INTERPRETABLE ARTIFICIAL INTELLIGENCE FOR PERSONALIZED HUMAN-ROBOT COLLABORATION

A Dissertation
Presented to
The Academic Faculty

By

Rohan Paleja

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Mechanical Engineering
Institute for Robotics & Intelligent Machines

Georgia Institute of Technology

December 2023

INTERPRETABLE ARTIFICIAL INTELLIGENCE FOR PERSONALIZED HUMAN-ROBOT COLLABORATION

Thesis committee:

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

Dr. Dorsa Sadigh
Computer Science Department
Stanford University

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

Dr. Peter Stone
Computer Science Department
The University of Texas at Austin

Dr. Seth Hutchinson
School of Interactive Computing
Georgia Institute of Technology

Date approved: August 16, 2020

It's the questions we can't answer that teach us the most. They teach us how to think. If you give a man an answer, all he gains is a little fact. But give him a question and he'll look for his own answers.

Patrick Rothfuss, The Wise Man's Fear

ACKNOWLEDGMENTS

I am incredibly grateful for the support and mentorship from my advisor, Dr. Matthew Gombolay. His endless support, continual guidance, and amazing mentorship have been absolutely essential in the preparation and completion of this work. Moreover, he has helped me grow as an academic and a person. Thank you!

I would like to thank the members of my thesis committee for their help in preparation of this work – Harish Ravichandar, whose mentorship, advice, and discussion have been incredibly beneficial in growing this work to what it is, Dorsa Sadigh, who helped to broaden my view through her questions during my thesis proposal and related work, Peter Stone, who always has great advice, and Seth Hutchinson, who inspires me to be better and challenges me through his difficult line of questioning.

To the family, Roshni, Rahul, Mom, Dad, without your support, I would not have made it this far. To all my amazing friends in the CORE Robotics Lab, Esi, Zac, Andrew, Mariah, Erin, Prad, Manisha, and everyone else I've interacted with, thank you for all the fruitful discussion, fun parties, and amazing experiences. To all my co-authors, thank you for all your support, discussion, and contribution.

The author gratefully acknowledges the support for this work offered by MIT Lincoln Laboratory, Sandia National Laboratories, the Office of Naval Research, Lockheed Martin, and Konica Minolta. Any views and conclusions contained herein are those of the author, and do not necessarily represent the official positions, express or implied, of the funders.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Collaborative robots (i.e., "cobots") and machine learning-based virtual agents are increasingly entering the human workspace with the aim of increasing productivity, enhancing safety, and improving the quality of our lives [1, 2]. These agents will dynamically interact with a wide variety of people in dynamic and novel contexts, increasing the prevalence of human-machine teams in healthcare [3], manufacturing [4], and search-and-rescue [5]. Within these domains, collaborators must have aligned objectives and maintain awareness over other agents' behaviors to avoid potential accidents. It is critical that AI agents are able to understand the similarities and differences across users and provide users with information to support mental model alignment.

In my thesis, I first study the nature of collaboration in simulated, large-scale multi-agent systems. Specifically, I explored techniques that utilize context-based communication among decentralized robots in partially observable settings and found that utilizing targeted communication (chapter 4) and accounting for teammate heterogeneity (chapter 5) is beneficial in generating effective coordination policies [6, 7]. Next, I transition to human-machine systems and develop a data-efficient, person-specific, and interpretable tree-based apprenticeship learning framework (chapter 6) to enable cobots to infer and understand decision-making behavior across heterogeneous users [8, 9]. Building on this foundation, I extend neural tree-based architectures to support learning interpretable control policies for robots via reinforcement learning [10]. This advancement not only allows end-users to inspect learned behavior models but also provides developers with the means to verify control policies for safety guarantees (chapter 7). Subsequently, I characterize the utility of Explainable AI (xAI) techniques, which offer the promise of enhancing team situational awareness and shared mental model development

[11] in human-machine teaming (chapter 8). Lastly, I enable end-users to interactively modify interpretable learned policies via a graphical user interface to support team development within a repeated human-machine collaboration paradigm (chapter 9).

The contributions of this thesis are as follows:

- **Creation of a novel communication-based multi-agent reinforcement learning (MARL) architecture**: I develop Multi-agent Graph Attention Communication (MAGIC) [6], a MARL architecture that utilizes targeted communication (agents actively determine "when" and "whom" with to communicate) in learning high-performance team coordination strategies among decentralized agents within partially observable settings. Team members develop an implicit shared mental model via information sharing and simulated experience with collaborators.

- **Creation of a MARL architecture to support heterogeneous robot teams**: I develop Heterogeneous Policy Networks (HetNet) [7], a MARL architecture that effectively models heterogeneous robot teams (i.e., composed of agents with different state, action, and observation spaces). Through HetNet, we facilitate communication across agents, utilizing a differentiable encoder-decoder channel to account for the heterogeneity of inter-class messages, "translating" the encoded messages into a shared, intermediate language among agents of a heterogeneous robot team.

- **Development of an interpretable, person-specific Learning from Heterogeneous Demonstration (LfHD) framework**: I propose a personalized and interpretable apprenticeship scheduling algorithm that infers an interpretable representation of all human task demonstrators by extracting decision-making criteria via an inferred, personalized embedding non-parametric in the num-

ber of demonstrator types [8, 9]. Through this technique, cobots can autonomously gain a personalized, implicit understanding of their human teammate's decision-making behavior, allowing for greater personalization in robotic counterparts.

- **Development of a tree-based model that can be optimized via modern, gradient-based, reinforcement learning approaches to produce high-performing, interpretable policies**: I introduce Interpretable Continuous Control Trees (ICCTs), an interpretable reinforcement learning architecture that allows for direct optimization in a sparse decision-tree-like representation [10]. Our novel architecture is a strong step forward in producing safe and verifiable machine-learning-based autonomous systems that are ready for real-world deployment and interaction with humans.

- **Characterization of the utility of Explainable AI (xAI) in human-machine teaming**: I conduct two novel human-subject experiments quantifying the benefits of deploying xAI techniques within a human-machine teaming scenario. I assessed the ability for human teammates to gain improved situational awareness through the augmentation of xAI techniques and quantified the subjective and objective impact of xAI-supported SA on human-machine team fluency [11]. Importantly, these findings emphasize the importance of developing the "right" xAI models for human-machine collaboration and the optimization methods to support learning these xAI models.

- **Identify a gap in the quality of collaborative agents produced via learning-based techniques and explore xAI-based techniques as a potential solution to improving human-machine collaboration performance.** We display that state-of-the-art collaborative agents within the field of human-machine teaming are rigid and focus on enhancing individualized contribution of the

machine agent rather than effective collaboration across the human-machine team. To absolve the gap in performance between individualized coordination and successful human-machine collaboration, we explore utilizing interpretable models alongside a Graphical User Interface that allows end-users to interact with interpretable robot policies trained via reinforcement learning. This GUI allows end-users to "go under-the-hood" of machine learning models and tune affordances or interactively and iteratively reprogram behavior. Importantly, we find evidence that users teaming with white-box agents supported by interactive modification can outperform teaming with white-box agents alone.

CHAPTER 1
INTRODUCTION

Robotics research has made incredible strides in recent years, providing benefits across a wide variety of applications, such as manufacturing [12], search-and-rescue [13], and healthcare. The creation of these intelligent machines in recent years has enabled the possibility of human-robot collaboration, moving towards the promise of combining the high-accuracy sensors and large computational capability of a machine with the dexterity and creativity of humans. The vision for robots to augment humans as collaborators has long been desired, tracing back to science fiction decades ago, with robots such as R2-D2 or C-3PO working with Luke in Star Wars or Rosey helping her family in the Jetsons. Human-Robot collaboration can broadly be defined as any interaction where a human and robot must collaborate to effectively achieve shared and/or individual objectives. Researchers in this field are concerned understanding, designing, and evaluating robotic systems that can collaborate well with humans [14]. This can range from proximate human-robot collaboration [15], where we have robots and humans working in tandem to assemble objects, to autonomous vehicles coordinating with human drivers [16] so that all vehicles can *safely* reach their destination. This collaboration will be crucial in increasing efficiency in production lines [17], reducing workload for healthcare professionals by creating healthcare robot aides, and saving lives through rapid and coordinated disaster response.

While collaborative robots (i.e., "cobots") have appeared in real-world manufacturing [18, 19, 20], healthcare [21, 22], search-and-rescue [5], and military [23] applications in the past, the "collaboration" has been highly predefined and constrained (e.g., robots will stop or slow down when humans are in the vicinity),

limiting the impact of such technologies. Effective collaboration has been very significant in human history, allowing humans to build at incredible speed and scale, and ultimately spearheading technological development and cultural growth. In this thesis, we present and address several challenges to create effective collaboration between humans and robots.

Human-robot collaboration can be conceptualized as a multi-agent system where multiple agents must work together to achieve objectives. In this system, there are several sources of partial observability that limit effective collaboration. These can include sensory limitations, where a human is limited to their biological senses, and a robot is limited to sensors that it is equipped with (where different sensors may have associated strengths and weaknesses), and the inability to understand a collaborator's intent and/or world model (both a human's mental model and robot's programming are black-box). In these cases, communication is essential for successful coordination and resolving partial observability [24, 25]. Furthermore, high-performing human-human teams exhibit *targeted* communication, where human experts judiciously choose when to communicate and whom to communicate with, communicating only when beneficial [26, 27, 28]. However, as machines receive abundant information, it is unclear how these agents can determine what to communicate, when to communicate, and whom to communicate with. In other words, the process of generating a cohesive message that will benefit the message receiver can be challenging. In the past, researchers have designed hand-designed communication protocols, which are time-consuming, not scalable, and leave much to be desired. In chapter 4, we propose a novel algorithm, Multi-Agent Graph-attentIon Communication (MAGIC), with a graph-attention communication protocol in which we autonomously learn 1) a Scheduler to help with the problems of when to communicate and whom to address messages to, and 2) a Message Processor using Graph Attention Networks (GATs) with dynamic graphs

to aggregate communication signals.

While targeted communication is a step in the right direction, the ubiquity of robotics will depend upon robots being able to team with and understand a *diverse* set of users. This requires not only targeted, contextual communication messages but also *stylized* communication. In Human-Human teams, typical communication patterns widely differ based on the task or role the human assumes [29]. Similarly, robots must be able to maintain effective modeling frameworks over heterogeneous team compositions to support the generation of stylized messages. As such, in chapter 5, we propose Heterogeneous Policy Networks (HetNet) to learn efficient and diverse communication models for coordinating cooperative heterogeneous teams, utilizing a heterogeneous graph-attention architecture that is able to support learning specialized sender-receiver-specific communication channels.

Active communication is beneficial in resolving partial observability, but can be cumbersome for agents. Agents can also passively observe their collaborator's behavior and develop a model of their teammate, generating an understanding over their teammate's behavior. This ability to decipher another person's mental state is known as the Theory of Mind (ToM) capability [30]. Augmenting machines with a model of human behavior has been shown to be beneficial and positively correlate with team performance [4, 31], ultimately allowing agents to generate longer-term collaboration plans. However, generating a model poses a significant challenge due to the complex nature of human behaviors. Due to the growth of Internet of Things (IoT) and the ability to effectively model data, data-driven techniques have become an effective paradigm for modeling human behavior [32]. However, often, a one-size-fits-all approach is used to model all human behavior, lacking the ability to capture person-specific tendencies. Different users may have certain eccentricities or person-specific qualities represented within their data, and this information must be understood by a robot attempting to effectively collaborate

with a specific user. For example, while attempting to infer a decision-making model of users in the face of unlabeled heterogeneity, we found that when trying to infer expert perfusionists' decision-making models during critical intraoperative situations, capturing interoperator disagreement (i.e., heterogeneity) presented in the curated dataset can lead to an improvement of $\approx$ 10%, a substantial gain in performance while predicting decision-making in the operating room [33]. As such, in chapter 6, we create personalized models of user behavior directly from a dataset of heterogeneous users. Our technique enables robots to gain an understanding of their human teammate's decision-making behavior via an inferred, *person-specific embedding*, non-parametric in the number of demonstrator types.

While in chapters 4 and 5, we enable robots to facilitate successful communication strategies emulating human-human teams, the communication modality utilized (real-valued vectors of information) may be unclear for humans if not correlated to something semantically meaningful. Furthermore, the approaches presented utilize black-box models (e.g., deep neural networks), which can be difficult to interpret, and thus limits their ability to be deployed in safety-critical and legally-regulated domains with humans [34, 35, 36, 37]. An alternative approach that facilitates humans to gain insight into a robot's policy is to utilize white-box approaches, as opposed to typical black-box models, to model decision processes in an *interpretable* human-readable representation. In a field such as autonomous driving, such models would provide insurance companies, law enforcement, developers, and passengers with insight into how an autonomous vehicle (AV) reasons about state features and makes decisions. Furthermore, interpretable policies can provide a human teammate insight into the AI's rationale, strengths and weaknesses, and expected behavior as well as help in assessing a system's flaws, verifying its correctness, and promoting its trustworthiness. However, prior interpretable models [38] utilized in representing a robot's behavioral policy

are not amenable to gradient-based learning techniques nor suitable for representing continuous control policies required in robotics. To address these drawbacks, in chapter 7, we design an interpretable yet differentiable tree-based framework for continuous control, allowing for direct synthesization of robot behavior within an interpretable representation. Specifically, we utilize a minimalistic tree-based architecture augmented with low-fidelity linear controllers, creating a novel interpretable reinforcement learning architecture, Interpretable Continuous Control Trees (ICCTs).

During human-robot collaboration, a human teammate can maintain situational awareness and effectively make decisions through the maintenance of an internal mental model of the robot's behavior. However, to develop such an understanding of the robot, the human may have to constantly observe or monitor the robot's behavior, a costly and tedious process. Utilizing transparent representations, such as decision trees, can provide *global* explanations of a decision-making policy that are valid throughout the input space [39]. As such, these models have the potential to provide users with a long-term understanding of a robot's decision-making behavior. This understanding can help facilitate team situational awareness and shared mental model development, ultimately improving the human teammate's ability to predict future behavior of the robot and develop a collaboration plan. Thus, in chapter 8, I assess the ability for human teammates to gain improved situational awareness (SA) through the augmentation of Explainable AI techniques (i.e., utilizing transparent policy visualizations) and quantified the subjective and objective impact of xAI-supported SA on human-machine team fluency. Importantly, this work assesses the preoccupation cost of online explanations in human-machine teaming.

