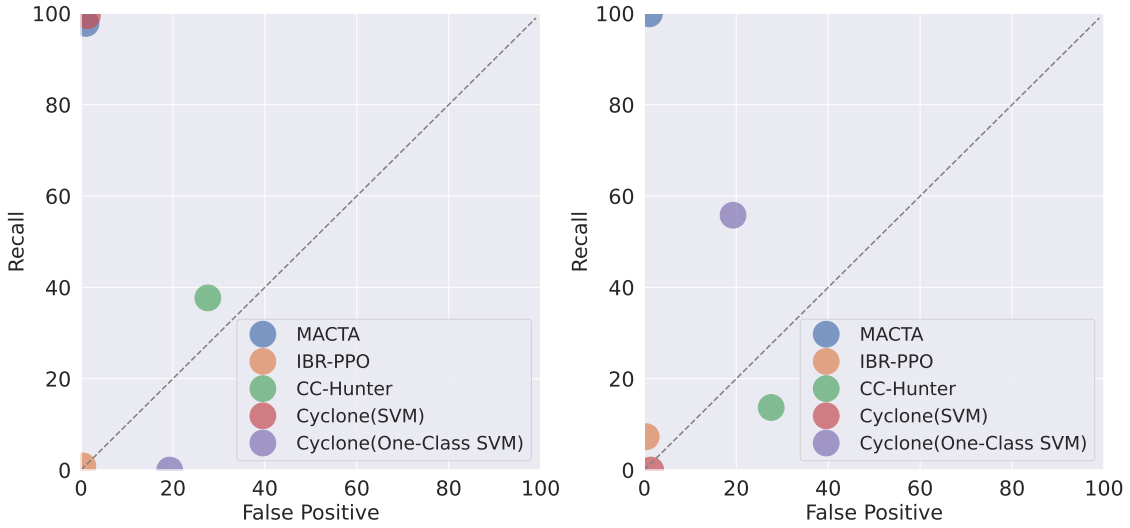


Figure B.2: The relative positions of all detectors' performance on the **ROC figure**. The recall is shown for the Prime+Probe attacks (**Left**) and the AutoCAT attacks (**Right**). The false positive rate is measured on the proposed test benign dataset. Here, Cyclone is trained on Prime+Probe attack sequences. But we did not provide the Prime+Probe attack sequences to MACTA detector explicitly.

random victim accesses to alter its behavior to be more similar to benign programs. Note that invoking random accesses from a victim can cause noise in the attacker's latency observation and make the steps needed for a successful attack longer. This means that to evade the MACTA detector, just inserting some extra victim invocation steps is far from enough. The attacker has to take a risk to invoke random accesses from a victim instead because the "easiest" policy space has been exhausted.

B.5 Real Hardware Analysis

Table B.2: Attack evaluation on commercial processors. We report the attack correct rates of MACTA attack sequences on three commercial Intel processors for 10,000 episodes. MACTA attackers achieve a $> 99.9\%$ correct rate in the simulator, and still $> 99\%$ on real hardware.

CPU	Cache Level	#Sets	#Ways	Attack Correct Rate ↑
i7-6700 (SkyLake)	L3	8	1(partitioned)	100.00%
i7-7700k (KabyLake)	L3	8	1(partitioned)	99.97%

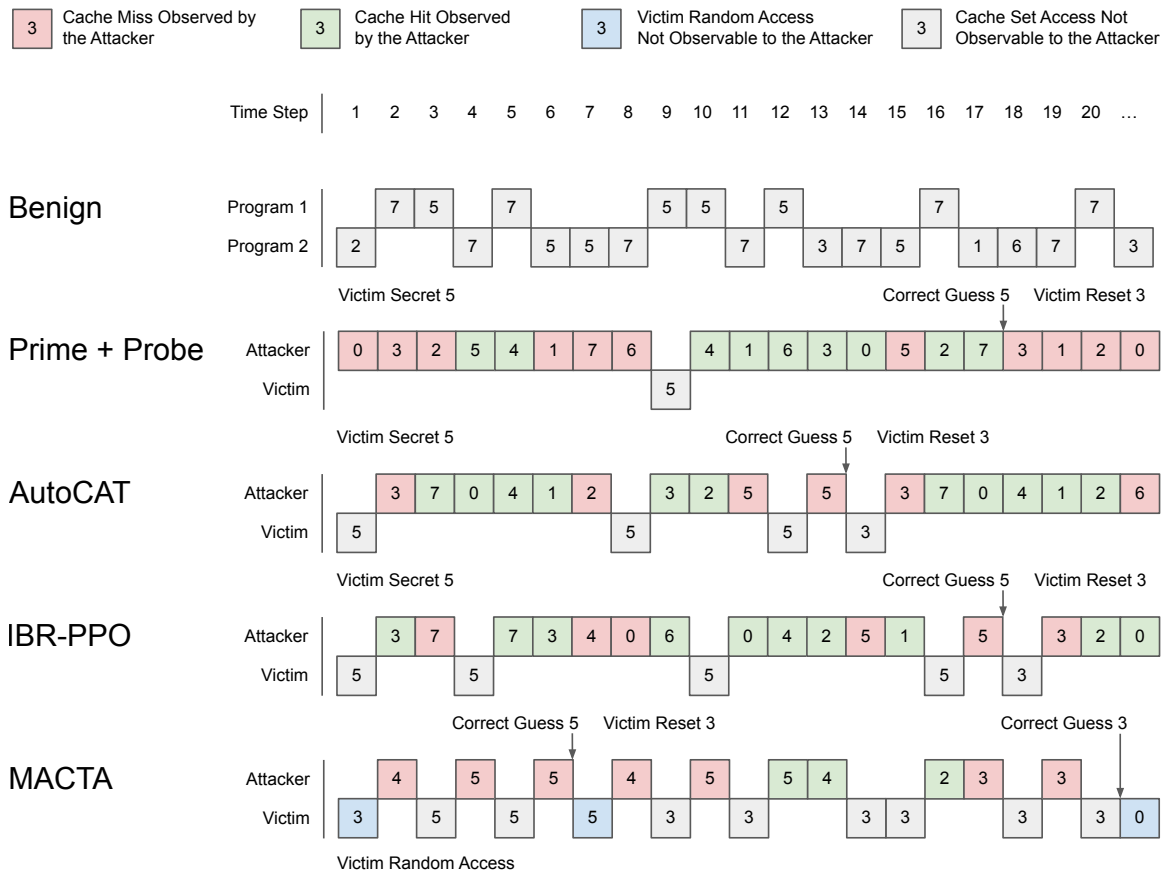


Figure B.3: Example trajectories of different attackers and benign agents in a 8-set 1-way L1 cache. The number indicates the cache set being accessed. Red and green boxes show the observation by the attacker. The latency of other programs (i.e., victim or benign) cannot be observed by the attacker, but they can be observed by the detectors. The program IDs are randomized during training, and the attacker can be any of the two programs in the system. The cache is initialized with random states.

We evaluated the effectiveness of the attack sequences produced by MACTA attackers on real hardware by running them on two commercial processors through CacheQuery (Vila et al., 2020). The attack sequences are generated from a MACTA attacker that is trained using a simulated environment for an 8-set 1-way cache configuration. Then, the attack sequences were performed on real hardware to obtain the attack correct rate. Table B.2 demonstrates the attacker policy from the simulator can be transferred to real hardware with negligible discrepancy.

B.6 Model Architectures

In Section 5.4.4, we compared three different neural network architectures: Transformers (Vaswani et al., 2017), LSTM (Hochreiter and Schmidhuber, 1997), and MLP. All of the methods are controlled at the same scale of parameters and trained with PPO (Schulman et al., 2017) without dual-clip (Ye et al., 2020) on two different machines. The details about the model architectures are listed below.

Transformer. In our experiments, all the policies use an 8-head 1-encoder-layer Transformer with $d_{model} = 128$ and $d_{feedforward} = 2048$. For the model architecture study, we study the changes in the number of heads in the multi-head attention mechanism, and the number of Transformer Encoder layers. Similar to Luo et al. (2023), we apply an average-pooling to reduce the step dimension.

LSTM. We employed a 1-layer LSTM with hidden dimension of 256. The input to the LSTM is the embedding of the pre-padding history of the observation. We concatenate the hidden and cell states of the last step and use it as the sequence embedding.

MLP. The MLP model we used directly feeds the input embeddings into 4 pre-activation residual blocks (He et al., 2016b;a) with hidden dimension 128. Each residual block is composed of 2 ReLU-Linear layers with a residual connection.

We obverse that the MLP model fails even when tested on multiple different machines while Transformer and LSTM models both work. The learning speed of Transformer and LSTM models can be different. Our hypothesis is that the CPU/GPU configurations of different machines may affect the policy lag (Petrenko et al., 2020) of Asynchronous PPO training, which may lead to different learning speed for different models.