Lastly, teamwork between humans and robots will not complete within a single moment but rather develop over time [40, 41, 42]. A challenge is that once robot

policies have been created, these machines lack the ability to effectively learn with and adapt to human teammates in real-time [43]. In ad hoc human-human teams (i.e., those without prior training), effective teaming is often developed through an iterative process, going through several stages before the team arrives at a fluent collaboration [44]. Similarly, we require robots that can dynamically learn new concepts and adapt learned behaviors to accomplish objectives and collaborate with a human that may exhibit changes. In chapter 9, we build towards adaptive, effective human-robot collaboration by creating a pathway of bi-directional communication, utilizing interpretable policy representations as a mechanism to allow users to understand their machine teammates and allowing for explicit teammate policy modification through an interface (users can modify the machine collaborator's policy via a Graphical User Interface).

1.1 Thesis Statement

In my thesis, I present six different works that push the frontier of real-world robotics systems towards those that understand human behavior, maintain interpretability, and can communicate efficiently and coordinate with high-performance. The central claim underlying my works is:

Thesis Statement: *By providing robots with effective communication abilities, we can produce higher-quality human-robot collaboration.*

The key challenges that are addressed in building up to this thesis statement are:

1. Building decentralized multi-agent coordination protocols that can support efficient, targeted communication for heterogeneous teams.

2. Creating neural tree-based architectures that can represent a robot's policy or

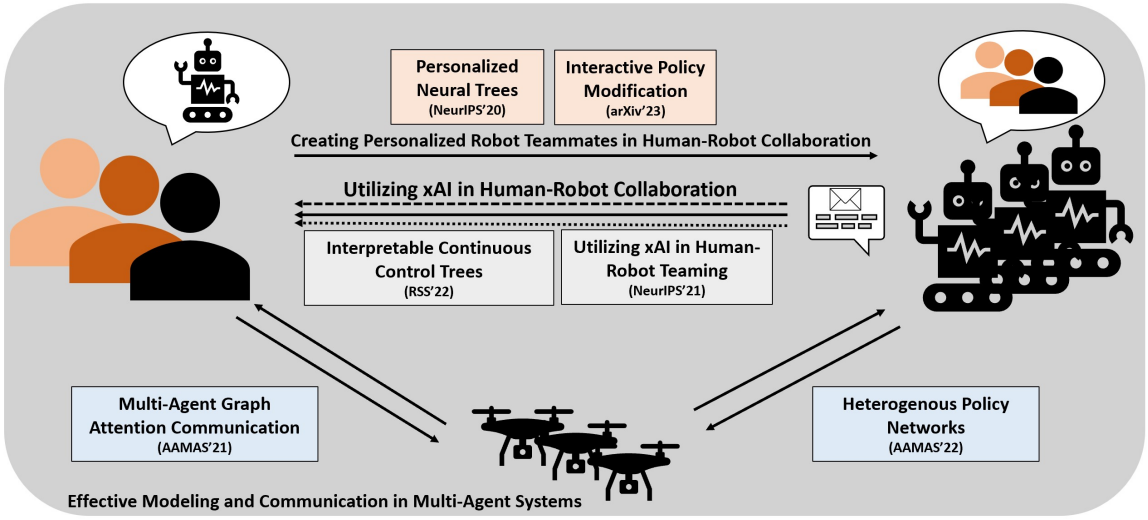


Figure 1.1: This figure shows an overview of my thesis. In chapter 4 and chapter 5, I utilize graph-based architectures to effectively model and facilitate communication in multi-agent systems. In chapter 6 and chapter 9, I allow for greater personalization in robotic counterparts. In chapter 7 and chapter 8, we facilitate directional communication between robots and humans through the use of Explainable AI techniques. These components together help to facilitate the development of shared mental models within a team and result in high-quality human-robot collaboration.

model of a human teammate's behavior that afford interpretability, allowing human teammates direct insight into the model.

3. Evaluating how communication afforded via policy interpretability can benefit a human teammate.

Below, I present an abbreviated discussion over the contributions and key findings of my thesis. I display a figure that ties each of the chapters together in Figure 1.1

1.2 The Importance of Communication in Multi-Agent Systems

In this first chapter, we study the nature of collaboration by developing a multi-agent reinforcement learning framework that focuses on the importance of *targeted communication*, building toward enabling robots to be able to selectively and effi-

ciently share information with humans. Following effective human-human teams, where experts judiciously choose when to communicate and whom to communicate with, communicating only when beneficial [26, 27, 28], we propose Multi-Agent Graph-attentIon Communication (MAGIC), a novel graph communication protocol that determines "when" and "whom" with to communicate via an end-to-end framework. We set a new state-of-the-art in communication-based multi-agent reinforcement learning (MARL) by modeling the topology of interactions among agents (the local and global characterization of connections between agents [45]) as a dynamic directed graph that accommodates time-varying communication needs and accurately captures the relations between agents. Our proposed framework emulates the features of an effective human-human team through its key components, 1) the Scheduler, which helps each agent to decide when it should communicate and whom it should communicate with, and 2) the Message Processor, which integrates and processes received messages in preparation for decision making. We evaluate our method and baselines in several environments, including Predator-Prey, Traffic Junction, and the more complex Google Research Football, achieving state-of-the-art performance.

1.2.1 Goal

Effectively coordinating decentralized agents under partial observability is a challenging computational problem [46, 47, 48, 49, 50]. Recent work has been able to generate high-performance solutions through multi-agent reinforcement learning (MARL), utilizing simulated experience with teammates as a mechanism to learn coordination strategies. Our aim is to augment these techniques in cases where agents are able to communicate and exchange messages, specifically focusing on creating targeted, sparse communication. The ability to communicate and process information can allow for much-increased performance.

1.2.2 Approach

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

At each time step, the observation for each agent is encoded utilizing several different processing modules (LSTM, Fully-Connected layers) to produce a message. Encoded agent messages are aggregated and passed into the Scheduler, which consists of a Graph Attention Network (GAT) encoder and a hard attention mechanism that uses a multi-layer perceptron (MLP) and a Gumbel Softmax function. The Scheduler will output an adjacency matrix, which is a directed graph that indicates the targeted receivers for each agent at each time step. This adjacency matrix is then passed and utilized by the Message Processor to produce a set of integrated messages for each agent at the current timestep. The Message Processor includes a GAT layer receiving messages from all agents, effectively utilizing attention to weight information from agents. The integrated messages for each agent are then incorporated into each agent's policy.

As this system supports end-to-end training, we employ a multi-threaded synchronous multi-agent policy gradient to train our model to generate multi-agent coordination policies with targeted communication.

1.2.3 Results

We evaluate the performance of our proposed method on three environments, including Predator-Prey [51], Traffic Junction [52], and Google Research Football (GRF) [53]. Each domain represents a different challenging multi-agent coordination problem, including search-and-rescue (discover a prey as quickly as possible),

navigation (travel through a noisy intersection without collision), and sports (coordinating against an adversarial team). We note that the last domain, GRF presents a challenging, high-dimensional, mixed cooperative-competitive, multi-agent scenario with high stochasticity and sparse rewards. Further, we benchmark our approach against a variety of state-of-the-art communication-based MARL baselines, including CommNet [52], IC3Net [51], GA-Comm [54], and TarMAC-IC3Net [55].

In Predator-Prey, we see that in both the five- and ten-agent coordination problems, our method converges faster and can achieve better performance than the baselines. Our method converges 52% faster than the next-quickest baseline while still achieving the highest performance. Further, while approaches such as GA-Comm and TarMAC learn competitive policies for the five-agent case, these benchmarks perform much worse than our algorithm in the ten-agent case, suggesting a lack of scalability. In Traffic Junction, our algorithm achieves near-perfect performance after convergence, widely outperforming all benchmarks in its ability to coordinate vehicles without collision. Finally, in GRF, our method converges to a higher-performing policy than all the baselines, achieving the highest scoring success rate in a 2v2 soccer scenario. Across multiple test domains, we set a new state-of-the-art in MARL performance, outperforming baselines, including [52, 51, 54, 55]. Across our domains, we achieve an average 5.8% improvement in steps taken (Predator-Prey), 1.9% improvement in success rate (Traffic Junction), 10.5% improvement in success rate (GRF), and 13.8% improvement in steps taken compared to the closest benchmark which varies across each domain. Further, we conduct an ablation to understand the benefit of targeted communication (i.e., utilizing the Scheduler), and find that the Scheduler provides a performance improvement. This last result signifies the importance of determining "when" and "whom" to communicate with.

Through MAGIC, we are able to develop effective coordination policies for

agents under partial observability, utilizing targeted, sparse communication to facilitate high-performance. While this chapter focuses on agent-agent communication, the ideologies are critical in the development of this thesis and progression toward effective human-robot collaboration.

1.3 Accounting for Heterogeneity in Multi-Agent Systems

Next, we develop a multi-agent reinforcement learning framework that focuses on capturing *heterogeneity* within a multi-agent system. Heterogeneity often appears in multi-agent systems and can be caused due to the diversity in capabilities across agents (including sensing and actuating). As humans can be incredibly diverse, a robot within a human-machine team must be able to account for such heterogeneity in the way that it coordinates and communicates. A simplified example motivating the importance of capturing heterogeneity would be of a robot collaborating with multiple humans, some with poor vision (e.g., has astigmatism) and others with better vision. The robot should be able to prioritize communication across teammates with better sensing capability, even though both senders are humans. Following human-human teaming literature, where communication patterns stylistically differ based on the task or role the human assumes, we develop Heterogeneous Policy Networks (HetNet), a novel, end-to-end heterogeneous graph-attention architecture for MARL that facilitates learning efficient, heterogeneous communication protocols among robot teams. Within our model, we design a differentiable encoder-decoder communication channel to facilitate learning efficient binary representations of states as an intermediate language among agents of different types to improve their cooperativity. We evaluate our method and baselines in several environments, setting a new SOTA in learning emergent cooperative policies by achieving at least an 8.1% to 434.7% performance improvement over baselines and across domains.

1.3.1 Goal

Heterogeneity in multi-agent systems can appear due to diverse robots (i.e., robots with different sensor and/or actuator capabilities) or humans. Allowing and supporting such heterogeneity across teammates can facilitate higher-performance coordination, allow for lightweight robots specialized to certain tasks, and allow for adaptability to a wider variety of situations. However, while MARL researchers have increasingly focused on developing computational models of team coordination and communication, these prior frameworks fail to explicitly model the heterogeneity within teams. Our aim is to enable MARL frameworks to effectively model heterogeneous teams and support stylized communication (i.e., by allowing for selective attention while communicating with certain types of agents).

1.3.2 Approach

Emulating high-performing teams that implicitly understand the different roles of heterogeneous team members and adapt their communication protocols accordingly, we formulate a MARL framework that supports heterogeneous teams with different state, action, and observation spaces. Specifically, we propose Heterogeneous Policy Networks (HetNet) to learn efficient and diverse communication models for coordinating cooperative heterogeneous teams.

In HetNet, we cast the cooperative MARL problem into a heterogeneous graph structure, and propose a novel heterogeneous graph-attention network capable of learning diverse communication strategies based on agent classes. Here, agents of different classes are defined as those with different action and observation spaces. We directly model each agent class with a unique node type. This allows agents to have different types of state-space content as input features according to their classes, as well as enabling different types of action spaces. Communication channels between agents are modeled as directed edges connecting the corresponding

agent nodes. When two agents move to a close proximity of each other, such that those agents fall within communication range, we add bidirectional edges to allow message passing between them. We use different edge types to model different class combinations of the sender and receiver agents to allow for learning heterogeneous communication protocols and intermediate representations.

At each timestep, our multi-agent heterogeneous attentional communication architecture generates agent-specific messages through a class-specific feature preprocessor. These messages are passed into a HetGAT communication channel that includes a class-specific encoder-decoder network, allowing heterogeneous agents to decode information in different ways. While decoding, the HetGAT layer utilizes attention for message aggregation, weighing received messages from different classes of agents. Upon completion, agents utilize the resultant embedding (representing communicated information as well as the agent's own observation) to determine an action.

To support training HetNet, we develop a modified Multi-Agent Heterogeneous Actor-Critic (MAHAC) framework for learning class-wise coordination policies.

1.3.3 Results

We evaluate the utility of HetNet against several baselines in three cooperative MARL domains that require learning collaborative behaviors. The first is a homogeneous-robot search-and-rescue domain, Predator-Prey. The second domain is a heterogeneous variant of Predator-Prey, Predator-Capture-Prey, where we create two classes of agents, *predator* and *capture* agents. Agents of the *predator* class have the goal of finding the prey with limited vision (similar to agents in PP). Agents of the *capture* class, have the goal of locating the prey *and* capturing it with an additional *capture-prey* action in their action-space, while not having any observation inputs (e.g., lack of scanning sensors). The third domain is a heteroge-

neous domain, FireCommander, that explores two classes of *perception* and *action* agents must collaborate as a composite team to extinguish a propagating firespot. Similar to the previous domain, the latter class of agents, *action* agents, do not have access to any observation input but have increased capability (e.g., put out fires). This challenging domain includes a realistic fire propagation module with high stochasticity. We benchmark against four end-to-end communicative MARL baselines: (1) CommNet [52], (2) IC3Net [51], (3) TarMAC [55] and, (4) MAGIC [6].

We find that through modeling heterogeneity, HetNet outperforms all baselines in all three domains, further outperforming baselines in domains with increasing complexity and heterogeneity. Specifically, in our most challenging domain, we see we are able to outperform baselines by an average of 434.7%. Further, we conduct ablations and show HetNet is robust to varying team compositions, setting a new state-of-the-art in learning emergent cooperative policies for heterogeneous robot teams.

Through HetNet, we see the importance of effective modeling of heterogeneity within a multi-agent system. Accounting for heterogeneity is essential within collaboration, and we bring this value into human-robot collaboration within the next chapter.

1.4 Inferring Behavioral Policies of Heterogeneous Human Decision-Makers

While supporting heterogeneous communication can facilitate understanding during collaboration, robots can also utilize a *model of the human teammate's behavior* to inform the generation of a collaboration plan. Prior works in human-robot collaboration have shown that maintaining and utilizing such models positively correlate with team performance [4, 31]. We specifically focus on a subset of approaches that utilize data-based techniques for modeling human behavior [32], directly learning a supervised policy mapping states to actions. However, when varying human

experts address complex problems (e.g., resource coordination), they utilize heterogeneous rules-of-thumb and strategies honed over decades of apprenticeship, creating unique heuristics depending on experts' varied experiences and personal preferences [56, 57]. However, such heterogeneity is not readily handled by traditional apprenticeship learning approaches that assume demonstrator homogeneity. A canonical example of this limitation is of human drivers teaching an autonomous car to pass a slower-moving car, where some drivers prefer to pass on the left and others on the right. Apprenticeship learning approaches assuming homogeneous demonstrations either fit the mean (i.e., driving straight into the car ahead of you) or fit a single mode (i.e., only pass to the left), producing a poor representation of a human's decision-making behavior. In chapter 6, we formulate a personalized and interpretable apprenticeship scheduling framework for heterogeneous Learning from Demonstration (LfD) that outperforms prior state-of-the-art approaches on both synthetic and real-world data across several domains through the use of personalized embeddings. The key to our approach is the utilization of a variational inference mechanism to maximize the mutual information between the personalized embedding and the modeled decision-maker, allowing us to simultaneously infer preferences in the face of unlabeled heterogeneity as well as decision-maker policies. I display a comparison between our approach and those previous in Figure 1.2.

1.4.1 Goal

Accurate mental models of teammates play a crucial role in effective collaboration. Human experts often employ diverse decision-making strategies based on their individual qualities and preferences. Therefore, accounting for personalization becomes essential in human-robot collaboration to avoid erroneous inferences by robots, which could result in poor teaming performance. Our aim is to infer

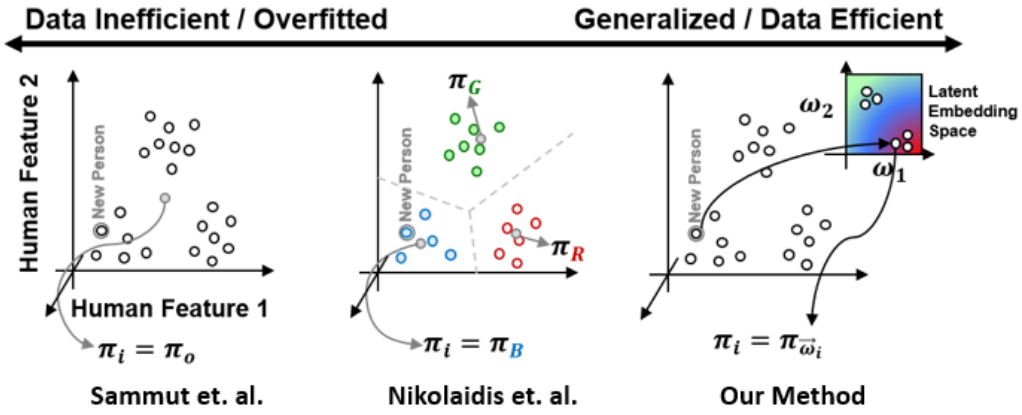


Figure 1.2: The use of personalized embeddings to help to capture the homo- and heterogeneity across human demonstrations.

accurate personalized decision-making models of user behavior directly from a dataset of unlabeled heterogeneous users and create a mechanism to support fast, online inference of personal characteristics to support the prediction of future behavior.

1.4.2 Approach

We propose a personalized and interpretable apprenticeship scheduling algorithm that infers an interpretable representation of all human task demonstrators by extracting decision-making criteria via an inferred, personalized embedding non-parametric in the number of demonstrator types. To create a model that is both interpretable and can effectively capture the heterogeneity across demonstrators, I proposed Personalized Neural Trees (PNTs) [9], by extending a tree-like neural network architecture termed Differentiable Decision Trees. A key component of my architecture is the addition of personalized embeddings as a latent variable, representing latent patterns of thinking for decision-makers. At each timestep, the current state and a decision-maker's personalized embedding are passed into the PNT to predict that decision-maker's action. As both the decision-making policy and latent space must be inferred simultaneously, we introduce an information-theoretic

regularization model and utilize a variational inference mechanism to maximize the mutual information between the embedding and the modeled decision-maker (i.e., to incentivize mode discovery within the latent space). The combination of this training mechanism with our novel architecture allows us to learn a general behavior model accompanied by personalized embeddings that fit distinct behavior modalities. To increase data-efficiency in an offline-training paradigm, we enforce the use of counterfactual reasoning that leverages person-specific embeddings as pointwise terms. Finally, our architecture supports a post-hoc transformation procedure to translate the differentiable tree model into an interpretable decision tree. Such an interpretable model of human behavior in resource allocation or planning tasks would be useful for a variety of reasons, from decision explanations to training purposes.

1.4.3 Results

We utilize resource scheduling and coordination as a testbed for evaluating our approach. We utilize three environments. The first is a synthetic low-dimensional environment where an expert will choose an action based on the state and one of two hidden heuristics. The second is a jobshop scheduling environment built on the **XD[ST-SR-TA]** scheduling domain defined by [58], representing one of the hardest scheduling problems, and where two agents must work together to complete a set of 20 tasks that have upper- and lower-bound temporal constraints (i.e., deadline and wait constraints), proximity constraints, and travel-time constraints. The last domain utilizes real-world data from users routing Taxis in a variant of the Taxi Domain in [59]. We benchmark our approach against a variety of baselines [60, 61, 62, 63, 64, 65].

In the low-dimensional environment, our method for learning a continuous, personalized embedding sets the state-of-the-art ($95.30\% \pm 0.3\%$ in human decision

prediction) for solving this latent-variable classification problem. Furthermore, we are able to infer the true number of latent preferences using our mutual information maximization formulation. In the jobshop scheduling domain, our personalized apprenticeship learning framework widely outperforms all other approaches, achieving 96.13% ± 2.3% accuracy in predicting demonstrator actions. Lastly, with real-world data, our personalized apprenticeship learning framework outperforms all other benchmarks and is the only method to achieve a prediction accuracy over 80%.

With this new architecture, machines can better learn from heterogeneous users and detect person-specific behaviors, allowing for greater personalization in robotic counterparts.

1.5 Generating Interpretable Robot Policies

While robots maintaining a mental model over human behavior is beneficial, it is also important for humans to develop a model over a robot's behavior, shifting towards a "mutual understanding." One approach for providing humans with an understanding of a robot's policy is to utilize human-readable, *interpretable policies*. Importantly, interpretable policies provide a human teammate insight into the AI's rationale, strengths and weaknesses, and expected behavior, and offer the promise of enhancing team situational awareness, shared mental model development, and human-robot teaming performance. Such insight is necessary for large-scale cobot adoption in safety-critical and legally-regulated domains [34] as these models allow for the ability to assess a system's flaws, verify its correctness, and promote its trustworthiness. However, such models have often underperformed black-box models in many domains, are not amenable to gradient-based learning techniques, nor suitable for continuous control problems in robotics.

In chapter 7, we design an interpretable yet differentiable tree-based frame-

work for continuous control, allowing for direct synthesization of robot behavior via reinforcement learning within an interpretable representation. Specifically, we utilize a minimalistic tree-based architecture augmented with low-fidelity linear controllers, creating a novel interpretable RL architecture, Interpretable Continuous Control Trees (ICCTs). *Our novel architecture and training procedure provide a strong step towards solutions for two grand challenges in interpretableML announced in 2021 [66]: (1) Optimizing sparse logical models such as decision trees and (2) Interpretable Reinforcement Learning*. I provide several extensions to prior differentiable decision tree frameworks within the ICCT architecture to support interpretable reinforcement learning: 1) a differentiable crispification procedure allowing for optimization in a sparse decision-tree like representation, and 2) the addition of sparse linear leaf controllers to increase expressivity while maintaining legibility. We evaluate our model in its ability to learn high-performance control behaviors across several continuous control problems, including four autonomous driving scenarios. We find that ICCTs are able to learn policy representations that parity or outperform baselines by up to 33% in autonomous driving scenarios while achieving a 300x-600x reduction in the number of policy parameters against deep learning baselines.

1.5.1 Goal

Reinforcement learning (RL) with deep function approximators has enabled the generation of high-performance continuous control policies across a variety of complex domains, from robotics and autonomous driving to protein folding and traffic regulation. However, while the performance of these controllers opens up the possibility of real-world adoption, the conventional deep-RL policies used in prior work lack *interpretability*, limiting deployability in safety-critical and legally-regulated domains [34, 35, 36, 37]. We aim to enable interpretable models, specifi-

cally tree-based models, to support gradient-based learning.

1.5.2 Approach

We present a novel tree-based architecture, Interpretable Continuous Control Trees (ICCT), that affords gradient-based optimization with modern reinforcement learning techniques to produce high-performance, interpretable policies for continuous control applications. Specifically, we utilize 1) a differentiable crispification procedure allowing for optimization in a sparse decision-tree-like representation and 2) the addition of sparse linear leaf controllers to increase expressivity while maintaining legibility.

We start with the base differentiable decision tree (DDT) architecture [67]. Similar to a Decision Tree, DDTs contain decision nodes and leaf nodes; however, each decision node within the DDT utilizes a sigmoid activation function (i.e., a "soft" decision), outputting the probability that the node evaluates to TRUE, instead of a Boolean decision (i.e., a "hard" decision). Furthermore, each decision node is not a function of one input variable as in a decision tree and rather is computed by weighting state features through decision-node-specific weights and subtracting a bias term (similar to a linear layer within a multi-layer perceptron).

To create an architecture that supports both an interpretable forward propagation through the model that traces down a single branch of a tree as well as gradient flow to update parameters of the neural tree model, we propose differentiable crispification. This consists of two components: 1) Decision node crispification, which recasts each decision node to split upon a single dimension of our input feature, and 2) Decision outcome crispification, which translates the outcome of a decision node so that the outcome is a Boolean decision rather than a set of probabilities. Both operations utilize the straight-through trick [68] to maintain gradients, allowing the possibility of utilizing gradient-based techniques (e.g., reinforcement

learning) to train the model. At the model leaf nodes, we replace the standard Gaussian distributions typically employed in continuous control domains with a sparse, linear controller. These sparse, linear controllers allow us a tradeoff between tree depth and leaf controller sparsity, allowing engineers to balance model depth, complexity, and performance.

1.5.3 Results

We evaluate the utility of ICCTs across 6 domains, including two common continuous control problems, Inverted Pendulum and Lunar Lander provided by OpenAI Gym [69], and four autonomous driving scenarios: Lane-Keeping provided by [70] and Single-Lane Ring Network, Multi-Lane Ring Network, and Figure-8 Network all provided by the Flow deep reinforcement learning framework for mixed autonomy traffic scenarios [71]. We compare against several baselines, including black-box methods (traditional DDTs and neural networks) alongside several interpretable models (Decision Trees and post-hoc translations of DDTs).

Comparing our ICCTs to black-box models, we see that in all domains, we parity or outperform deep highly-parameterized models in performance while reducing the number of parameters required by orders of magnitude. In the difficult Multi-Lane Ring scenario, we see that we can outperform MLPs by 33% on average while achieving a 300x-600x reduction in the number of policy parameters required. Comparing our ICCTs to white-box models, we see that across four of the six control domains, our ICCT is able to widely outperform both the DT and discretized continuous DDT model. In complex autonomous driving domains such as Lane Keeping, we are able to widely outperform prior work and outperform DT $w\backslash$ DAGGER by 42.11% while maintaining fewer parameters.

Overall, we find extremely positive support for our Interpretable Continuous Control Trees, displaying the ability to at least parity black-box approaches while

maintaining high interpretability. Our novel architecture is a strong step forward in producing safe and verifiable machine-learning-based autonomous systems that are ready for real-world deployment and interaction with humans.

1.6 The Utility of Explainable AI in Human-Robot Collaboration

Recent advances in machine learning have led to growing interest in xAI to enable humans to gain insight into the decision-making of machine learning models. Despite this, the utility of xAI techniques has not yet been characterized in human-machine teaming. Importantly, xAI offers the promise of enhancing team situational awareness (SA) and shared mental model development, which are the key characteristics of effective human-machine teams. Rapidly developing such mental models is especially critical in ad hoc human-machine teaming, where agents do not have a priori knowledge of others' decision-making strategies.

In chapter 8, we present two novel human-subject experiments quantifying the benefits of deploying xAI techniques within a human-machine teaming scenario. We assess the ability for human teammates to gain improved situational awareness through the augmentation of xAI techniques and quantify the subjective and objective impact of xAI-supported SA on human-machine team fluency.

1.6.1 Approach

First, We design a complex HMT scenario within the Malmo Minecraft AI Project [72], where a human and AI must work together to build a multi-level house. During the interaction, we utilize one of the following explanations for the robot policy: a decision tree (describing a transparent representation of the robot's behavioral policy), a text-based explanation (describing the robot's current intent), or no explanation. The robot policy is "human-aware" in that there is a tree-based policy that infers the human's current action (in the same sense as the PNT) and a second

tree-based behavioral policy that determines a complementary action conditioned on the human teammate's current behavior (in the same sense as the ICCT).

We conduct and design two human-subject studies; first, we investigate how different abstractions of the robot's policy can influence a human's situational awareness (SA), i.e., by helping a human perceive the current environment (Level 1), comprehend the AI's decision-making model (Level 2), and project into the future to develop a collaboration plan (Level 3). This study is offline (i.e., the user is not actively collaborating with the robot but viewing gameplay from a third-person point of view). We utilize the Situational Awareness Global Assessment Technique in gauging the different levels. Second, we conduct a study on ad hoc HMT investigating how *online* xAI-based support, generated via robot policy abstractions, and the human teammate's ability to process higher levels of information affect teaming performance. This study is online, where users must collaborate with an unfamiliar robot in an ad hoc setting while paying attention to explanations that may help users develop a collaboration plan. Importantly (and uniquely), this study assesses the preoccupation cost of online explanations in human-machine teaming.

1.6.2 Results

First, we find that using interpretable models that can support information sharing with humans lead to increased SA ($p < 0.05$). Specifically, full transparency (i.e., providing the user a decision-tree representation of the teammate's behavior) performs best in increasing the user's SA, allowing them to better comprehend the robot's decision-making and predict into the future. Second, we examine how different SA levels induced via a collaborative AI policy abstraction affect ad hoc human-machine teaming performance. Importantly, we find that the benefits of xAI are not universal, as there is a strong dependence on the composition of the

human-machine team. Novices benefit from xAI providing increased SA ($p < 0.05$) but are susceptible to cognitive overhead ($p < 0.05$). On the other hand, expert performance degrades with the addition of xAI-based support ($p < 0.05$), indicating that the cost of paying attention to the xAI outweighs the benefits obtained from being provided additional information to enhance SA.