B.7 Algorithm and Training Hyper-parameters

The MACTA algorithm is explained in Algorithm 3, and detailed training hyperparameters can be found in Table B.3. In MACTA, the policy pool is created by sampling a new model to do batch actions per step. It is an infrastructure implementation to produce faster sampling speed, so it is not strictly a per-step sampling, but it is per-step sampling considering a large amount of data. Additionally, we are exploring the per-trajectory policy sampling because it has nice theoretical properties. Nevertheless, it needs further infrastructure support.

Algorithm 3 MACTA

Initialize Number of Fictitious Play Iterations I, Attacker Policy Pool $\mathcal{P}_A$, Detector Policy Pool $\mathcal{P}_D$, Number of Epoch per Fictitious Play Iteration E, Add a policy to Pool per N epochs. `PPO` the Proximal Policy Optimization. $\mathbb{U}$ the uniform random sampling of policies per step. $i \leftarrow 0, j \leftarrow 0, k \leftarrow 0$
while $i < I$ **do**
    $j \leftarrow 0$
    while $j < E$ **do**
        π_{Aj}=`PPO`$(\mathbb{U}(\mathcal{P}_D))$ ▷ Train attacker policy against the pool of the detectors
        if j mod $N - 1$==0 **then**
            $\mathcal{P}_A \leftarrow \mathcal{P}_A \cup \pi_{Aj}$ ▷ Add an attacker checkpoint to the attacker pool
        end if
        $j \leftarrow j + 1$
    end while
    $j \leftarrow 0$
    while $j < E$ **do**
        π_{Dj}=`PPO`$(\mathbb{U}(\mathcal{P}_A))$ ▷ Train detector policy against the pool of the attackers
        if j mod $N - 1$==0 **then**
            $\mathcal{P}_D \leftarrow \mathcal{P}_D \cup \pi_{Dj}$ ▷ Add a detector checkpoint to the detector pool
        end if
        $j \leftarrow j + 1$
    end while
end while
return π_A, π_D ▷ return the last policy of attacker and detector

Table B.3: Training hyper-parameters for MACTA.

Parameter Group	Parameter Name	Parameter Value
Fictitious Play	Fictitious Iterations	18 iterations
Fictitious Play	Epochs per Iteration per Agent	50 epochs
Fictitious Play	Training Steps per Epoch	3000 steps
Fictitious Play	Frequency of Adding one Policy to Pool	10 epochs
Computing Resource	Number of Sampling Actors	72 Actors
Computing Resource	Sampling Instance per Worker	3 Actors / Worker
Computing Resource	Remote Model Push Frequency	10 steps
Computing Resource	GPU Information	4 Nvidia Tesla V100 16G / 32G
Computing Resource	CPU Information	80 Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz
Proximal Policy Optimization	Replay Buffer Size	262144
Proximal Policy Optimization	Training Batch Size	512
Proximal Policy Optimization	Learning Rate	1e-4
Proximal Policy Optimization	Entropy Coefficient	0.03
Proximal Policy Optimization	Discount Factor γ	0.99
Proximal Policy Optimization	Max Gradient Norm	1.0
Proximal Policy Optimization	GAE λ	0.95
Proximal Policy Optimization	Policy Ratio Clipping ϵ	0.2
Proximal Policy Optimization	Value Clipping ϵ	0.2
Proximal Policy Optimization	Value Loss Coefficient	0.5
Proximal Policy Optimization	Dual-Clip Threshold	3.0
Model Architecture	Number of Transformer Encoder Layers	1
Model Architecture	Transformer d_model	128
Model Architecture	Transformer nhead	8
Model Architecture	Transformer dim_feedforward	2048
Model Architecture	Transformer dropout	0.0

B.8 Heuristic Cache Timing Attacks and Detectors

B.8.1 Heuristic Attacker Algorithms

Prime+Probe (Algorithm 4) Osvik et al. (2006). First, in the prime phase, the attacker fills the cache set with its address value (lines 3) in a randomized way, then waits for the victim to access the cache set. Next, the victim accesses one of the cache sets, then replaces the loaded address value with its address (lines 5). Lastly, in the probe phase, the attacker accesses the cache sets again in a random permutation order, then measures the cache latency to load each set of the primed address value (lines 7 to 8). In a cache set accessed by the victim, the attacker observes increased

latency (cache miss) and makes a guess.

Algorithm 4 Prime+Probe Attack

```
step ← step + 1
if step < len(attacker_address_range) then
    action = prime_address(step, cache_size) ▷ attacker fills cache by attacker's address
else if step = len(attacker_address_range) then
    action = trigger_victim(step)        ▷ victim accesses a cache and fills its own address
else
    action = probe_address(step, cache_size)        ▷ attacker access caches again
    measure latency(action)
end if
if latency = 1 then                                  ▷ attacker observes for any cache miss
    action = guess(action, cache_size)     ▷ attacker makes a guess on a victim's secret address
end if
Return action
```

B.8.2 Detector Algorithms

CC-Hunter Chen and Venkataramani (2014). Cache timing channels rely on the latency of events to perform timing modulation. To send information, two processes (*i.e.*, the trojan and the spy) generate a sufficient number of alternating conflict events (cache misses) to allow the adversary to decode the transmitted bit based on the average memory access times (hit/miss). Those behaviors show periodic, oscillating patterns of conflicts between the two processes. Therefore, autocorrelation is used to identify those patterns. Autocorrelation is the correlation coefficient of the signal with a time-lagged version of itself, along with measuring the event train X, as a variable at a time instance of t. Two cases of conflict miss, *i.e.*, either the victim eviting the attacker's cache line or the attacker evicting the victim's cache line, are considered for the event trains. For example, we can check the autocorrelation C_p at

a time lag p, which is expressed as:

$$C_p = \frac{\sum_{i=0}^{n-p} \left[\left(X_i - \bar{X}\right)\left(X_{i+p} - \bar{X}\right)\right]}{\sum_{i=0}^{n} \left(X_i - \bar{X}\right)^2}$$

If there exists a time lag p which $1 \leq p \leq P$, where P is a predefined parameter such that makes C_p larger than a threshold value, then it is assumed as an attack.

We tune the threshold to be $p = 0.45$ on our validation set, and this threshold yields a 7.5% false alarm rate and a 38% detection rate on Prime+Probe. However, this threshold fails to generalize to the test set, giving us a 27% average false positive rate, as reported in this chapter. The main issue with applying CC-Hunter to our environment is that our episode length is short, and the cache is initialized with random loads after resetting. As a result, even attackers' latency histories can have low auto-correlations.

Cyclone Harris et al. (2019). The concept of *cyclic* interference is commonly found in all known cache contention side-channel attacks and has been used for detecting those attack patterns. Interference occurs from the attacker to the victim process or vice versa, considered directional, and affects the behavior of microarchitecture in a disruptive manner. The cyclic interference can be noted as (a $\rightsquigarrow$ b $\rightsquigarrow$ a), where interference (a $\rightsquigarrow$ b) is followed by (b $\rightsquigarrow$ a). However, interference including a third party between attacker and victim, like (a $\rightsquigarrow$ b $\rightsquigarrow$ c), is not considered as cyclic interference. To distinguish attack and benign patterns, Cyclone uses a one-class support vector machine (One-Class SVM). In our experiments, however, we found that the one-class SVM is not effective in detecting our Prime+Probe implementation and has a high false positive rate under our testing configuration. Consequently, we also experimented with an SVM trained to classify Prime+Probe attack and benign traces, as the Cyclone paper suggests the feature can be used for other classifiers.

To extract the programs' cyclic features, we use 8 buckets and an observation window of 17 steps (which is the same as Prime+Probe's frequency). We train both

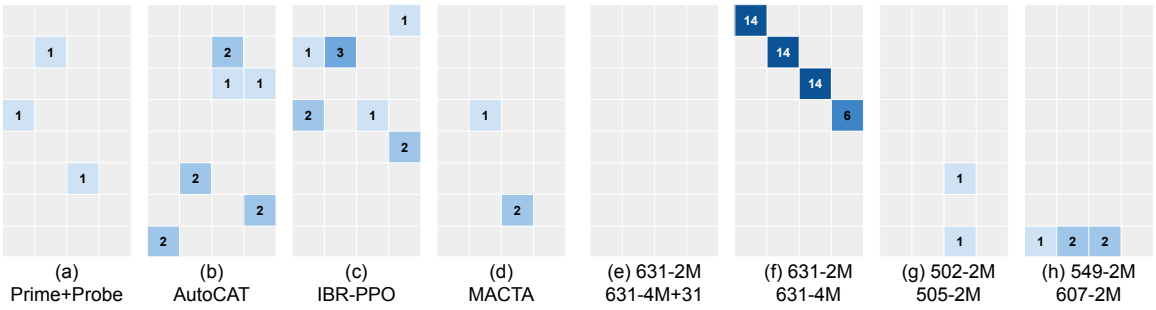


Figure B.4: Example learned Cyclone features for various scenarios: (a, b, c, d) represent typical features when attackers interact with a victim; (e, f, g, h) depict typical features resulting from interactions between benign programs. The feature value in the grey areas is 0, and the intensity of the blue color indicates the frequency of cyclic inference, with darker shades representing more frequent occurrences. (e) illustrates the interaction between program 631.deepsjeng_s starting at 2 million (M) steps and the same program at 4M+31 steps. (f) demonstrates a trace of 631.deepsjeng_s self-mix from the test set, (g) shows a trace from the training set, and (h) presents a sample trace from the validation set. Typical test set features are similar to those of the train and validation set.

Cyclone (One-Class SVM) and Cyclone (SVM) models with Gaussian kernels using 24,000 normalized train samples until convergence. For Cyclone (One-Class SVM), we choose the upper bound on the fraction of training errors and a lower bound of the fraction of support vectors to be 1%, with all training data generated from the benign (DBB) scenario. For Cyclone (SVM), the regularization coefficient C is set to 100, with 50% of the train data from benign scenarios, and 50% of them from malicious (DAV) scenario.

We observed that Cyclone (One-Class SVM) fails to detect Prime+Probe attacks and exhibits high false positive rates on datasets where two programs execute identical tasks. This inability to detect Prime+Probe is primarily due to the low bandwidth of the implemented attack. Regarding the elevated false positive rates, several factors contribute to this issue. One reason is that the cache configuration and system setting we employ could potentially bring high cyclic behaviors even between two benign programs. In addition, we discovered that when two benign programs with periodic memory access patterns are mixed, the difference in the starting time of the two programs could significantly impact the cyclic behaviors between the programs.

Notably, for 631-2M_631-4M in the test set, Cyclone (One-Class SVM) exhibits a 100% false alarm rate (Figure B.4(f)); however, if we shift the start time of the program, the false alarm rate decreases to 0% (Figure B.4(e)). Cyclone (One-Class SVM) also demonstrates fluctuations with the program start time on the 641 self-mix but is generally more stable on other program combinations. In contrast, we found that the MACTA detector and Cyclone (SVM) are not sensitive to the program start-time offsets or combinations of programs, consistently showing low false alarm rates across various test sets and offsets. For fair comparisons, we report the statistics of all methods on the same test set without counterfactual dataset selection, but we encourage the readers to be mindful of the potential bias.

Appendix C

Additional Details on L-BRDiv

This appendix provides supplementary information for Chapter 6, including implementation details, experimental setups, and extended analyses that support the evaluation of the L-BRDiv framework. We begin by describing the construction of heuristic-based teammate policies used for evaluating ad hoc team (AHT) performance across different environments. We then present a mathematical analysis illustrating failure cases of existing diversity-driven methods (BRDiv and LIPO) in discovering the full maximal covering set (MCS) of policies. This analysis highlights how optimizing their diversity metrics can lead to suboptimal coverage and poor generalization in AHT settings.

We further explore the behavior of L-BRDiv's Lagrange multipliers, showing how they evolve during training to enforce the optimization constraints. In addition, we provide detailed hyperparameter settings for both teammate generation and AHT training, ensuring reproducibility and fair comparisons with baseline methods. These details collectively support the robustness and effectiveness of L-BRDiv in generating diverse and strategically relevant teammate policies for ad hoc teamwork.

C.1 Teammate Policies for AHT Evaluation

We outline the different types of teammate policies in the set of teammates we use for AHT evaluation, Π^{eval}. For each environment, teammate policies in Π^{eval} are based on simple heuristics. Details of heuristics used for each environment are outlined in the following sections.

C.1.1 Repeated Matrix Game

Since the Repeated Matrix Game is a simple environment without any states, we only implemented six simple heuristics which details are provided below:

- **H1**. Agents that follow this heuristic will always choose the first action.
- **H2**. This heuristic will get an agent to always choose the second action.
- **H3**. Agents using this heuristic will always choose the third action.
- **H4**. Unlike H1-H3, this heuristic gives agents a policy that chooses the first, second, and third action with probabilities of 0.7, 0.15, and 0.15 respectively.
- **H5**. This is a policy that chooses the first, second, and third action with probabilities of 0.15, 0.7, and 0.15 respectively.
- **H6**. Agents following this heuristic will choose the third action 70% of the time. Meanwhile, it is also equally likely to choose between the first and second actions.

C.1.2 Cooperative Reaching and Weighted Cooperative Reaching

For the Cooperative Reaching and Weighted Cooperative Reaching environment, we implement 15 types of teammate heuristics whose behaviour are detailed below:

- **H1.** H1 controls an agent to always move to the closest corner cell from its initial location.
- **H2.** This heuristic moves an agent towards the furthest corner cell from its the agent's initial location at the beginning of the episode.
- **H3.** H3 controls an agent to move towards the closest corner cell between corner cells A and B.
- **H4.** Based on the agent's initial location at the beginning of an episode, H4 will move agents towards the furthest cell between cells A and B.
- **H5.** H5 moves an agent towards the closest cell between cells C and D.
- **H6.** Depending on the agent's position at the beginning of an episode, H6 controls the agent to move towards the furthest cell between cells C and D.
- **H7.** At the beginning of each interaction, H7 randomly picks a destination cell between A, B, C, and D with equal probability. For the remainder of each episode, the agent will be controlled to move towards the destination cell.
- **H8-H11**. H8-H11 move agents towards corner cells A-D respectively.
- **H12**. H12 moves an agent towards corner cell A with a 55% chance. Meanwhile, the other corner cells are equally likely to be chosen as destination cells.
- **H13**. H13 moves an agent towards corner cells A, B, C, and D with a 15%, 55%, 15%, and 15% chance respectively.
- **H14**. H14 moves an agent towards corner cells A, B, C, and D with a 15%, 15%, 55%, and 15% chance respectively.
- **H15**. H15 moves an agent towards corner cell D 55% of the time. Meanwhile, the remaining corner cells are equally likely to be chosen as destination cells.

C.1.3 Level-based Foraging

Experiments in the Level-based Foraging environment evaluate AHT agents against Π^{eval} consisting of 8 heuristic types defined below:

- **H1.** Agents under H1 will move towards the closest item from its current location and collect it. This process is repeated until no item is left.
- **H2.** At the beginning of an episode, agents under heuristic H2 will move towards the furthest object from its location and collect it. Every time its targeted item is collected, the agent will then move to collect the remaining item whose location is furthest from the agent's current location. This process is repeated until no item remains.
- **H3-H8.** H3-H8 each corresponds to a heuristic that collects items following one of the six possible permutations of collecting the three items available in the environment.

C.2 Analyzing Baseline Failure in Repeated Matrix Game & Weighted Cooperative Reaching

In this section, we mathematically demonstrate that no constant and uniform $\alpha > 0$ can make BRDiv or LIPO identify all policies in MCS(E) for the Repeated Matrix Game and Weighted Cooperative Reaching environment. Section C.2.1 details our argument regarding the baselines' failure in the repeated matrix game. Meanwhile, the same argument for the Weighted Cooperative Reaching environment is provided in Section C.2.2.

C.2.1 Repeated Matrix Game

Based on the payoff matrix provided in Figure 6.3a, it is clear that the MCS of the Repeated Matrix Game environment consists of the three deterministic policies

	π(A)	π(B)	π(C)
1	1	0	0
2	0	1	0
3	0	1	0

(a) A set of policies that appear more optimal than MCS(E) for BRDiv and LIPO.

10	0	0
0	6	6
0	6	6

(b) Cross-play matrix for the policies discovered in Figure C.1a.

Figure C.1: An example failure mode of BRDiv and LIPO. The above figures provide an example set of policies that will appear to be more optimal than MCS(E) if we optimize the diversity metric used by LIPO and BRDiv.

displayed in Figure 6.5a. Ideally, L-BRDiv, BRDiv, and LIPO should all produce MCS^{est}(E) and Π^{train} containing policies displayed in Figure 6.5a. However, we show it is impossible to find $\alpha > 0$ that can make BRDiv and LIPO discover MCS^{est}(E) for this environment and generate a set of teammate policies to maximize the AHT agent's robustness.

LIPO and BRDiv fail in this simple environment because another set of policies produces a higher adversarial diversity metric compared to the ideal MCS^{est}(E) and Π^{train} for any $\alpha > 0$. An example set of policies producing a higher adversarial diversity metric than the ideal MCS^{est}(E) is displayed in Figure C.1. Compared to discovering MCS(E) as MCS^{est}(E) and Π^{train} that results in a cross-play matrix like the payoff matrix, the cross-play matrix from discovering policies in Figure C.1a has a lower sum of non-diagonal elements while having the same trace.