Our results demonstrate that researchers must deliberately design and deploy the right xAI techniques in the right scenario by carefully considering human-machine team composition and how the xAI method augments SA.

1.7 Reducing Rigidity in Human-Robot Collaboration

Interpretability is effective in providing users with awareness over teammate behavior, exhibiting a sense of directional communication from the robot to the human. However, this falls short of bidirectional communication, where a human would also be able to "tell" a machine online to perform a desired collaborative behavior. While many techniques can generate collaborative robot policies, the result may not actually be helpful or what the human wants. In these scenarios, bidirectional communication is essential. *Humans, when teaming with machines, should be able to intuitively update what the robot has learned or change it based upon preferences that may evolve over time.*

In chapter 9, we first characterize prior work in HMT [73, 74], finding that machine behavior is unable to adapt to human-preferred strategies, and that high performance is typically driven by independent machine actions rather than collaboration, which can ultimately result in a higher team score. The inability to adapt inhibits *team development*. Team development often consists of several stages [44], including "Forming", "Storming", "Norming" and "Performing." The Forming stage is associated with a drop in performance as team members are unfamiliar with each other and how to collaborate. In the Storming stage, team members

continue to understand each other and begin to establish roles and strategies. In the Norming stage, the team performance begins to improve as agents learn to collaborate harmoniously. Finally, in the Performing stage, the team is achieving its full potential, exhibiting the highest level of cooperation and score. Such development is critical in developing coordination strategies [44], and is associated with the calibration of trust, assignment of roles, and development of a shared mental model.

Building on prior chapters, we facilitate a sense of bi-directional communication by creating a Graphical User Interface (GUI) to allow users to modify an interpretable AI teammate's behavior to their specifications. This capability is incredibly promising, enabling end-users to "go under-the-hood" of interpretable machine learning models and tune affordances or interactively and iteratively reprogram behavior.

1.7.1 Approach

This work contains three key contributions to improve the field of human-machine teaming. First, we identify a gap in human-machine teaming approaches by evaluating published agents. We find that while recent approaches have improved the quality of human-machine teaming (HMT), the learned strategies are rigid and focus on enhancing individualized contribution of the machine agent rather than effective collaboration across the human-machine team. As these behaviors are rigid, humans are unable to explore strategies that may improve performance beyond the AI's training. Next, we create an interpretable machine learning architecture, Interpretable Discrete Control Trees (IDCTs), that can be used directly with reinforcement learning to produce interpretable teammate policies. While this architecture is a straightforward adaptation of the previously presented ICCTs to discrete action-spaces, we provide a contextual pruning algorithm to improve

ease-of-training and interpretability of neural-tree based models. Furthermore, we build towards adaptive, effective HMT by creating a pathway of bi-directional communication, utilizing interpretable policy representations as a mechanism to allow users to understand their machine teammates and allowing for explicit teammate policy modification through an interface (users can modify the machine's policy via a GUI). Finally, we conduct a 50-participant between-subjects user study assessing the effects of interpretability and policy modification across four repeated interactions with an AI. Specifically, we are interested in understanding which mechanisms promote high-performance teaming (i.e., how well can the human and AI team work together?) and teaming development (i.e., how does the performance of teaming change over time, and relate to Tuckman's stages of teaming?).

1.7.2 Results

In understanding the quality of the current state-of-the-art in human-machine teaming, we evaluate publicly available agents in simple domains, exploring whether AIs can support near-optimal human-preferred strategies. We find that agents from [73] are unable to coordinate with humans well under the human-preferred strategy. The AI teammate freezes in place as it likely did not encounter such a teaming strategy in its training, receiving a very low score. Furthermore, in a second domain, we find that trained policies from prior work [74] converge to individualized strategies, even though effective collaborative behavior will result in a higher team score. As this domain is also used within our between-subjects experiment, we see that rigid AIs that converged to individualized behaviors are not able to exhibit much higher team scores than those training.

In our experiment, we find several interesting results. Objectively, we find that white-box models underperformed black-box models from prior work in teaming with humans. This is confounded by black-box models being easier

to train, and thus producing higher-performance agents. Furthermore, we find that 1) users found low-performance black-box models incredibly frustrating, 2) white-box teaming supported with interactive modification outperforms white-box approaches alone, and 3) users that were extroverted, had familiarity with decision trees, and were experienced in gaming were better able to create high-performing personalized AI teammates. Given these findings, in the future, to absolve the gap in performance between individualized coordination and successful human-machine collaboration, researchers must focus on developing better interpretableML approaches to support the generation of high-performance white-box teammates, the modality of communication between agents and humans (tree policies may be difficult to understand for some users), and effective mixed-initiative interfaces that allow users, who may vary in ability and experience, to interact and improve team behavior.

CHAPTER 2
RELATED WORK

In this chapter, I introduce related work across several disciplines, including multi-agent coordination, modeling human behavior, and interpretable policy representations.

2.1 Multi-Agent Coordination

Coordinating multi-agent teams is a challenging computational problem [75, 47, 48, 49, 50, 76]. In multi-agent settings, each agent observes other agents as part of the environment, causing the environment to appear dynamic and non-stationary. Further difficulty arises due to the issue of credit assignment, where it is difficult for each agent to deduce its own contribution to the team's success (especially when there are only global rewards). To solve these multi-agent challenges, many researchers in multi-agent reinforcement learning (MARL) [77] have pursued centralized training and decentralized execution. Further extensions allow agents to exchange messages during execution, allowing for increased performance. Here, we present recent work in MARL, focusing on MARL with a centralized critic, MARL frameworks augmented with agent-agent communication, and MARL frameworks utilizing graph modeling.

MARL with Centralized Critic – Some works extend variants of actor-critic algorithms to multi-agent settings and learn decentralized policy through centralized critics without explicit communication channels [78, 79, 80]. MADDPG [78] is a MARL framework based on Deep Deterministic Policy Gradient, and can be applied in both cooperative and competitive scenarios. COMA [79] extends on-policy actor-critic and proposes a counterfactual baseline to address the credit assignment

problem. MAAC [80] is developed from Soft-Actor-Critic [81] and takes advantage of the idea of the counterfactual baseline from COMA. The authors propose a specialized attention mechanism over agents when training the critic, which allows for better scalability as its input space increases linearly, instead of exponentially, with respect to the number of agents. While these works present critical improvements in the field of MARL, the ability to communicate and process information allows for much-increased performance.

MARL with Communication – Recently, the use of communication (enabling agents to communicate and exchange messages during execution) in MARL has been shown to enhance the collective performance of learning agents in cooperative MARL problems [51, 82, 55, 52, 54, 83, 84, 85]. Differentiable Inter-Agent Learning (DIAL) [86] builds up limited-bandwidth differentiable discrete communication channels among agents. CommNet [52] extends to a continuous communication protocol designed for fully cooperative tasks. Agents receive averaged encoded hidden states from other agents and use the messages to make decisions. IC3Net [51] uses a gating mechanism to enable the agents to decide when to communicate, and thus is amenable to competitive scenarios, utilizing a similar message aggregation scheme. However, both IC3Net and CommNet process messages with a simple average. The proper integration of these messages is critically important for communication. TarMAC [55] achieves targeted communication through an attention mechanism that improves performance compared to prior work. Nevertheless, TarMAC requires high-bandwidth message-passing channels, and its architecture is reported to perform poorly in capturing the topology of interaction [54]. ATOC [87], employs an attention mechanism to decide if an agent should communicate in its observable field. SchedNet [88] proposes a weight-based scheduler to pick agents who should broadcast their messages. However, both ATOC and SchedNet have to manually configure their communication groups. Furthermore, across the

several MARL works discussed in this section, including [52, 55, 51], a key limitation is that these frameworks required an individualized reward scheme to scale to larger task configurations. Individualized reward schemes, while assisting with the credit assignment problem in MARL [79], can learn suboptimal coordination strategies in heterogeneous teaming configurations. A more effective, albeit more difficult-to-learn strategy is to use a shared team reward scheme, enabling agents to learn cohesive policies that achieve the composite team's goal, rather than learning policies that benefit the skill of a particular class of agent.

MARL with Graph Neural Networks (GNN) – Graph Neural Networks (GNNs) are powerful tools for learning from data with graph structures [89, 90]. To model the interactions between agents, MARL has utilized GNNs to allow for a graph-based representation [91, 92, 93, 94]. Deep Graph Network (DGN) [95] represents dynamic multi-agent interaction as a graph convolution to learn cooperative behaviors. This seminal work in MARL demonstrates that a graph-based representation substantially improves performance. In [91], an effective communication topology is proposed by using hierarchical GNNs to propagate messages among groups and agents. G2ANet [54] proposed a game abstraction method combining a hard and a soft-attention mechanism to dynamically learn interactions between agents.

Heterogeneity in Multi-agent Systems – In [96], several types of heterogeneity induced by agents of different capabilities are discussed. As opposed to homogeneous teams, the diversity among agents in heterogeneous teams makes it challenging to hand-design intelligent communication protocols. In [97], a control scheme is hand-designed for a heterogeneous multi-agent system by modeling the interaction as a leader-follower system. More recently, HMAGQ-Net [98] utilized GNNs and Deep Deterministic Q-network (DDQN) to facilitate coordination among heterogeneous agents (i.e., those with different state and action spaces).

2.2 Inferring a Model of Human Behavior

Accurately modeling and understanding human behavior has been a long-standing challenge in the fields of Psychology [99], Behavioral Science, Cognitive Science, and more recently, Human-Robot/Human-Computer Interaction [100, 101]. While scientists have developed heuristic models of human behavior or attempted to manually solicit and encode "tribal knowledge" into a rule-based computational framework [102], these approaches are prone to error and not scalable. As such, we instead utilize data-based techniques seek to enable robots to autonomously extract a human's decision-making model through observation to scale the power of the domain expert and remove the need for a programmer.

Recently, the field of learning from demonstration (LfD) has had substantial success in capturing domain-expert knowledge directly from demonstration [32, 103, 104, 105, 106]. The process of inferring a mental model of human behavior offline (i.e., without the need to simulate a policy) can be accomplished through Behavioral Cloning, an LfD technique [107]. LfD mechanisms are often based on a Markov Decision Process (MDP), defined in Chapter 3. The goal in LfD is to receive both a set of trajectories provided by a human demonstrator $\{\langle s_t, a_t \rangle, \forall t \in \{1, 2, \dots T\}\}$ as well as an MDP$\backslash R$, and then to recover a policy that can predict the correct state-action sequence a human would take in a novel situation.

As humans are diverse and can be influenced by a number of internal and external factors, including trust in robots [108], stress levels [109], physical capability [11], engagement [110], sleep deprivation and caffeine or alcohol intake [111], a one-size-fits-all model representing behavior will be insufficient. It is important to relax the assumption of homogeneous demonstrations by explicitly capturing modes in human behavior across users (i.e., capturing heterogeneity within the data). There has been growing interest in understanding decision-maker hetero-

geneity [61, 64, 62, 63, 8, 112, 113]. Nikolaidis et al. [61] first used an expectation-maximization formulation to cluster decision-maker behavior before applying inverse reinforcement learning (IRL) for each cluster k. Negatively, this approach requires interaction with an environment model, and the IRL algorithm only has access to $\sim 1/k^{th}$ of the data to learn from. More recently, Li et al. [62] presented InfoGAIL, which used mutual information maximization to learn discrete, latent codes; however, InfoGAIL requires access to an environment simulator and a ground-truth reward signal. While InfoGAIL argues its latent codes afford interpretability, but its model structure is still a black-box neural network [114]. Hsiao et al. [64] presented an approach to discover latent factors within demonstrations using a categorical latent variable with limited expressivity. Finally, Tamar et al. [63] used a sampling-based approach to learn the modalities within the data, but the approach requires voluminous data due to the algorithm's high-variance estimation framework.

Capturing Domain-Expert Behavior in Resource Coordination - Researchers have also sought to learn scheduling policies from demonstration [65, 48, 115, 46]. Gombolay et al. [65] consider learning scheduling policies but does not consider heterogeneity. [115] proposed PTIME to learn to schedule calendar appointments; however, this approach requires manually soliciting user ranking data and computationally-intensive nonlinear optimization.

2.3 Interpretable Policy Representations

2.3.1 Explainable AI

Due to recent accidents with autonomous vehicles (cf. [116]), there has been growing interest in developing Explainable AI (xAI) approaches to understand an AV's decision-making and ensure robust and safe operation. xAI is concerned with understanding and interpreting the behavior of AI systems [117]. In recent years, the necessity for human-understandable models has increased greatly for safety-

critical and legally-regulated domains, many of which involve continuous control (e.g., specifying joint torques for a robot arm or the steering angle for an autonomous vehicle) [118, 34]. In such domains, prior work [119, 120, 121, 81] has typically used highly-parameterized deep neural networks in order to learn high-performance policies, completely lacking in model transparency.

Interpretable machine learning approaches refers to a subset of xAI techniques that produce globally transparent policies (i.e., humans can inspect the entire model, as in a decision tree [38, 122, 123] or rule list [124, 125, 35, 126]). In particular, tree-based frameworks could represent complex decision-making processes while maintaining interpretability. Decision trees [38] represent a hierarchical structure where an input decision can be traced to an output via evaluation of decision nodes (i.e., "test" on an attribute) until arrival at a leaf node. Decision nodes within the tree are able to split the problem space into meaningful subspaces, simplifying the problem as the tree gets deeper [127, 128, 129]. *Decision trees provide global explanations of a decision-making policy that are valid throughout the input space [39], as opposed to local explanations typically provided via "post-hoc" explainability techniques [130, 131, 9].* Several approaches have attempted to distill trained neural network models into decision trees [132, 133]. While these approaches produce interpretable models, the resulting model is an approximation of the neural network rather than a true representation of the underlying model. Our work, in chapter 7, instead, directly learns an interpretable tree-based policy via reinforcement learning, producing a model that can be directly verified without utilizing error-prone post-hoc explainability techniques. We emphasize that explainability stands in contrast to interpretability, as explanations may fail to capture the true decision-making process of a model or may apply only to a local region of the decision-space, thereby preventing a human from building a clear or accurate mental model of the entire policy [114, 134, 135, 11].