We now evaluate the value of LIPO and BRDiv's optimized diversity metric when both $\text{MCS}^{\text{est}}(\text{E})$ and Π^{train} equals MCS(E) and when it instead discovers the set of policies displayed in Figure C.1a, which we denote as Π^{alt}. Note that the adversarial diversity metric maximized by BRDiv, $\text{BRDiv}(\{\pi^i\}_{i=1}^K, \{\pi^{-i}\}_{i=1}^K)$, can be expressed as:

$$\begin{aligned}
&\sum_{i \in \{1,\ldots,K\}} \mathbb{E}_{s_0 \sim p_0}\left[\mathbf{R}_{i,-i}(H_t)\right] + \\
&\sum_{\substack{i,j \in \{1,\ldots,K\} \\ i \neq j}} \alpha \left(\mathbb{E}_{s_0 \sim p_0}\left[\mathbf{R}_{i,-i}(H_t) - \mathbf{R}_{j,-i}(H_t)\right]\right) + \\
&\sum_{\substack{i,j \in \{1,\ldots,K\} \\ i \neq j}} \alpha \left(\mathbb{E}_{s_0 \sim p_0}\left[\mathbf{R}_{i,-i}(H_t) - \mathbf{R}_{i,-j}(H_t)\right]\right),
\end{aligned} \tag{C.1}$$

for some $\alpha > 0$. Meanwhile, the adversarial diversity metric optimized by LIPO, $\text{LIPO}(\{\pi^i\}_{i=1}^K, \{\pi^{-i}\}_{i=1}^K)$, is given by the following expression:

$$\begin{aligned}
&\sum_{i \in \{1,\ldots,K\}} \mathbb{E}_{s_0 \sim p_0}\left[\mathbf{R}_{i,-i}(H_t)\right] - \\
&\sum_{\substack{i,j \in \{1,\ldots,K\} \\ i \neq j}} \alpha \left(\mathbb{E}_{s_0 \sim p_0}\left[\mathbf{R}_{j,-i}(H_t) + \mathbf{R}_{i,-j}(H_t)\right]\right),
\end{aligned} \tag{C.2}$$

assuming $\alpha > 0$. For $\alpha > 0$, the resulting BRDiv and LIPO objective for both sets of policies are provided in the following table: From Table C.1, it is clear that discovering

Table C.1: Value of LIPO and BRDiv objectives for the Repeated Matrix Game. The expressions that evaluate LIPO and BRDiv's optimized diversity metric for the Repeated Matrix Game are provided below. No $\alpha > 0$ enables MCS(E) to have higher diversity objectives than Π^{alt}.

Method	MCS(E)	Π^{alt}
BRDiv	22+56α	22+64α
LIPO	22-16α	22-12α

Π^{alt} will always produce higher diversity metrics for BRDiv and LIPO. It is then impossible to discover MCS(E) while optimizing both of these objectives. Its inability

	π(A)	π(B)	π(C)	π(D)
1	1	0	0	0
2	1	0	0	0
3	0	1	0	0
4	0	1	0	0

(a) Denoting π(X) as the probability of ending up in a corner cell having an ID of X, the above set of policies produce higher diversity metrics than MCS(E) in the Weighted Cooperative Reaching environment for BRDiv and LIPO.

10	10	0	0
10	10	0	0
0	0	10	10
0	0	10	10

(b) Cross-play matrix between policies discovered in Figure C.2a.

Figure C.2: Another example failure mode of BRDiv and LIPO in Weighted Cooperative Reaching. By not discovering policies that move towards corner cells C and D, BRDiv and LIPO can achieve a higher diversity metric than when discovering MCS(E).

to discover some members of MCS(E) and instead discover other members twice eventually leads LIPO and BRDiv towards producing AHT agents with significantly worse returns than L-BRDiv.

C.2.2 Weighted Cooperative Reaching

To show the shortcomings of LIPO and BRDiv in Weighted Cooperative Reaching, we also construct a set of policies that will produce higher diversity metrics for both BRDiv and LIPO. This set of policies that appears more desirable for LIPO and BRDiv than MCS(E) is denoted by Π^{alt} and is visualized by Figure C.2. Instead

of discovering four policies moving towards different corner cells in the environment, Π^{alt} discovers policies moving towards cells A and B twice. Discovering Π^{alt} and using it as MCS^{est}(E) and Π^{train} results in a cross-play matrix displayed in Figure C.2b.

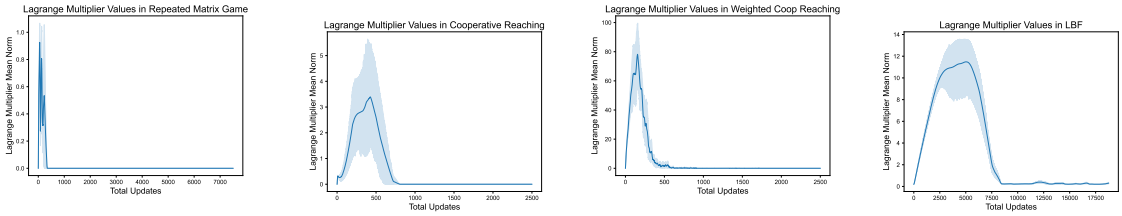
Compared to MCS(E) that produces a cross-play matrix that is the same as Figure 6.3c, the cross-play matrix from Π^{alt} has a higher sum of self-play returns and a lower sum of cross-play returns. As a result, no $\alpha > 0$ should make MCS(E) appear more desirable to LIPO and BRDiv. We show the expressions evaluating LIPO and BRDiv's diversity metrics for MCS(E) and Π^{alt} in Table C.2. Since a set of policies like Π^{alt} that does not discover all members of MCS(E) appear more preferable than MCS(E), LIPO and BRDiv end up yielding AHT agents that cannot robustly interact with teammate policies whose best-response policies are not discovered.

Table C.2: Value of LIPO and BRDiv objectives for Weighted Cooperative Reaching. The expressions that evaluate LIPO and BRDiv's optimized diversity metric for Weighted Cooperative Reaching are provided below. No $\alpha > 0$ enables MCS(E) to have higher diversity objectives than Π^{alt}.

Method	MCS(E)	Π^{alt}
BRDiv	36+120α	40+160α
LIPO	36-48α	40-40α

C.3 Analyzing the Lagrange Multipliers of L-BRDiv

The role of the Lagrange multipliers in the learning process undergone by L-BRDiv is highlighted in Figure C.3. Since the randomly initialized teammate policies cannot fulfill the upheld constraints in the beginning, optimizing Equation 6.12 encourages the increase of the Lagrange multipliers' values. The increasingly large Lagrange multipliers then force the learned policies to start fulfilling these constraints. Once policies learn to fulfil a constraint, the Lagrange multiplier associated with that constraint will decrease towards zero. At the end of the optimization process, we see that all Lagrange multipliers eventually converge to zero after all constraints are fulfilled.



(a) Repeated Matrix Game. **(b)** Coop Reaching. **(c)** Weighted Coop Reaching. **(d)** LBF.

Figure C.3: The changing values of L-BRDiv's Lagrange multipliers. Figure C.3a, Figure C.3b, Figure C.3c, and Figure C.3d all show how L-BRDiv's Lagrange multipliers change over time. Since a randomly initialized policy will not fulfill the constraints upheld by L-BRDiv, the Lagrange multipliers will initially increase their value to add more pressure to the policies to fulfill the constraints. Finally, the Lagrange multipliers will decrease to zero once constraints are fulfilled.

Table C.3: Hyperparameter values for L-BRDiv's Experiments. The specific hyperparameter values used in our teammate generation experiments in Repeated Matrix Games (RPM), Cooperative Reaching (CR), Weighted Cooperative Reaching (WCR), and Level-based Foraging (LBF) are provided below.

	RPM	CR	WCR	LBF
K	3	4	4	6
λ_π	10^{-3}	10^{-4}	10^{-4}	10^{-4}
λ_V	10^{-3}	10^{-4}	10^{-4}	10^{-4}
λ_α	0.05	0.5	0.5	0.05
γ	0.99	0.99	0.99	0.99
T	10^6	3.2×10^7	3.2×10^7	2.4×10^8
N_{threads}	40	160	160	160
T_{update}	2	8	8	8
T_{lagrange}	10	10	10	10
τ	1	0.2	0.5	0.1
w_{ent}	10^{-3}	5×10^{-3}	5×10^{-3}	8×10^{-4}

We ensure a fair comparison between L-BRDiv and the baseline methods by using the same hyperparameter values and network architecture. However, note that BRDiv and LIPO still require us to set α to a value that facilitates the generation of Π^{train} that facilitates the training of robust AHT agents. Since teammate generation and AHT training is computationally expensive , we follow these steps to tune α:

Table C.4: Network size for L-BRDiv's experiments. The size of models in our experiments in the Repeated Matrix Games (RPM), Cooperative Reaching (CR), Weighted Cooperative Reaching (WCR), and Level-based Foraging (LBF) environment are detailed below.

	RPM	CR	WCR	LBF
π^i_θ (Layer 1)	32	128	128	128
π^i_θ (Layer 2)	32	256	256	128
π^i_θ (Layer 3)	N/A	256	256	N/A
π^i_θ (Layer 4)	N/A	128	128	N/A
V_{θ_c} (Layer 1)	32	128	128	128
V_{θ_c} (Layer 2)	32	256	256	128
V_{θ_c} (Layer 3)	N/A	256	256	N/A
V_{θ_c} (Layer 4)	N/A	128	128	N/A

1. We initially run LIPO and BRDiv with $\alpha \in \{0.1, 0.5, 1, 5, 10\}$. Two experiment runs are done for each α.

2. We look at the generated teammates and see which tested α discover more members of MCS(E).

3. Based on the α producing the best estimate of MCS(E), we then do slight tuning to α by finding values close to α producing the best approximate to MCS(E).