To clarify our definitions of interpretability and explainability in relation to the models generated via machine learning techniques, we define them below following [114].

1. Interpretability: While a domain-specific notion, we define interpretability as a model that can be directly understood by inspection (i.e., a human can *easily* trace decisions from input to output). By utilizing an interpretable model, users have transparent insight into the underlying factors driving model predictions.

2. Explainability: We define explainability as works that utilize a second post-hoc model to explain a black-box, typically a neural network model. Utilizing a second post-hoc approximation to generate an understanding of a black-box model can lead to unreliable explanations that are not faithful to the true model. As we focus on generating models that work with humans (i.e., high-stakes scenarios), explainable models can result in harmful downstream effects and should be avoided.

Neurosymbolic AI

Interpretable models reason over semantically meaningful features to produce a desired output. While these models can be directly understood by humans and can be produced given a set of pre-processed data and an objective function, these models do not necessarily learn generalizable representations from data. Emulating the following capabilities, 1) making sense of large amounts of raw data and converting this information into succinct representations (commonly referred to as "System 1" within a human's cognitive processes [136]) and 2) reasoning over said representations to perform a desired objective ("System 2"), within a model results in a *neurosymbolic model* [137]. Neurosymbolic models utilize a combination of neural-network-based techniques and symbolic approaches to allow for enhanced

model generalization (i.e., robustness to outliers) alongside the production of an interpretable model that can enhance trust and be inspected for safety guarantees. Within the subfield of AI that aims to create neurosymbolic models, there are numerous methods, ranging from learning signal temporal logic robot control policies directly from raw human demonstrations [138] to multi-stage approaches where abstract representations are first inferred and then utilized in creating a downstream high-level policy [139]. In several of the presented contributions in this thesis, we utilize neural-tree-based models, which can be classified as a neurosymbolic model. These models, further discussed in chapter 6 and chapter 7, have the benefits of neural networks in that they support gradient-based learning, can learn from raw data, and support online learning, as well as symbolic approaches that are able to represent model behavior in the form of a hierarchical tree-based structure and support human readability. Importantly, our proposed neural-tree-based models can support data-efficient learning, infer structured rules relating symbolic representations directly from raw data, and support manipulation from end-users (i.e., human intervention). While our neurosymbolic tree-based model has several benefits that allow it to effectively learn interpretable, high-performance models directly from demonstration (chapter 6) or via reinforcement learning (chapter 7), it is important to note that our model may not be able to incorporate higher-order logical relationships (e.g., second-order logic) as well as other neurosymbolic models.

2.3.2 Human-Machine Teaming

The field of human-machine teaming is concerned with understanding, designing, and evaluating machines for use by or with humans [140]. This growing field has recently attracted much attention from researchers, exploring how to facilitate better collaborative performance between humans and machines [141, 142, 143]. A

common technique that has been used to produce AI agents is Deep Reinforcement Learning, where researchers have concentrated on approaches to reduce the dissimilarity between training data and testing with end-users. Common approaches that have achieved success in the past include utilizing human gameplay data to finetune simulated training partners to behave more human-like [73], which can be costly, and training with a diverse-skilled population of synthetic partners to create an agent that can better generalize to non-expert end-users [74].

xAI in Human-Machine Teaming – Explainable AI in human-machine teaming is a promising direction as automation with the ability to explain will allow users to better understand the behavior of their AI teammates. Prior work attempts to induce transparency in a human-robot team by the explanation of failure modes [144, 145], synthesis of policy descriptions [146, 147], and the verbalization of experiences [148, 149]. While these approaches are successful are instilling a sense of understanding of robot behavior within the human teammate, the level of collaboration is limited, and these works do not assess the preoccupation cost of online explanations in human-machine teaming. More recently, [150, 151, 152, 153, 154], and [155] investigate the type and accuracy of an xAI explanation to a human's trust and reliance on the setting of human-AI teaming. However, while this prior work deploys xAI-based support in classification problems (utilizing the AI as a recommender system), our task and scenario are widely different in that we consider a complex sequential decision-making and planning problem where the AI is a collaborative agent that actively shapes the world. [156] provides insight into how different xAI explanations relate to a human player's mental model generation in a simple real-time strategy (RTS) game.

CHAPTER 3

PRELIMINARIES

3.1 Markov Decision Process

A Markov Decision Process (MDP), a five-tuple $M = \langle S, A, T, \gamma, R \rangle$ where S is a set of states, A is a set of actions, $T : S \times A \times S \rightarrow [0, 1]$ is a transition function, in which $T(s, a, s')$ is the probability of being in state s' after executing action a in state s, R: $S \rightarrow \mathbb{R}$ (or $R : S \times A \rightarrow \mathbb{R}$) is the reward function, and $\gamma \in [0, 1]$ is the discount factor.

3.2 Partially Observable Markov Game

A Markov Game [157] is the multi-agent version of Markov Decision Process (MDP). We are primarily concerned with a partially observable Markov game. A partially observable Markov game (POMG) for N agents can be defined by a set of global states, S, a set of private observations for each agent, $O_1, O_2, \ldots, O_N$, a set of actions for each agent, $A_1, A_2, \ldots, A_N$, and the transition function, $T : S \times A_1 \times \ldots \times A_N \mapsto S$. In each time step, agent i chooses action, $a_i \in A_i$, obtains reward as a function of state, S, and its action $r_i : S \times A_i \mapsto \mathbb{R}$, and receives a local observation $o_i : S \mapsto O_i$. The initial state is defined by a initial state distribution ρ. Agent i aims to maximize its discounted reward $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$, where $\gamma \in [0, 1]$ is a discounted factor. Our work is based on the framework of POMG augmented with communication.

3.3 Reinforcement Learning: Policy Gradients

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

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

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

3.4 Actor-Critic (AC) Methods

Actor-Critic (AC) methods [158, 159] are an approach to RL that utilize function approximation, in which each agent j has a policy, $\pi^j_\theta(a|s)$, parameterized by θ, that specifies which action, a, to take in each state, s, to maximize the expected future discounted reward. AC methods apply gradient ascent to the actor's parameters, θ, based upon a *critic*, $Q^{\phi}(s, a)$, action-value function [160], parameterized by ϕ, where $Q^{\phi}(s, a)$ approximately solves the credit-assignment problem [161]. By the policy gradient theorem [162], the expected reward maximization (i.e., the AC objective), $J(\theta)$, is maximized via $\nabla_\theta J(\theta) = \mathbb{E}_{\pi^j_\theta}\left[\nabla_\theta \log \pi^j_\theta(a^j_t|o^j_t) Q^{\phi}(o^j_t, a^j_t)\right]$, where a^j_t and o^j_t are the action and observation of agent j, respectively.

3.5 Graph Neural Networks

Graph Neural Networks (GNNs) are a class of deep neural networks that capture the structural dependency among nodes of a graph via message-passing between the nodes, where each node aggregates feature vectors of its neighbors to com-

pute a new feature vector [163, 95]. The canonical feature update procedure via graph convolution operator can be shown as $\bar{h}'_j = \sigma\left(\sum_{k \in N(j)} \frac{1}{c_{jk}} \omega \bar{h}_k\right)$, where $\bar{h}'_j$ is the updated feature vector for node j, $\sigma(.)$ is the activation function and, ω represents the learnable weights. $k \in N(j)$ includes the immediate neighbors of node j where k is the index of neighbor, and c_{jk} is the normalization term which depends on the graph structure. A common choice of c_{jk} is $\sqrt{|N(j)N(k)|}$. In an L-layer aggregation, a node j's representation captures the structural information within the nodes that are reachable from j in L hops or fewer. However, the fact that c_{jk} is structure-dependent can impair generalizability of GNNs when scaling the graph's size. Thus, a direct improvement is to replace c_{jk} with attention coefficients, α_{jk}, computed via Eq. Equation 3.2. In Eq. Equation 3.2, $\bar{W}_{att}$ is the learnable weight, $\|$ represents concatenation, and $\sigma'(.)$ is the LeakyReLU nonlinearity. The Softmax function is used to normalize the coefficients across all neighbors k, enabling feature dependent and structure free normalization [164, 90].

$$\alpha_{jk} = \text{softmax}_k\left(\sigma'\left(\bar{W}_{att}^T\left[\omega \bar{h}_j \,\|\, \omega \bar{h}_k\right]\right)\right) \tag{3.2}$$

3.6 Differentiable Decision Trees (DDTs)

Prior work has proposed differentiable decision trees (DDTs) [67, 131, 165] – a neural network architecture that takes the topology of a decision tree (DT). Similar to a decision tree, DDTs contain decision nodes and leaf nodes; however, each decision node within the DDT utilizes a sigmoid activation function (i.e., a "soft" decision) instead of a Boolean decision (i.e., a "hard" decision). Each decision node, i, is represented by a sigmoid function, displayed in Equation Equation 3.3.

$$y_i = \frac{1}{1 + \exp(-\alpha(\vec{w}_i^T \vec{x} - b_i))} \tag{3.3}$$

Here, the features vector describing the current state, $\vec{x}$, are weighted by $\vec{w}_i$, and a splitting criterion, b_i, is subtracted to form the splitting rule. y_i is the probability of decision node i evaluating to TRUE, and α governs the steepness of the sigmoid activation, where $\alpha \to \infty$ results in a step function. Prior work with discrete-action DDTs modeled each leaf node with a probability distribution over possible output classes [131]. Leaf node distributions, $\vec{L}$, are then weighted by the probability of reaching the respective leaf and summed to produce a final action distribution over possible outputs. For a simple 4-leaf tree, this results in an *fuzzy* output distribution with a complex interplay of node and leaf probabilities, displayed in Equation Equation 3.4.

$$P(a|x) = \vec{L}_1(y_1 * y_2) + \vec{L}_2(y_1 * (1 - y_1)) + \vec{L}_3((1 - y_1) * y_3) + \vec{L}_4((1 - y_1)(1 - y_3)) \quad (3.4)$$

[0]For figure simplicity, when displaying the crisp node (blue block), we assume $\alpha > 0$ in the fuzzy node (orange block). If $\alpha < 0$, the sign of the inequality would be flipped (i.e., $w_i^{k_i} x^{k_i} < b$).

CHAPTER 4
THE IMPORTANCE OF COMMUNICATION IN MULTI-AGENT COORDINATION

In this chapter, we design a multi-agent coordination algorithm, Multi-Agent Graph AttentIon Communication (MAGIC) [6], that allows agents to determines "when" and "whom" with to communicate via an end-to-end framework. Utilizing our novel formulation with multi-agent reinforcement learning, we see that we can utilize simulation to produce high-performance collaborative policies and find benefits in the quality of coordination by utilizing precise, targeted communication.

4.1 Introduction

Communication is a key component of successful coordination, enabling the agents to convey information and cooperate to collectively achieve shared goals [166, 24]. In high-performing human teams, human experts judiciously choose when to communicate and whom to communicate with, communicating only when beneficial [26, 27, 28]. Each team member exhibits the role of a communicator and message receiver, relaying valuable information to the right teammates and incorporating received information effectively. Communicators typically hold minimal interactions, spending their energy and resources to communicate with only those that require information. On the side of message receivers, redundant information can disturb decision making and lower working efficiency, while timely and targeted information is beneficial. What is more, knowing how to use and interpret the messages from others is also a crucial step.

In this chapter, we propose Multi-Agent Graph-attentIon Communication (MAGIC), a novel graph communication protocol that determines "when" and "whom" with

to communicate via an end-to-end framework. We set a new state-of-the-art in communication-based multi-agent reinforcement learning (MARL) by modeling the topology of interactions among agents (the local and global characterization of connections between agents [45]) as a dynamic directed graph that accommodates time-varying communication needs and accurately captures the relations between agents. Our proposed framework emulates the features of an effective human-human team through its key components, 1) the Scheduler, which helps each agent to decide when it should communicate and whom it should communicate with, and 2) the Message Processor, which integrates and processes received messages in preparation for decision making. We find MAGIC produces high-performance, cooperative behavior through its efficient communication protocol.

There has been recent success in MARL for Multiplayer Online Battle Arena (MOBA) games such as StarCraft II and Dota II [167, 168, 169]. MARL seeks to enable agents to share information to improve team performance [51, 52, 86, 55, 170, 171]. However, most prior work in MARL fails to capture the complex relations among agents, leading to low-performance and inefficient communication. While [51] and [88] are able to efficiently decide when to broadcast messages, agents will broadcast these messages to all other agents without targets, resulting in wasteful communication. Even with targeted communication [55], failure to assess when to communicate results in poor performance, as we display in section 4.4. However, determining when to communicate and whom to communicate with is not enough. Selectively utilizing received messages can significantly improve performance. Yet, none of these methods simultaneously address "when" and with "whom", and "how" to communicate while modeling agent interaction topology.

Our communication protocol, MAGIC, utilizes a Scheduler consisting of a graph attention encoder and a differentiable hard attention mechanism to decide when to communicate and whom to communicate with. This information is encoded

within a directed graph, allowing us to represent the interaction among agents precisely. The Message Processor consisting of a Graph Attention Network (GAT), utilizes received messages and the directed graph to intelligently and efficiently process messages. The encoded messages are then used in each agent's policy, leading to high-performance cooperation and efficient communication, as shown in section 4.4. We provide the following contributions:

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

2. Enable GATs in the Message Processor to deal with differentiable graphs, which is not supported by standard GATs. In this way, the framework is completely differentiable and can be trained in an end-to-end manner.

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

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

4.2 Method

In this section, we introduce our proposed Multi-Agent Graph-attentIon Communication protocol, MAGIC. We consider a partially observable setting of N agents, where agent i receives local observation, o_i^t, at time, t, containing local information from the global state, S. The agent, i, learns a communication-based policy, π_i, to

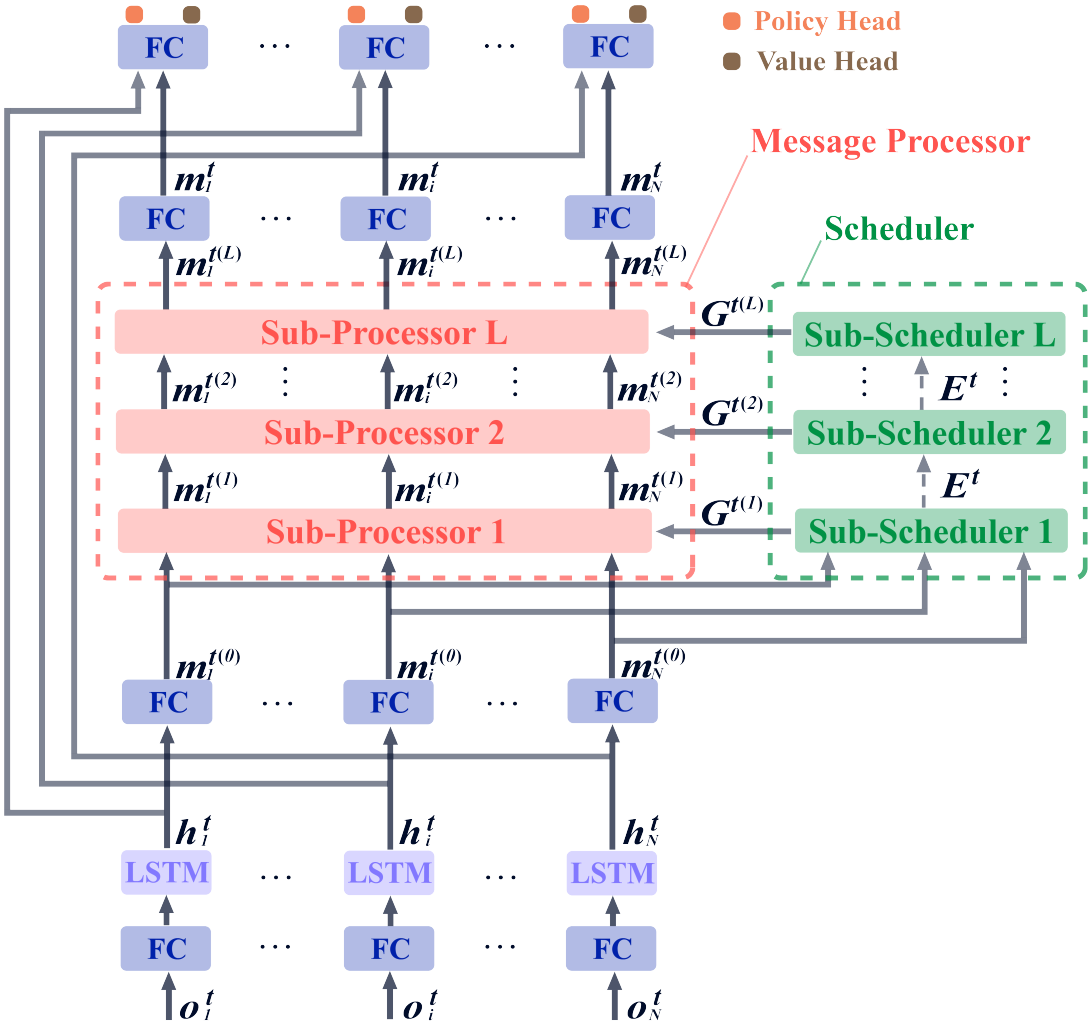


Figure 4.1: This figure displays the framework of our multi-agent graph-attention communication protocol.

output a distribution over actions, $a_i^{(t)} \sim \pi_i$, at each time step, t. Here, we present an overview of our framework, the description of our protocol's key components (i.e., the Scheduler and Message Processor), and our training procedure.

4.2.1 Overview

Our proposed graph-attention communication protocol is displayed in Figure 4.1. At each time step, t, the observation for each agent, o_i^t, is first encoded using an agent-specific fully-connected layer (FC). The encoded observation is passed into an agent-specific LSTM cell to generate a hidden state, h_i^t, as shown in Equation 4.1.

$$h_i^t, c_i^t = LSTM(e(o_i^t), h_i^{t-1}, c_i^{t-1}) \tag{4.1}$$

In this equation, c_i^t is the cell state for agent, i, at time step, t, and $e(\cdot)$ is a fully-connected layer acting as an encoder for the observation. The hidden state, h_i^t, is then encoded as a message, $m_i^{t(0)} = e_m(h_i^t)$, through the encoder, $e_m(\cdot)$ (a fully-connected layer). Here, the exponent notation for the message, $m_i^{t(0)}$, denotes that message is for agent i, and is prior to any message aggregation or processing. We refer to this stage, where the message has not been processed, as round 0, giving the exponent notation, $t(0)$.

As shown in Figure 4.1, we define the function module to help agents decide whom to send messages at each time step as the "Scheduler" and define the function module to process messages as "Message Processor." The Scheduler and the Message Processor may include multiple sub-schedulers and sub-processors, respectively. Prior work has termed the procedure of processing messages for multiple iterations as multi-round communication [55]. As multi-round communication has been shown to improve performance, our protocol supports L rounds of communication, where $L \in \mathbb{N}$. A round of communication, l, is defined as a forward pass through a sub-scheduler and sub-processor. As shown in Figure 4.1, the encoded messages, $m_i^{t(0)}$ are passed into Sub-Scheduler 1 and Sub-Processor 1 (i.e., the sub-scheduler and sub-processor at round 1).

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

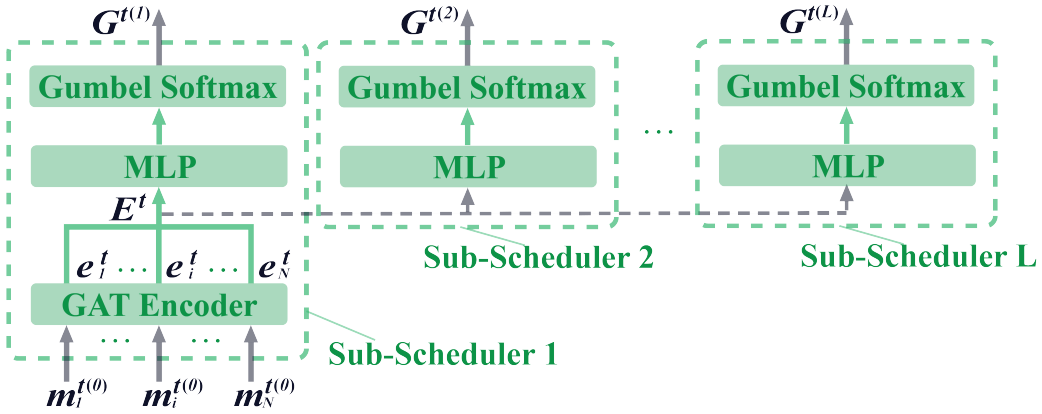


Figure 4.2: This figure displays the details and components of the Scheduler.

further processed, the set of messages outputted from round l, $\{m_i^{t(l)}\}_1^N$, are passed into the Sub-Scheduler $l+1$ and the Sub-Processor $l+1$, producing adjacency matrix $G^{t(l+1)}$ and messages $\{m_i^{t(l+1)}\}_1^N$ respectively.

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

4.2.2 The Scheduler

The Scheduler decides when each agent should send messages and whom each agent should address messages to, as shown in Figure 4.2. As a black-box, the Scheduler takes as input the encoded messages, $\{m_i^{t(0)}\}_1^N$, and outputs the directed graphs, $\{G^{t(l)}\}_1^L$, as represented in $f_{Sched}(\cdot)$ shown in Equation 4.2.

$$\{G^{t(l)}\}_1^L = f_{Sched}\left(m_1^{t(0)}, \cdots, m_N^{t(0)}\right) \tag{4.2}$$

As noted in subsection 4.2.1, the Scheduler consists of L Sub-Schedulers, each producing an adjacency matrix $G^{t(l)}$. A Sub-Scheduler consists of a GAT encoder and a hard attention mechanism that uses a multi-layer perceptron (MLP) and a Gumbel Softmax function [172]. The GAT encoder helps encode local or global information for an agent efficiently, and it is only used in the first Sub-Scheduler. We adopt the same form of GATs as proposed in [164], where the attention mechanism is expressed in Equation 4.3.

$$\alpha_{ij}^{S} = \frac{exp\left(LReL\left(a_S^T[W_S m_i^{t(0)} \| W_S m_j^{t(0)}]\right)\right)}{\sum_{k \in N_i^t \cup i} exp\left(LReL\left(a_S^T[W_S m_i^{t(0)} \| W_S m_k^{t(0)}]\right)\right)} \tag{4.3}$$

Here, $LReL(\cdot)$ is the LeakyReLU [173], $a_S \in R^{D'}$ is a weighting vector, $N_i^t \cup i$ is the set of neighboring agents for the i-th agent, including agent i, at time step, t, and $W_S \in R^{D' \times D}$ is a weighting matrix, where D' and D refer to the output feature cardinality and message cardinality, respectively. The node features of each agent are obtained through Equation 4.4.

$$e_i^t = ELU\left(\sum_{j \in N_i^t \cup i} \alpha_{ij}^S W_S m_j^{t(0)}\right) \tag{4.4}$$

Here, $ELU(\cdot)$ is the Exponential Linear Unit (ELU) function. After concatenating the node feature vectors pairwise, supposing $E_{i,j}^t = (e_i^t \| e_j^t)$, we obtain a matrix $E^t \in \mathbb{R}^{N \times N \times 2D}$, where $E_{i,j}^t$ represents a high-level representation of relational features between the i-th and j-th agent. Setting E^t as the input to an MLP followed by a Gumbel Softmax function, we can get an adjacency matrix $G^{t(l)}$, consisting of binary values, representing a directed graph. If the element $g_{ij}^{t(l)}$ in $G^{t(l)}$ is 1, the j-th agent will send a message to i-th agent. Otherwise ($g_{ij}^{t(l)} = 0$), the j-th agent will not send any message to i-th agent. As the "Straight-Through" trick is used in the Gumbel Softmax [172], sampling discrete values maintain the gradients, and the adjacency

matrix $G^{t(l)}$ is differentiable.

4.2.3 The Message Processor

The Message Processor helps agents integrate messages for intelligent decision making. As a black-box, it takes in the encoded messages, $\{m_i^{t(0)}\}_1^N$, and the graphs generated by the Scheduler, $G^{t(1)} \cdots, G^{t(L)}$, and outputs the processed messages, $\{m_i^{t(L)}\}_1^N$. We represent the Message Processor in Equation 4.5.

$$\{m_i^{t(L)}\}_1^N = f_{MP}\left(m_1^{t(0)}, \cdots, m_N^{t(0)}, G^{t(1)}, \cdots, G^{t(L)}\right) \tag{4.5}$$

As stated in subsection 4.2.1, the Message Processor consists of L Sub-Processors, each producing a set of encoded messages, $\{m_i^{t(l)}\}_1^N$. A Sub-Processor, for a single round of communication, includes a GAT layer receiving messages, $\{m_i^{t(l-1)}\}_1^N$, from all agents and the adjacency matrix, $G^{t(l)}$, as input. The Sub-Processor helps agents process received messages. The calculation of the attention coefficient in the GAT layer is shown in Equation 4.6.

$$\alpha_{ij}^P = \frac{g_{ij}^{t(l)} exp\left(LReL\left((a_P^{(l)})^\top [W_P^{(l)} m_i^{t(l-1)} \| W_P^{(l)} m_j^{t(l-1)}]\right)\right)}{\sum_{k=1}^N g_{ik}^{t(l)} exp\left(LReL\left((a_P^{(l)})^\top [W_P^{(l)} m_i^{t(l-1)} \| W_P^{(l)} m_k^{t(l-1)}]\right)\right)} \tag{4.6}$$

Here, l is the round of communication, $g_{ij}^{t(l)} \in \{0, 1\}$ is a binary value in the adjacency matrix, $G^{t(l)}$, $W_P^{(l)} \in R^{D'' \times D}$ is a weighting matrix, and $a_P^{(l)} \in R^{D''}$ is a weighting vector. It should be noted that our graphs are capable of a self-loop, where an agent will "send" a message to itself, and use its own message in the integration of received messages. While the calculation of the coefficient for a standard GAT layer is a non-differential operation for the graph, using Equation 4.6 allows us to retain the gradient of $g_{ij}^{t(l)}$. Thus, the Scheduler can preserve the gradient flow for end-to-end training, avoiding the need to design an extra loss function to train the Scheduler. In practice, we find using multi-head attention [164] and adding a bias to the output

message to be beneficial. The output message of sub-processor l can be obtained via Equation 4.7.