Following this process, the final hyperparameter value that we end up using for LIPO and BRDiv is summarized in Table C.5. In alignment with the findings from Charakorn et al. (2023), note that LIPO ends up using small α values since larger α results in incompetent policies that cannot even achieve high returns against their intended partner in self-play. The only exception is Cooperative Reaching where MCS(E) consists of policies whose cross-play returns are zero, which enables the use of a large α. This emergence of incompetent policies is a natural consequence of optimizing Equation C.2, which cross-play return term's magnitude can overwhelm the self-play return term for large enough α.

Table C.5: α for baseline methods. The value of α used by baseline methods in their respective objectives for the Repeated Matrix Games (RPM), Cooperative Reaching (CR), Weighted Cooperative Reaching (WCR), and Level-based Foraging (LBF) environment are detailed below.

	RPM	CR	WCR	LBF
LIPO	0.5	8	0.25	0.08
BRDiv	1	10	1	0.4

C.4 AHT Experiment Hyperparameters

As we mention in Section 6.4.3, we use the RL^2 algorithm to train AHT agents based on the set of teammates generated by each compared method. The hyperparameters of the RL^2 algorithm are listed below:

- λ_π: Policy learning rate.
- λ_V: Critic learning rate.
- γ: Discount rate.
- T: Number of experiences used in learning.
- N_{threads}: Number of parallel threads for data collection during training.
- T_{update}: Number of timesteps between update.
- w_{ent}: Entropy weight term to encourage exploration.
- L_{rep}: The length of representation vectors to characterize teammates.

For each environment used in our experiments, hyperparameter values that we use in each environment is provided in Table C.6.

Apart from these hyperparameters, our policy and critic networks have a similar architecture to the teammate generation process. The only difference is that we use an LSTM layer as our final layer. We use the LSTM layer to enable agents to

Table C.6: Hyperparameter values for L-BRDiv's Experiments. The specific hyperparameter values used in our Repeated Matrix Games (RPM), Cooperative Reaching (CR), Weighted Cooperative Reaching (WCR), and Level-based Foraging (LBF) environments are provided below.

	RPM	CR	WCR	LBF
λ_π	10^{-4}	10^{-4}	10^{-4}	10^{-4}
λ_V	10^{-4}	10^{-4}	10^{-4}	10^{-4}
γ	0.99	0.99	0.99	0.99
T	10^6	1.2×10^7	1.2×10^7	4.8×10^7
N_{threads}	10	16	16	16
T_{update}	2	8	8	8
w_{ent}	10^{-4}	2.5×10^{-4}	2.5×10^{-4}	8×10^{-4}
L_{rep}	16	32	32	64

process the previous sequence of observations and experienced rewards to model the type of teammates the AHT agent is interacting with.

Appendix D

Additional Details on Traffic Congestion Reduction

This appendix provides supplementary implementation details for the experiments presented in Chapter 7, which investigates centralized and distributed learning approaches for reducing traffic congestion using reinforcement learning. Specifically, we detail the full set of hyperparameters used for training both centralized and distributed agents from scratch, as well as the configuration for simulating human-like driving behavior via proxy controllers.

A Hyper-parameters

A.1 Centralized Agent Training

Table D.1 lists the hyperparameters used to train centralized agents from scratch using Proximal Policy Optimization (PPO). These settings were employed in the experiments for the Simple Merge and I-696 Merge scenarios, corresponding to results reported in Table 7.1 and Table 7.2, respectively. The centralized controller operates with full access to the joint observation space and takes a global action across all controlled vehicles.

Table D.1: Hyper-parameters for training centralized agents from scratch

Parameter	Value
Algorithm	Proximal Policy Optimization (PPO)
Horizon	2000
Simulation Time Stepsize	0.5
Optimizer	Stochastic Gradient Descent
Learning Rate	5×10^{-4}
Discount Factor (γ)	0.99
GAE Lambda (λ)	0.97
Actor Critic	True
Value Function Clip Parameter	10^6
Number of SGD Update per Iteration	10
Model hiddens	[100,50,25]
Clip Parameter	0.3
Entropy Coefficient	0
Sgd Minibatch size	128
Train Batch Size	40000
Value Function Share Layers	False
KL Coefficient	0.2
KL Target	0.01
Max Acceleration	2.6
Max Deceleration	4.5
Training Iterations	500
Number of Rollouts per Iteration	20

A.2 Distributed Agent Training

Table D.2 summarizes the hyperparameters used to train distributed agents in a decentralized setting, where each agent operates based on its local observations. The agents are jointly optimized using PPO with synchronized updates. These settings were applied in experiments investigating the effects of feature augmentation and reward shaping, as reported in Table 7.3 and Table 7.4.

Table D.2: Hyper-parameters for training distributed agents from scratch

Parameter	Value
Algorithm	Proximal Policy Optimization (PPO)
Horizon	2000
Simulation Time Stepsize	0.5
Optimizer	Stochastic Gradient Descent
Learning Rate	piece-wise linearly decreasing starting from 5×10^{-4} (From scratch)
Discount Factor (γ)	0.998
GAE Lambda (λ)	0.95
Actor Critic	True
Value Function Clip Parameter	10^8
Number of SGD Update per Iteration	10
Model hiddens	[100,50,25]
Clip Parameter	0.2
Entropy Coefficient	10^{-3}
Sgd Minibatch size	4096
Train Batch Size	60000
Value Function Share Layers	True
Value Loss Coefficient	0.5
KL Coefficient	0.01
KL Target	0.01
Max Acceleration	2.6
Max Deceleration	4.5
Training Iterations	500
Number of Rollouts per Iteration	30

A.3 Human Vehicles

To simulate background traffic, we employ a human-proxy controller based on the default Intelligent Driver Model (IDM) provided by SUMO. The hyperparameters for this controller are provided in Table D.3. These settings aim to approximate realistic driving behavior and are held fixed across all training and evaluation episodes involving mixed-autonomy scenarios.

Table D.3: Hyper-parameters for human-proxy controller

Parameter	Value
Controller	Sumo Default Controller(IDM)
Max Acceleration	2.6
Max Deceleration	4.5
Expected Time Headway	1 second

References

Works Cited

J. P. Agapiou, A. S. Vezhnevets, E. A. Duéñez-Guzmán, J. Matyas, Y. Mao, P. Sunehag, R. Köster, U. Madhushani, K. Kopparapu, R. Comanescu, et al. Melting pot 2.0. *arXiv preprint arXiv:2211.13746*, 2022.

C. Angliss, J. Cui, J. Hu, A. Rahman, and P. Stone. A benchmark for generalizing across diverse team strategies in competitive pok\'emon. *arXiv preprint arXiv:2506.10326*, 2025.

A. H. Anwar, G. Atia, and M. Guirguis. Toward a protected cloud against side channel attacks: A game-theoretic framework. In *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 78–83. IEEE, 2018.

N. Baimukan and Q. Zhu. Concealment charm (ConcealGAN): Automatic generation of steganographic text using generative models to bypass censorship. *Game Theory and Machine Learning for Cyber Security*, pages 357–365, 2021.

A. Bakhtin, N. Brown, E. Dinan, G. Farina, C. Flaherty, D. Fried, A. Goff, J. Gray, H. Hu, et al. Human-level play in the game of diplomacy by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074, 2022a.

A. Bakhtin, D. J. Wu, A. Lerer, J. Gray, A. P. Jacob, G. Farina, A. H. Miller, and N. Brown. Mastering the game of no-press diplomacy via human-

regularized reinforcement learning and planning. *arXiv preprint arXiv:2210.05492*, 2022b.

M. Bansal, A. Krizhevsky, and A. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018.

N. Bard, J. N. Foerster, S. Chandar, N. Burch, M. Lanctot, H. F. Song, E. Parisotto, V. Dumoulin, S. Moitra, E. Hughes, et al. The hanabi challenge: A new frontier for ai research. *Artificial Intelligence*, 280:103216, 2020.

S. Barrett, A. Rosenfeld, S. Kraus, and P. Stone. Making friends on the fly: Cooperating with new teammates. *Artificial Intelligence*, October 2016. doi: 10.1016/j.artint.2016.10.005. URL http://www.sciencedirect.com/science/article/pii/S0004370216301266.

D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4):819–840, 2002.

D. Bezzina, M. L. Buonarosa, et al. Ccat ann arbor connected environment (aace) operations and maintenance. Technical report, University of Michigan. Center for Connected and Automated Transportation, 2023.

V. Bier, S. Oliveros, and L. Samuelson. Choosing what to protect: Strategic defensive allocation against an unknown attacker. *Journal of Public Economic Theory*, 9(4):563–587, 2007.

N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, et al. The gem5 simulator. *ACM SIGARCH computer architecture news*, 39(2):1–7, 2011.

N. Brandizzi, D. Grossi, and L. Iocchi. Rlupus: Cooperation through emergent communication in the werewolf social deduction game. *Intelligenza Artificiale*, 15(2):55–70, 2022.