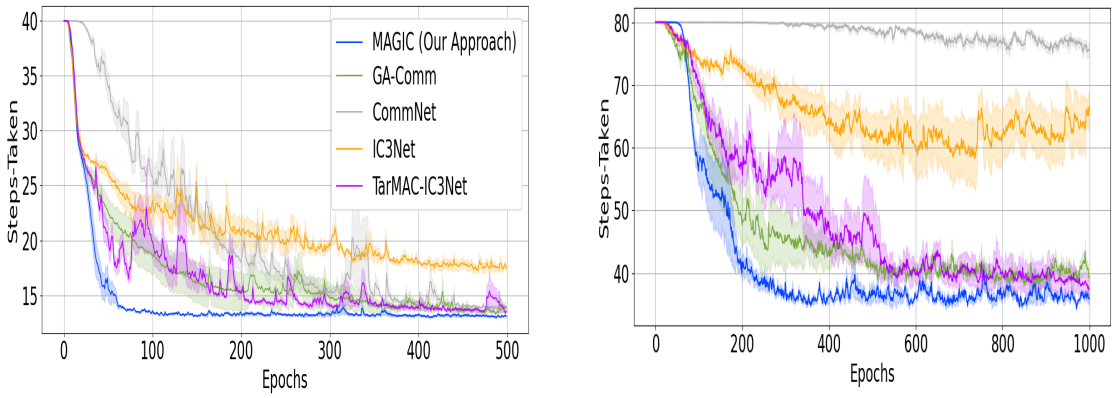
$$m_i^{t(l)} = ELU\left(\sum_{j=1}^{N} \alpha_{ij}^{P} W_P^{t(l)} m_j^{t(l-1)}\right) \tag{4.7}$$

4.2.4 Training

In our experiments, the parameters of the fully-connected layers and LSTM in the policy network are shared across homogeneous agents to improve training efficiency. We employ a multi-threaded synchronous multi-agent policy gradient [51] and utilize an extra value head in the policy network to estimate the value function, $V_\phi(o_i^t)$, at observation o_i^t, which will serve as a state-independent baseline. In addition to optimizing the discounted total reward with policy gradient, the model also minimizes the squared error between the estimated value and the Monte-Carlo estimate. The two loss functions are balanced by a coefficient, β. We define the overall loss function as $\mathcal{L}(\cdot)$ and the policy function denoted as $\pi_\theta(a_i^t|o_i^t)$. Parameters, θ, of the policy and, ϕ, of the value function, share most of their parameters except the parameters in the policy and value heads. Our model is updated via minimizing the loss function displayed in Equation 3.1.

$$\begin{aligned}\nabla_{\theta,\phi}\mathcal{L}(\theta,\phi) = \frac{1}{t_{max}} \sum_{i=1}^{N} \sum_{t=1}^{t_{max}} [-\nabla_\theta \log \pi_\theta(a_i^t|o_i^t)(R_i^t - V_\phi(o_i^t)) \\ +\beta\nabla_\phi(R_i^t - V_\phi(o_i^t))^2]\end{aligned} \tag{4.8}$$

Here, R_i^t is the discounted total reward for agent i in an episode, and t_{max} is the number of steps taken within a batch. Different threads in training share the parameters, θ, and ϕ, and calculate their own gradients. The threads synchronously accumulate gradients and update θ and ϕ within each batch. A summary of the training procedure of our multi-agent graph-attention communication model is described in Algorithm Algorithm 1. In Algorithm Algorithm 1, we start by initializing the number of agents alongside several training parameters, including



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

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

Figure 4.3: This figure displays the average steps taken to finish an episode as training proceeds in each level of the Predator-Prey environment. The shaded regions represent standard error. A lower value for steps taken on the vertical axis is better.

threads, batch size, maximum steps in an episode, and maximum steps in a batch, shown in Step 1. For each update per thread, we start by initializing a set of thread parameters and a replay buffer, D, as displayed in Step 4. After receiving the initial hidden state and observation for each agent (shown in Steps 7 and 8), we can utilize the Scheduler (containing L sub-schedulers) to output adjacency graphs, determining the communication pattern. The Message Processor (containing L sub-processors) can use these graphs and the encoded messages from round zero to produce messages for each agent, as shown in Steps 13 and 14. Once each agent has its selected message inputs, we can determine an action probability distribution from the policy and perform the action sampled from this distribution, shown in Steps 15 and 16. Storing this information in our replay buffer, we can then complete the episode and proceed to compute gradients with Equation 3.1, as shown in Step 26. Accumulating gradients across threads, we can update our policy and value function, as shown in Steps 29 and 31.

4.3 Evaluation Environments

We utilize three domains, including the Predator-Prey [51], Traffic Junction [52] and complex Google Research Football environment [53], to evaluate the utility of the proposed communication protocol. Predator-Prey and Traffic Junction are common MARL benchmarks [52, 55]. Google Research Football (GRF) presents a difficult challenge, as it has sparse rewards, stochasticity, and adversarial agents.

4.3.1 Predator-Prey

We utilize the predator-prey environment from [51]. Here, there are N predators with limited visions searching for a stationary prey. A predator or a prey occupies a single cell within the grid world at any time, and its location is initialized randomly at the start of each episode. The state at each point in the grid is the concatenation of a one-hot vector that represents its own location and binary values indicating the presence of predator and prey at this location. The observation of each agent is a concatenated array of the states of all points within the agent's vision. The predators can take actions *up*, *down*, $left$, *right* or *stay*. We utilize the "mixed" mode of Predator-Prey in which the predator incurs a reward -0.05 for each time step until the prey is found. An episode is defined as successful if all the predators find the prey before a predefined maximum time limit. We create two levels of difficulty in this environment. The difficulty varies as the grid size, and the number of predators increase, as more coordination is required to achieve success. The corresponding grid sizes and the number of predators are set to 10×10 with 5 predators and 20×20 with 10 predators. We set the maximum steps for an episode (i.e., termination condition) to be 40 and 80, respectively. The vision is set to a unit length. We define a higher-performing algorithm in this domain as one that minimizes the average steps to complete an episode. A depiction of the more

challenging variant of Predator-Prey (10-agent) is displayed in Figure 4.4.

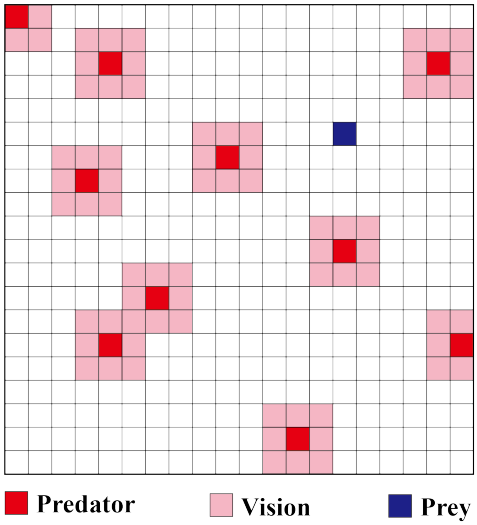


Figure 4.4: The visualization of the 10-agent Predator-Prey task. The predators (in red) with limited visions (light red region) of size one are searching for a randomly initialized fixed prey (in blue).

4.3.2 Traffic Junction

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

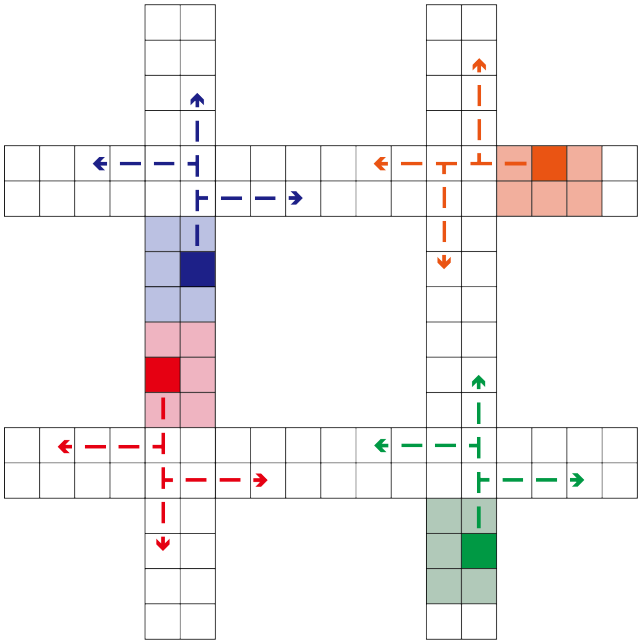


Figure 4.5: The visualization of the hard level Traffic Junction task. This task consists of four, two-way roads on a 18×18 grid with eight arrival points, each with seven different routes. Each agent is with a limited vision of size 1.

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

We validate our algorithm on three difficulty levels. The easy level consists of two, one-way roads on a 7×7 grid with at most five agents ($N_{max} = 5$, $p_{arrive} = 0.3$). For the medium level, the junction consists of two, two-way roads on a 14×14 grid with at most ten agents in the domain ($N_{max} = 10$, $p_{arrive} = 0.2$). Hard consists of four, two-way roads on a 18×18 grid with at most twenty agents in the domain ($N_{max} = 20$, $p_{arrive} = 0.05$). The goal is to maximize the success rate (i.e., no collisions within an episode). We display the hard level in Figure 4.5, where the domain consists of four two-way roads on an 18×18 grid with eight arrival points, each with seven different routes, and there are at most twenty agents ($N_{max} = 20$, $p_{arrive} = 0.05$).

Figure 4.6: The visualization of 3 vs. 2 in Google Research Football. The five people shown in this figure are three offending players, one defending player and the goalie (left to right).

4.3.3 Google Research Football

Our final domain of Google Research Football [53] presents a challenging, mixed cooperative-competitive, multi-agent scenario with high stochasticity and sparse rewards. Google Research Football (GRF) is a physics-based 3D soccer simulator for reinforcement learning. This last domain presents an additional challenge as there are opponent artificial agents (AIs), significantly increasing the complexity of the state-action space. We present a depiction of this environment in Figure 4.6. To align with the partially observable setting, we extract the local observations from the provided global observations. The local observations include the relative positions of the players on both teams, the relative position of the ball, and one-hot encoding vectors which represent the ball-owned team and the game mode. GRF provides 19 actions, including moving actions, kicking actions, and other actions such as dribbling, sliding, and sprint. GRF provides several pre-defined reward signals, consisting of a scoring and a penalty box proximity reward. The penalty box proximity reward is shaped to push attackers to move forward towards certain locations. Many MARL frameworks have required these highly shaped rewards functions to perform well [53]. However, we choose to use only the

scoring reward to verify the ability for our algorithm and baselines to function in a high-complexity stochastic domain with sparse rewards. **Accordingly, the only reward all agents will receive in our evaluation is** +1 **when scoring a goal.** The termination criterion is the team scoring, ball out of bounds, or possession change. We evaluate algorithms in the football academy scenario 3 vs. 2, where we have 3 attackers vs. 1 defender, and 1 goalie. The three offending agents are controlled by the MARL algorithm, and the two defending agents are controlled by a built-in AI. We find that utilizing a 3 vs. 2 scenario challenges the robustness of MARL algorithms to stochasticity and sparse rewards. In this domain, we seek to maximize the average success rate (i.e., a goal is scored) and minimize the average steps taken to complete an episode, thereby scoring a goal in the shortest amount of time. We show in subsection 4.4.3 that our method outperforms all prior state-of-the-art baselines.

Training Details: We distribute the training over 16 threads and each thread runs batch learning with a batch size of 500. The threads share the parameters of the policy network and update synchronously. There are 10 updates in one epoch. We use RMSProp with a learning rate of 0.001 in all the domains except Predator-Prey ten-agent scenario where we use 0.0003. The value coefficient β and discount factor λ are set to 0.01 and 1 respectively. The size of each agent's hidden state for LSTM is 128. The sizes of original encoded messages and the final messages for decision making are 128. 2/3 layers of GNNs have been used in practice and shown to work well [164]. Empirically, we find that two rounds of communication achieve the best performance with comparable training speeds to simpler methods such as CommNet and IC3Net. As such, we use two rounds of communication to test the performance of our method in all domains, and the number of heads for the first GAT layer (sub-processor 1) is set to be 4, 4, 1 in Predator-Prey, Traffic Junction and GRF respectively, and the number of heads for the output GAT layer

(sub-processor 2) is set to be 1. We use one-round communication for efficiency evaluation for fair comparison, and the number of heads for the GAT layer is 1. The output size of the GAT encoder in the Scheduler is set to 32. We implement our method and baselines on each task over 5 random seeds and average the results. We provide our code at https://github.com/CORE-Robotics-Lab/MAGIC.

4.4 Results and Discussion

In this section, we evaluate the performance of our proposed method on three environments, including Predator-Prey [51], Traffic Junction [52], and Google Research Football [53]. We benchmark our approach against a variety of state-of-the-art communication-based MARL baselines, including CommNet [52], IC3Net [51], GA-Comm [54], and TarMAC-IC3Net [55]. We implement our method and baselines on each task, averaging the best performance at convergence over 5 random seeds. Following an analysis of performance, we evaluate MAGIC's communication efficiency, concluding MAGIC presents a new state-of-the-art in both performance and efficiency in MARL.

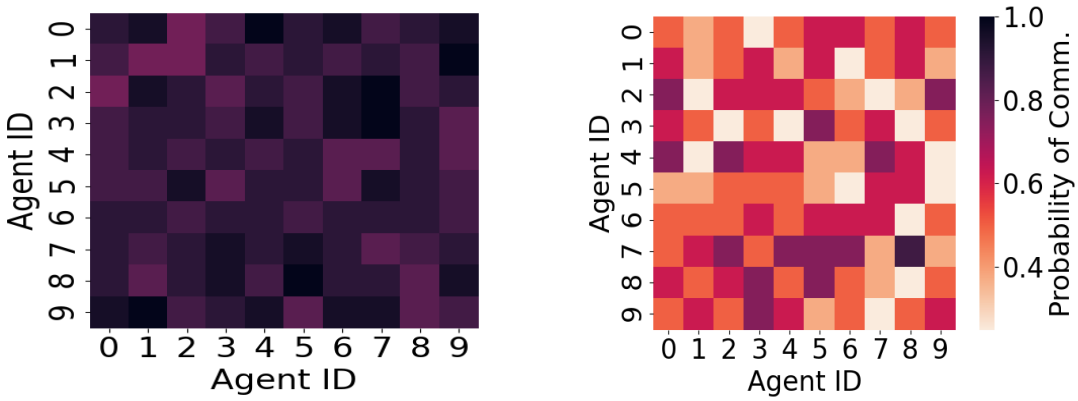
4.4.1 Predator-Prey

Figure 4.3 depicts the average steps taken for the predators to locate the prey. In

Table 4.1: This table presents the number of steps taken to complete an episode at convergence in Predator-Prey.

Method	10×10, 5 agents	20×20, 10 agents
MAGIC (Our Approach)	**12.72 ± 0.03**	**32.88 ± 0.14**
CommNet [52]	13.16 ± 0.04	73.12 ± 0.68
IC3Net [51]	15.60 ± 0.35	55.13 ± 4.80
TarMAC-IC3Net [55]	13.32 ± 0.11	36.16 ± 0.97
GA-Comm [54]	13.06 ± 0.09	35.78 ± 0.37

both the five- and ten-agent cases, our method converges faster and can achieve



(a) Before prey found.

(b) After prey found.

Figure 4.7: This figure displays the average steps taken to finish an episode as training proceeds in each level of the Predator-Prey environment. The shaded regions represent standard error. A lower value for steps taken on the vertical axis is better.

better performance than the baselines. Our method converges 52% faster than the next-quickest baseline while still achieving the highest performance. Figure 4.3 shows that as the number of agents increases, some baselines (CommNet and IC3Net) are unable to learn a coherent policy. Table Table 4.1 shows the results of average steps taken to reach the prey at convergence. While approaches such as GA-Comm and TarMAC-IC3Net learn competitive policies for the five agent case, these benchmarks perform much worse than our algorithm in the ten agent case, suggesting a lack of scalability.