S. Briongos, P. Malagón, J. M. Moya, and T. Eisenbarth. RELOAD+ REFRESH: Abusing cache replacement policies to perform stealthy cache attacks. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1967–1984, 2020.

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016a.

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016b.

G. W. Brown. Iterative solution of games by fictitious play. *Act. Anal. Prod Allocation*, 13(1):374, 1951.

J. Bucek, K.-D. Lange, and J. v. Kistowski. Spec cpu2017: Next-generation compute benchmark. In *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering*, pages 41–42, 2018.

H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

R. Charakorn, P. Manoonpong, and N. Dilokthanakul. Generating diverse cooperative agents by learning incompatible policies. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=UkU05GOH7_6.

D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl. Learning by cheating. In *Conference on Robot Learning*, pages 66–75. PMLR, 2020.

D. Chen, V. Koltun, and P. Krähenbühl. Learning to drive from a world on rails. In *arXiv preprint arXiv:2105.00636*, 2021.

J. Chen and G. Venkataramani. Cc-hunter: Uncovering covert timing channels on shared processor hardware. In *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 216–228. IEEE, 2014.

Q. Chen, S. Tang, Q. Yang, and S. Fu. Cooper: Cooperative perception for connected autonomous vehicles based on 3d point clouds. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 514–524. IEEE, 2019.

F. Christianos, L. Schäfer, and S. V. Albrecht. Shared experience actor-critic for multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4693–4700. IEEE, 2018.

F. Codevilla, E. Santana, A. M. López, and A. Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9329–9338, 2019.

B. Cui, A. Lupu, S. Sokota, H. Hu, D. J. Wu, and J. N. Foerster. Adversarial diversity in hanabi. In *The Eleventh International Conference on Learning Representations*, 2023a.

C. Cui, Y. Ma, X. Cao, W. Ye, and Z. Wang. Receive, reason, and react: Drive as you say with large language models in autonomous vehicles. *arXiv preprint arXiv:2310.08034*, 2023b.

J. Cui, H. Qiu, D. Chen, P. Stone, and Y. Zhu. Coopernaut: end-to-end driving with cooperative perception for networked vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17252–17262, 2022.

J. Cui, X. Yang, M. Luo, G. Lee, P. Stone, H.-H. S. Lee, B. Lee, G. E. Suh, W. Xiong, and Y. Tian. MACTA: A multi-agent reinforcement learning approach for cache timing attacks and detection. In *The Eleventh International Conference on Learning Representations*, 2023c. URL https://openreview.net/forum?id=CDlHZ78-Xzi.

X. Ding, J. Han, H. Xu, W. Zhang, and X. Li. HiLM-D: Towards high-resolution understanding in multimodal large language models for autonomous driving. *arXiv preprint arXiv:2309.05186*, 2023.

A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. Carla: An open urban driving simulator. *arXiv preprint arXiv:1711.03938*, 2017.

A. Downs. *Stuck in traffic: Coping with peak-hour traffic congestion*. Brookings Institution Press, 2000.

K. Dresner. AIM: autonomous intersection management., 01 2008.

Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel. RL^2: Fast reinforcement learning via slow reinforcement learning, 2016.

I. Durugkar, E. Liebman, and P. Stone. Balancing individual preferences and shared objectives in multiagent reinforcement learning. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI 2020)*, July 2020.

T. Eghtesad, Y. Vorobeychik, and A. Laszka. Adversarial deep reinforcement learning based adaptive moving target defense. In *International Conference on Decision and Game Theory for Security*, pages 58–79. Springer, 2020.

Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing*, 6(9):1305–1315, 1997.

R. Elderman, L. J. Pater, A. S. Thie, M. M. Drugan, and M. A. Wiering. Adversarial reinforcement learning in a cyber security simulation. In *ICAART (2)*, pages 559–566, 2017.

S. Fan, S. M. Zahedi, and B. C. Lee. Distributed strategies for computational sprints. *Communications of the ACM*, 62(2):98–106, 2019.

L. Gallo and J. Harri. Short paper: A lte-direct broadcast mechanism for periodic vehicular safety communications. In *IEEE Vehicular Networking Conference 2013*, pages 166–169. IEEE, Dec 2013. doi: 10.1109/VNC.2013.6737604.

X. Gao, Y. Wu, R. Wang, C. Liu, Y. Zhou, and Z. Tu. LangCoop: Collaborative driving with language. *arXiv preprint arXiv:2504.13406*, 2025.

P. Gu, M. Zhao, J. Hao, and B. An. Online ad hoc teamwork under partial observability. In *International Conference on Learning Representations*, 2021.

Y. Guo, X. Xin, Y. Zhang, and J. Yang. Leaky way: a conflict-based cache covert channel bypassing set associativity. In *2022 IEEE International Symposium on Microarchitecture (MICRO)*, pages 1458–1473. IEEE, 2022a.

Y. Guo, A. Zigerelli, Y. Zhang, and J. Yang. Adversarial prefetch: New cross-core cache side channel attacks. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1458–1473. IEEE, 2022b.

A. Harris, S. Wei, P. Sahu, P. Kumar, T. Austin, and M. Tiwari. Cyclone: Detecting contention-based cache information leaks through cyclic interference. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 57–72, 2019.

P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016a.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016b.

J. Heinrich and D. Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

S. Hu, Z. Fang, Z. Fang, X. Chen, and Y. Fang. AgentsCoDriver: Large language model empowered collaborative driving with lifelong learning. *arXiv preprint arXiv:2404.06345*, 2024.

iSmartWays Technology Inc. ismartways performance measurement, 2018. URL https://fccid.io/2AQQ3IM2RSE/Test-Report/FCC-Part22-4039626.

K. Jang, E. Vinitsky, B. Chalaki, B. Remer, L. Beaver, A. A. Malikopoulos, and A. Bayen. Simulation to scaled city: zero-shot policy transfer for traffic control via autonomous vehicles. In *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, pages 291–300, 2019.

Y. Jin, X. Shen, H. Peng, X. Liu, J. Qin, J. Li, J. Xie, P. Gao, G. Zhou, and J. Gong. Surrealdriver: Designing generative driver agent simulation framework in urban contexts based on large language model. *arXiv preprint arXiv:2309.13193*, 2023.

B. Johannesmeyer, J. Koschel, K. Razavi, H. Bos, and C. Giuffrida. Kasper: Scanning for generalized transient execution gadgets in the linux kernel. In *NDSS Symposium 2022*, 2022.

J. B. Kenney. Dedicated short-range communications (dsrc) standards in the united states. *Proceedings of the IEEE*, 99(7):1162–1182, July 2011. ISSN 0018-9219. doi: 10.1109/JPROC.2011.2132790.

N. Kheterpal, K. Parvate, C. Wu, A. Kreidieh, E. Vinitsky, and A. Bayen. Flow: Deep reinforcement learning for control in sumo. *EPiC Series in Engineering*, 2:134–151, 2018.

P. Kocher, J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, et al. Spectre attacks: Exploiting speculative execution. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1–19. IEEE, 2019.

D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker. Recent development and applications of sumo-simulation of urban mobility. *International journal on advances in systems and measurements*, 5(3&4), 2012.

A. R. Kreidieh, C. Wu, and A. M. Bayen. Dissipating stop-and-go waves in closed and open networks via deep reinforcement learning. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 1475–1480. IEEE, 2018.

V. Krishnamurthy, M. Maskery, and M. H. Ngo. Game theoretic activation and transmission scheduling in unattended ground sensor networks: A correlated equilibrium approach. *Wireless Sensor Networks: Signal Processing and Communications Perspectives*, pages 349–388, 2007.

M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, K. Tuyls, J. Pérolat, D. Silver, and T. Graepel. A unified game-theoretic approach to multiagent reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019.

J. Laufer. Freeway capacity, saturation flow and the car following behavioural algorithm of the vissim microsimulation software. In *30th Australasian Transport Research Forum*, volume 25, 2007.

A. Lazaridou, A. Potapenko, and O. Tieleman. Multi-agent communication meets natural language: Synergies between functional and structural language learning. *arXiv preprint arXiv:2005.07064*, 2020.

M. Lewis, D. Yarats, Y. N. Dauphin, D. Parikh, and D. Batra. Deal or no deal? end-to-end learning for negotiation dialogues. *arXiv preprint arXiv:1706.05125*, 2017.

Q. Li, Y. Wang, Y. Wang, and H. Zhao. Hdmapnet: A local semantic map learning and evaluation framework, 2021a.

Y. Li, S. Ren, P. Wu, S. Chen, C. Feng, and W. Zhang. Learning distilled collaboration graph for multi-agent perception. In *Thirty-fifth Conference on Neural Information Processing Systems (NeurIPS 2021)*, 2021b.

N. Lichtlé, E. Vinitsky, M. Nice, R. Bhadani, M. Bunting, F. Wu, B. Piccoli, B. Seibold, D. B. Work, J. W. Lee, et al. From sim to real: A pipeline for training and deploying traffic smoothing cruise controllers. *IEEE Transactions on Robotics*, 2024.

M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, A. Fogh, J. Horn, S. Mangard, P. Kocher, D. Genkin, et al. Meltdown: Reading kernel memory from user space. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 973–990, 2018.

F. Liu, Y. Yarom, Q. Ge, G. Heiser, and R. B. Lee. Last-level cache side-channel attacks are practical. In *2015 IEEE symposium on security and privacy*, pages 605–622. IEEE, 2015.

M. Luo, W. Xiong, G. Lee, Y. Li, X. Yang, A. Zhang, Y. Tian, H. H. S. Lee, and G. E. Suh. AutoCAT: Reinforcement learning for automated exploration of cache-timing attacks. In *29th Sympisum on High Performance Computer Architecture (HPCA)*, 2023.

A. Lupu, B. Cui, H. Hu, and J. Foerster. Trajectory diversity for zero-shot coordination. In *International conference on machine learning*, pages 7204–7213. PMLR, 2021.

Y. Ma, Y. Cao, J. Sun, M. Pavone, and C. Xiao. Dolphins: Multimodal language model for driving, 2023.

A. Mahajan, T. Rashid, M. Samvelyan, and S. Whiteson. Maven: Multi-agent variational exploration. *Advances in Neural Information Processing Systems*, 32, 2019.

J. Mao, M. Niu, C. Jiang, H. Liang, J. Chen, X. Liang, Y. Li, C. Ye, W. Zhang, Z. Li, et al. One million scenes for autonomous driving: Once dataset. *arXiv preprint arXiv:2106.11037*, 2021.

J. Mao, Y. Qian, H. Zhao, and Y. Wang. Gpt-driver: Learning to drive with gpt. *arXiv preprint arXiv:2310.01415*, 2023a.

J. Mao, J. Ye, Y. Qian, M. Pavone, and Y. Wang. A language agent for autonomous driving, 2023b.

H. B. McMahan, G. J. Gordon, and A. Blum. Planning in the presence of cost functions controlled by an adversary. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 536–543, 2003.

J. Meinilä, P. Kyösti, T. Jämsä, and L. Hentilä. Winner ii channel models, 2009.

S. Mirbagher-Ajorpaz, G. Pokam, E. Mohammadian-Koruyeh, E. Garza, N. Abu-Ghazaleh, and D. A. Jiménez. Perspectron: Detecting invariant footprints of microarchitectural attacks with perceptron. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1124–1137. IEEE, 2020.

S. Mirbagher-Ajorpaz, D. Moghimi, J. N. Collins, G. Pokam, N. Abu-Ghazaleh, and D. Tullsen. EVAX: Towards a practical, pro-active & adaptive architecture for high performance & security. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1218–1236. IEEE, 2022.

R. Mirsky, I. Carlucho, A. Rahman, E. Fosong, W. Macke, M. Sridharan, P. Stone, and S. V. Albrecht. A survey of ad hoc teamwork research. In *European Conference on Multi-Agent Systems*, pages 275–293. Springer, 2022.

J.-B. Mouret and J. Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

W. G. Najm, R. Ranganathan, G. Srinivasan, J. D. Smith, S. Toma, E. Swanson, A. Burgett, et al. Description of light-vehicle pre-crash scenarios for safety applications based on vehicle-to-vehicle communications. Technical report, United States. National Highway Traffic Safety Administration, 2013.

National Academy of Engineering. 14 grand challenges for engineering of the 21st century., 2007. URL http://www.engineeringchallenges.org/challenges/cyberspace.aspx.

M. Nie, R. Peng, C. Wang, X. Cai, J. Han, H. Xu, and L. Zhang. Reason2Drive: Towards interpretable and chain-based reasoning for autonomous driving. *arXiv preprint arXiv:2312.03661*, 2023.

O. Oleksenko, B. Trach, M. Silberstein, and C. Fetzer. SpecFuzz: Bringing spectre-type vulnerabilities to the surface. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1481–1498, 2020.

Y. Oren, V. P. Kemerlis, S. Sethumadhavan, and A. D. Keromytis. The spy in the sandbox: Practical cache attacks in javascript and their implications. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1406–1418, 2015.

D. A. Osvik, A. Shamir, and E. Tromer. Cache attacks and countermeasures: the case of AES. In *Cryptographers' track at the RSA conference*, pages 1–20. Springer, 2006.

G. Papoudakis, F. Christianos, and S. Albrecht. Agent modelling under partial observability for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35, 2021.

E. Parisotto, F. Song, J. Rae, R. Pascanu, C. Gulcehre, S. Jayakumar, M. Jaderberg, R. L. Kaufman, A. Clark, S. Noury, et al. Stabilizing transformers for reinforcement learning. In *International conference on machine learning*, pages 7487–7498. PMLR, 2020.

J. Parker-Holder, A. Pacchiano, K. M. Choromanski, and S. J. Roberts. Effective diversity in population based reinforcement learning. *Advances in Neural Information Processing Systems*, 33:18050–18062, 2020a.

J. Parker-Holder, A. Pacchiano, K. M. Choromanski, and S. J. Roberts. Effective diversity in population based reinforcement learning. *Advances in Neural Information Processing Systems*, 33:18050–18062, 2020b.

A. Petrenko, Z. Huang, T. Kumar, G. S. Sukhatme, and V. Koltun. Sample factory: Egocentric 3d control from pixels at 100000 FPS with asynchronous reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 7652–7662. PMLR, 2020. URL http://proceedings.mlr.press/v119/petrenko20a.html.

J. Plungis. Toyota's v2v move shows industry still interested in cars talking to each other, 2018. URL https://www.consumerreports.org/automotive-technology/toyota-v2v-vehicle-to-vehicle-communications/.

D. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *NeurPIS*, 1988.

A. Prakash, K. Chitta, and A. Geiger. Multi-modal fusion transformer for end-to-end autonomous driving. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

J. K. Pugh, L. B. Soros, and K. O. Stanley. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, page 40, 2016.

C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017a.

C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017b.

Z. Qi, Q. Feng, Y. Cheng, M. Yan, P. Li, H. Yin, and T. Wei. SpecTaint: Speculative taint analysis for discovering spectre gadgets. In *NDSS*, 2021.

T. Qian, J. Chen, L. Zhuo, Y. Jiao, and Y.-G. Jiang. NuScenes-QA: A multi-modal visual question answering benchmark for autonomous driving scenario. *arXiv preprint arXiv:2305.14836*, 2023.

H. Qiu, F. Ahmad, F. Bai, M. Gruteser, and R. Govindan. Avr: Augmented vehicular reality. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, pages 81–95, 2018.

H. Qiu, P. Huang, N. Asavisanu, X. Liu, K. Psounis, and R. Govindan. Autocast: Scalable infrastructure-less cooperative perception for distributed collaborative driving. *CoRR*, abs/2112.14947, 2021. URL https://arxiv.org/abs/2112.14947.

Qualcomm. Lte direct proximity services, 2019. URL https://www.qualcomm.com/invention/technologies/lte/direct.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

A. Rahman, N. Höpner, F. Christianos, and S. V. Albrecht. Towards open ad hoc teamwork using graph-based policy learning. In *International Conference on Machine Learning*, volume 139. PMLR, 2021.

A. Rahman, E. Fosong, I. Carlucho, and S. V. Albrecht. Generating teammates for training robust ad hoc teamwork agents via best-response diversity. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=l5BzfQhROl.

J. Ravichandran, W. T. Na, J. Lang, and M. Yan. Pacman: attacking arm pointer authentication with speculative execution. In *ISCA*, pages 685–698, 2022.

S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011a.

S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011b.

T. Roughgarden. Algorithmic game theory. *Communications of the ACM*, 53 (7):78–86, 2010.

G. Saileshwar, C. W. Fletcher, and M. Qureshi. Streamline: a fast, flushless cache covert-channel attack by enabling asynchronous collusion. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1077–1090, 2021.

V. Sanh, L. Debut, J. Chaumond, and T. Wolf. DistilBERT, a distilled version of bert: smaller, faster, cheaper and lighter. In *NeurIPS EMC2 Workshop*, 2019.

A. Sauer, N. Savinov, and A. Geiger. Conditional affordance learning for driving in urban environments. In *Conference on Robot Learning*, pages 237–252. PMLR, 2018.

A. Schlenker, H. Xu, M. Guirguis, C. Kiekintveld, A. Sinha, M. Tambe, S. Sonya, D. Balderas, and N. Dunstatter. Don't bury your head in warnings: A game-theoretic approach for intelligent allocation of cyber-security alerts, 2017.

A. Schlenker et al. Deceiving cyber adversaries: A game theoretic approach. In *International Conference on Autonomous Agents and Multi Agent Systems*, 2018.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

H. Sha, Y. Mu, Y. Jiang, L. Chen, C. Xu, P. Luo, S. E. Li, M. Tomizuka, W. Zhan, and M. Ding. Languagempc: Large language models as decision makers for autonomous driving. *arXiv preprint arXiv:2310.03026*, 2023.

H. Shao, Y. Hu, L. Wang, S. L. Waslander, Y. Liu, and H. Li. LMDrive: Closed-loop end-to-end driving with large language models. *arXiv preprint arXiv:2312.07488*, 2023.

S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *nature*, 529 (7587):484–489, 2016.

C. Sima, K. Renz, K. Chitta, L. Chen, H. Zhang, C. Xie, P. Luo, A. Geiger, and H. Li. DriveLM: Driving with graph visual question answering. *arXiv preprint arXiv:2312.14150*, 2023.

R. Stahlmann, A. Festag, A. Tomatis, I. Radusch, and F. Fischer. Starting european field tests for car-2-x communication: the drive c2x framework. In *ITS World Congress and Exhibition*, Oct 2011.

R. E. Stern, S. Cui, M. L. Delle Monache, R. Bhadani, M. Bunting, M. Churchill, N. Hamilton, H. Pohlmann, F. Wu, B. Piccoli, et al. Dissipation of stop-and-go

waves via control of autonomous vehicles: Field experiments. *Transportation Research Part C: Emerging Technologies*, 89:205–221, 2018.

P. Stone, G. Kaminka, S. Kraus, and J. Rosenschein. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 1504–1509, 2010.

D. Strouse, K. McKee, M. Botvinick, E. Hughes, and R. Everett. Collaborating with humans without human data. *Advances in neural information processing systems*, 34:14502–14515, 2021.

Y. Sugiyama, M. Fukui, M. Kikuchi, K. Hasebe, A. Nakayama, K. Nishinari, S.-i. Tadaki, and S. Yukawa. Traffic jams without bottlenecks—experimental evidence for the physical mechanism of the formation of a jam. *New journal of physics*, 10(3):033001, 2008.

P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993.

M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7), 2009.

O. Thakoor, P. Vayanos, M. Tambe, and M. Yu. Game theory for strategic DDoS mitigation, 2020.

C. Thompson. 18 awesome innovations in the new mercedes e-class, 2016. URL https://www.businessinsider.com/mercedes-e-class-2017-features-2016-6.

M. Toromanoff, E. Wirbel, and F. Moutarde. End-to-end model-free reinforcement learning for urban driving using implicit affordances. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

M. Treiber and A. Kesting. The intelligent driver model with stochasticity-new insights into traffic flow oscillations. *Transportation research procedia*, 23: 174–187, 2017.

M. Treiber, A. Hennecke, and D. Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 62(2):1805, 2000.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

P. Vila, P. Ganty, M. Guarnieri, and B. Köpf. CacheQuery: Learning replacement policies from hardware caches. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 519–532, 2020.

E. Vinitsky, A. Kreidieh, L. Le Flem, N. Kheterpal, K. Jang, C. Wu, F. Wu, R. Liaw, E. Liang, and A. M. Bayen. Benchmarks for reinforcement learning in mixed-autonomy traffic. In *Conference on Robot Learning*, pages 399–409, 2018.

O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

C. Wang, A. Rahman, J. Cui, Y. Sung, and P. Stone. ROTATE: Regret-driven open-ended training for ad hoc teamwork. *arXiv preprint arXiv:2505.23686*, 2025.

T.-H. Wang, S. Manivasagam, M. Liang, Y. Bin, W. Zeng, J. Tu, and R. Urtasun. V2VNet: Vehicle-to-vehicle communication for joint perception and prediction. In *ECCV*, 2020.

X. Wang, S. Zhang, W. Zhang, W. Dong, J. Chen, Y. Wen, and W. Zhang. Zsc-eval: An evaluation toolkit and benchmark for multi-agent zero-shot coordination. *Advances in Neural Information Processing Systems*, 37:47344–47377, 2024.

Y. Wang, Z. R. Shi, L. Yu, Y. Wu, R. Singh, L. Joppa, and F. Fang. Deep reinforcement learning for green security games with real-time information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1401–1408, 2019.

Wayve. LINGO-1: Exploring natural language for autonomous driving, 2023. URL https://wayve.ai/thinking/lingo-natural-language-autonomous-driving/.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

M. P. Wellman. Methods for empirical game-theoretic analysis. In *AAAI*, volume 980, pages 1552–1556, 2006.

M. P. Wellman, K. Tuyls, and A. Greenwald. Empirical game theoretic analysis: A survey. *Journal of Artificial Intelligence Research*, 82:1017–1076, 2025.

L. Wen, D. Fu, X. Li, X. Cai, T. Ma, P. Cai, M. Dou, B. Shi, L. He, and Y. Qiao. Dilu: A knowledge-driven approach to autonomous driving with large language models. *arXiv preprint arXiv:2309.16292*, 2023a.

L. Wen, X. Yang, D. Fu, X. Wang, P. Cai, X. Li, T. Ma, Y. Li, L. Xu, D. Shang, et al. On the road with gpt-4v (ision): Early explorations of visual-language model on autonomous driving. *arXiv preprint arXiv:2311.05332*, 2023b.

C. Wu, A. Kreidieh, K. Parvate, E. Vinitsky, and A. M. Bayen. Flow: Architecture and benchmarking for reinforcement learning in traffic control. *arXiv preprint arXiv:1710.05465*, page 10, 2017.

C. Wu, A. M. Bayen, and A. Mehta. Stabilizing traffic with autonomous vehicles. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6012–6018, 2018.

S. Wu, J. Yao, H. Fu, Y. Tian, C. Qian, Y. Yang, Q. FU, and Y. Wei. Quality-similar diversity via population based reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=bLmSMXbqXr.

D. Xing, Q. Liu, Q. Zheng, G. Pan, and Z. Zhou. Learning with generated teammates to achieve type-free ad-hoc teamwork. In *IJCAI*, pages 472–478, 2021.

W. Xiong and J. Szefer. Leaking information through cache LRU states. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 139–152. IEEE, 2020.

R. Xu, H. Xiang, X. Xia, X. Han, J. Liu, and J. Ma. OPV2V: An open benchmark dataset and fusion pipeline for perception with vehicle-to-vehicle communication. In *2022 IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

Y. Xu, S. Wang, P. Li, F. Luo, X. Wang, W. Liu, and Y. Liu. Exploring large language models for communication games: An empirical study on werewolf. *arXiv preprint arXiv:2309.04658*, 2023a.

Z. Xu, Y. Zhang, E. Xie, Z. Zhao, Y. Guo, K. K. Wong, Z. Li, and H. Zhao. Drivegpt4: Interpretable end-to-end autonomous driving via large language model. *arXiv preprint arXiv:2310.01412*, 2023b.

M. Yan, Y. Shalabi, and J. Torrellas. ReplayConfusion: detecting cache-based covert channel attacks using record and replay. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–14. IEEE, 2016.

B. Yang, M. Liang, and R. Urtasun. Hdnet: Exploiting hd maps for 3d object detection. In *Conference on Robot Learning*, pages 146–155. PMLR, 2018.

S. Yang, J. Liu, R. Zhang, M. Pan, Z. Guo, X. Li, Z. Chen, P. Gao, Y. Guo, and S. Zhang. Lidar-llm: Exploring the potential of large language models for 3d lidar understanding. *arXiv preprint arXiv:2312.14074*, 2023.

X. Yang, B. Cui, T. Li, and Y. Tian. RLMeta: A flexible framework for distributed reinforcement learning, 1 2022. URL https://github.com/facebookresearch/rlmeta.

Y. Yarom and K. Falkner. FLUSH+ RELOAD: A high resolution, low noise, l3 cache side-channel attack. In *23rd USENIX security symposium (USENIX security 14)*, pages 719–732, 2014.

D. Ye, Z. Liu, M. Sun, B. Shi, P. Zhao, H. Wu, H. Yu, S. Yang, X. Wu, Q. Guo, et al. Mastering complex control in moba games with deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6672–6679, 2020.

C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu. The surprising effectiveness of PPO in cooperative multi-agent games. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL https://openreview.net/forum?id=YVXaxB6L2Pl.

J. Yu, T. Chen, C. Gutterman, S. Zhu, G. Zussman, I. Seskar, and D. Kilper. Cosmos: Optical architecture and prototyping. In *2019 Optical Fiber Communications Conference and Exhibition (OFC)*, pages 1–3. IEEE, 2019.

X. Zhang, A. Zhang, J. Sun, X. Zhu, Y. E. Guo, F. Qian, and Z. M. Mao. Emp: Edge-assisted multi-vehicle perception. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, MobiCom '21, page 545–558, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383424. doi: 10.1145/3447993.3483242. URL https://doi.org/10.1145/3447993.3483242.

H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16259–16268, October 2021.

Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.

L. Zintgraf, S. Devlin, K. Ciosek, S. Whiteson, and K. Hofmann. Deep interactive bayesian reinforcement learning via meta-learning. *arXiv preprint arXiv:2101.03864*, 2021.

B.-E. Zolbayar, R. Sheatsley, P. McDaniel, and M. Weisman. Evading machine learning based network intrusion detection systems with GANs. *Game Theory and Machine Learning for Cyber Security*, pages 335–356, 2021.