Predator-Prey Communication Heatmaps - Figure 4.7 depicts communication heatmaps in Predator-Prey domain with 10 agents in an episode of 31 steps. The color is associated with the probability of communication, with darker colors representing more intensive communication between the two agents. The vertical axis represents message receivers, and the horizontal axis represents message senders. Agent 5 first reaches the prey at step 23, and the other 9 agents quickly reach the prey in the following 7 steps. Figure 4.7a displays the communication before the first agent (agent 5) reaches its prey and Figure 4.7b displays the communication

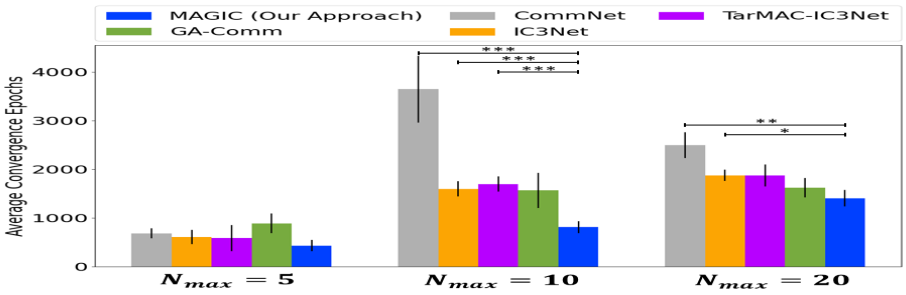


Figure 4.8: This figure displays the average number of epochs for convergence in Traffic Junction with standard error bars.

Table 4.2: This table presents the success rate at convergence in Traffic Junction.

Method	7×7, $N_{max} = 5$, $p_{arrive} = 0.3$	14×14, $N_{max} = 10$, $p_{arrive} = 0.2$	18×18, $N_{max} = 20$, $p_{arrive} = 0.05$
MAGIC (Our Approach)	**99.9 ± 0.1 %**	**99.9 ± 0.1 %**	**98.0 ± 0.8 %**
CommNet [52]	99.3 ± 0.6%	97.2 ± 0.3%	66.7 ± 1.6%
IC3Net [51]	97.8 ± 1.0%	96.0 ± 0.7%	85.4 ± 2.5%
TarMAC-IC3Net [55]	84.8 ± 4.5%	95.5 ± 1.3%	88.1 ± 1.9%
GA-Comm [54]	95.9 ± 0.1%	97.1 ± 0.7%	95.8 ± 1.1%

afterwards. We can see that agents communicate with each other intensively before finding the prey. After finding the prey, communication becomes unnecessary, and the learned communication graphs should be sparse, which is the behavior we see in Figure 4.7b. This inspection supports that MAGIC learns to communicate only when beneficial to team performance.

4.4.2 Traffic Junction

We evaluate our method in the Traffic Junction domain for the cases of a maximum of 5 agents, 10 agents and 20 agents at a junction. Table Table 4.2 shows the success rate (i.e., no collision in an episode) for each method at convergence in Traffic Junction. Our algorithm achieves near-perfect performance after convergence, widely outperforming all benchmarks in its success rate. Figure 4.8 depicts the average number of epochs taken to converges for each method. Our method

Table 4.3: This table displays the success rate and average steps taken to finish an episode in GRF.

Method	Success Rate	Steps Taken
MAGIC (Our Approach)	**98.2 ± 1.0%**	**34.30 ± 1.34**
Ours (without the Scheduler)	91.0 ± 4.6%	36.31 ± 2.59
CommNet [52]	59.2 ± 13.7%	39.32 ± 2.35
IC3Net [51]	70.0 ± 9.8%	40.37 ± 1.22
TarMAC-IC3Net [55]	73.5 ± 8.3%	41.53 ± 2.80
GA-Comm [54]	88.8 ± 3.9%	39.05 ± 3.05

maintains quick convergence as the number of agents increase. However, several of the benchmarks experience a slowdown in their convergence rate.

Impact of the Message Processor - In Traffic Junction, we allow all agents to communicate to accelerate training for all methods, common in highly vision-limited environments [51]. As we are using complete graphs (i.e., Scheduler is not used), our SOTA performance in the Traffic Junction domain displays that the MP considerably contributes to the success of our algorithm.

4.4.3 Google Research Football

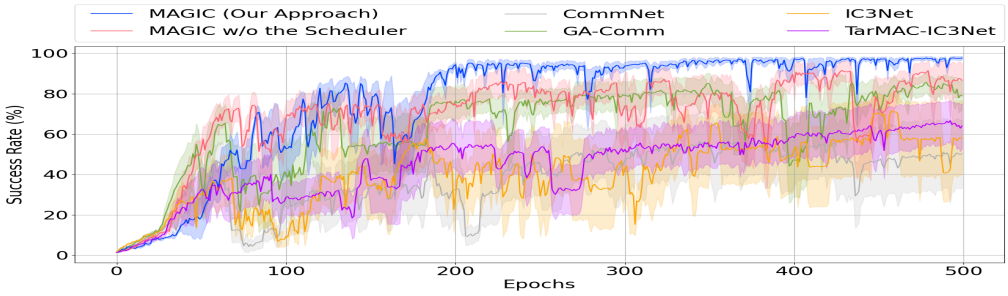


Figure 4.9: This figure displays the success rate in GRF as training proceeds. As shown, our method achieves the highest performance, acheiving near-perfect success at scoring.

Lastly, we evaluate our algorithm and several state-of-the-art MARL baselines in the GRF environment. The results presented provide some insight into each algorithm's ability to handle stochasticity, sparse rewards, and a high-complexity

state-action space. We utilize the scoring success rate as our metric of evaluation.

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

Figure 4.9 displays the success rate for offending agents to score as the training proceeds. Our method converges to a higher-performing policy than all the baselines. While utilizing the Scheduler may adversely affect early performance, it leads to a significant improvement in the performance of our method. In our settings, although each agent only has local observations, it is able to completely observe the state space. MAGIC without the ability to utilize a scheduler performs poorly. It is interesting to note that even with unlimited vision, utilizing a complete graph for communication performs much worse than utilizing a scheduler that gives precise and targeted communication. Table Table 4.3 displays the success rate and average steps taken to finish an episode at convergence. Our method has a success rate of 98.5%, which is approximately a 10.5% improvement over the next-best baseline of GA-Comm. Furthermore, our method has approximately a 8x lower average variance than others at convergence. Even though we do not have rewards or time penalties on finishing an episode, and the discount factor is set to 1, we find our learned policy can still score within a short time. Our method achieves the lowest number of average steps, scoring 13.8% more quickly than the closest benchmark of GA-Comm. As we have have outperformed all benchmarks within the previous two domains and in the complex GRF environment, we conclude that our method of "Multi-Agent Graph-attentIon Communication" (MAGIC) is high-performing, scales well to the number of agents, robust to stochasticity, and performs well with sparse rewards.

4.4.4 Communication Efficiency

We present an analysis of the communication efficiency of our method in Table Table 4.4. Communicating efficiently can save resources and allow for messages to be processed more easily. To gauge the efficiency, we utilize the performance improvement due to communication and divide the communication graph density. The results here are averaged over 3 random seeds. The graph densities are determined using the sparsity of the adjacency matrix. A fully-connected graph corresponds to a density of 1, and a graph with no connections corresponds to a density of 0. We perform this analysis within a Predator-Prey domain of grid size $5x5$ with 3 predators, where a performance improvement refers to a reduction in steps. To obtain the performance improvement due to communication, we evaluate a communication-blocked variant of each method and compare the performance to the method itself. All methods use the same message size and one round of communication, as some baselines do not support multi-round communication. Utilizing the performance improvement due to communication divided by the graph density as our metric for communication efficiency, we see that MAGIC is the most efficient. MAGIC communicates 27.4% more efficiently on average than baselines while also achieving the highest performance within this domain. Alongside achieving state-of-the-art performance, we are able to most efficiently send and receive messages, displaying the strength of our method.

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

Table 4.4: Communication efficiency measured as the performance improvement with communication divided by graph density.

Method	Graph Density	Avg. Steps w/ Comms	Performance Improvement	$\frac{\text{Improvement}}{\text{Density}}$
MAGIC (Our Approach)	**0.644**	**8.504**	**7.562**	**11.743**
CommNet [52]	0.667	9.216	6.455	9.681
IC3Net [51]	0.638	9.208	6.421	10.058
TarMAC-IC3Net[55]	0.856	9.376	5.958	6.956
GA-Comm [54]	0.514	9.334	5.868	11.391

4.4.5 Discussion

Across multiple test domains, we set a new state-of-the-art in MARL performance, outperforming baselines, including [52, 51, 54, 55]. Across our domains, we achieve an average 5.8% improvement in steps taken (Predator-Prey), average 1.9% improvement in success rate (Traffic Junction), 10.5% improvement in success rate (GRF), and 13.8% improvement in steps taken compared to the closest benchmark which varies across each domain. The strong performance improvement we achieve in GRF suggests our approach is better able to scale to high-dimension state-action spaces while effectively handling stochasticity and sparse rewards. While our architecture shares some attributes as GA-Comm [54], GA-Comm has several drawbacks including large epoch training times due to its bi-directional LSTM in its hard attention mechanism, a dot-product soft attention mechanism *without scaling*, and the inability to extend to multi-round communication. Our approach takes significantly less compute by avoiding any recurrent structures in the Scheduler. Specifically, GA-Comm requires 2x long to train with 3 agents, 3x long with 5 agents, and 4x as long with 10 agents. Prior work [174] has also shown that additive attention (used in the GAT layers in our Message Processor (MP)) outperforms dot-product attention without scaling when the message size is large. MAGIC's Scheduler can explicitly learn and generate different graphs for different

rounds of communication, allowing for higher performance. Additionally, MAGIC achieves the highest ratio of performance improvement to communication graph density, outperforming benchmarks by 27.4% on average.

4.5 Physical Robot Demonstration

We present a demonstration of our algorithm in a similar 3-vs.-2 soccer scenario on physical robots in the Robotarium, a remotely accessible swarm robotics research platform [175]. This demonstration displays the feasibility of trajectories produced by our MARL algorithm. We present a depiction of MAGIC's deployed trajectory in Figure 4.10. A video is attached in the supplementary.

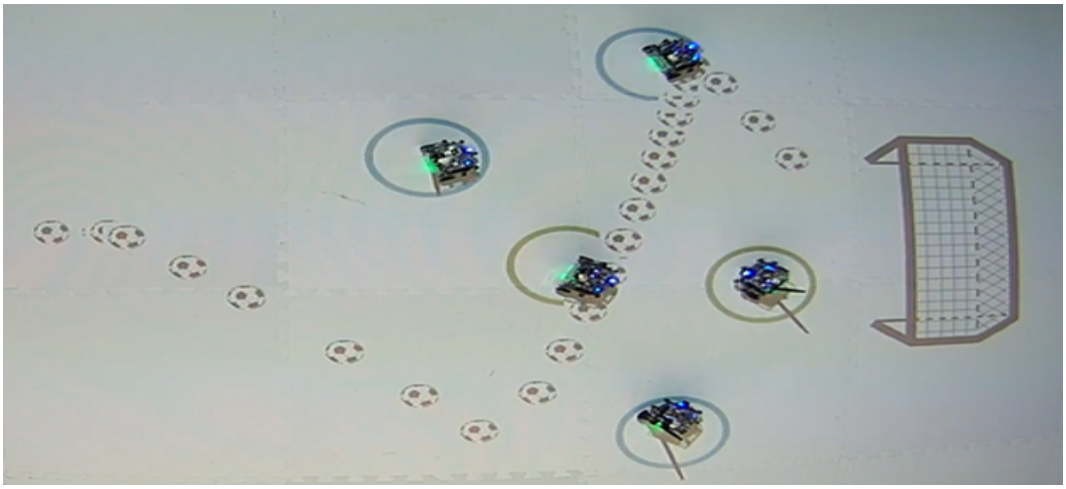


Figure 4.10: This figure displays a demonstration of our algorithm on physical robots on the Robotarium platform. The display shows a 3 vs. 2 soccer scenario, with blue agents as the attackers, and red agents as defenders.

4.6 Conclusion

In this chapter, we propose a novel, end-to-end-trainable, graph-attention communication protocol, MAGIC, that utilizes a Scheduler to solve the problems of when to communicate and whom to address messages to, and a Message Processor to integrate and process messages. We evaluate our method and baselines in

several environments, including Predator-Prey, Traffic Junction, and the more complex Google Research Football, achieving state-of-the-art performance. In GRF, we achieve a 98.5%, near-perfect success rate in scoring, while most baselines struggle to reach 70%. Not only does MAGIC produce state-of-the-art results, MAGIC is able to converge 52% faster than the next-quickest baseline, and communicates 27.4% more efficiently than the average baseline. Finally, we demonstrate feasibility of MAGIC on a physical robot testbed.

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

Initialize max updates M, agents N, threads L, max steps in an episode T_e, max steps in a batch T_b
for update = 1 to M **do** $d\theta \leftarrow 0$, $d\phi \leftarrow 0$
  for thread $k = 1$ to K **do in parallel**
    Initialize params $\theta_k \leftarrow \theta$, $\phi_k \leftarrow \phi$, buffer D, step-count $t \leftarrow 1$
    while $t < T_b$ **do**
      Initialize thread step counter $t' \leftarrow 1$
      Initialize h_i^{t-1}, c_i^{t-1} for each agent i
      Reset environment and get o_i^t for each agent i
      while $t' < T_e$ and not terminal **do**
        $h_i^t, c_i^t = LSTM(e(o_i^t), h_i^{t-1}, c_i^{t-1})$ for each agent i
        $m_i^{t(0)} = e_m(h_i^t)$ for each agent i
        $\{G^{t(l)}\}_1^L = f_{Sched}\left(m_1^{t(0)}, \cdots, m_N^{t(0)}\right)$
        $\{m_i^{t(L)}\}_1^N = f_{MP}\left(m_1^{t(0)}, \cdots, m_N^{t(0)}, G^{t(1)}, \cdots, G^{t(L)}\right)$
        $m_i^t = e'_m(m_i^{t(L)})$ for each agent i
        Calculate $\pi_{\theta_k}(a_i^t|o_i^t)$ and $V_{\phi_k}(o_i^t)$ for each agent i
        Perform $a_i^t \sim \pi_{\theta_k}(a_i^t|o_i^t)$ for each agent i
        Receive r_i^t and o_i^{t+1} for each agent i
        Store $(o_i^t, a_i^t, \pi_{\theta_k}(a_i^t|o_i^t), V_{\phi_k}(o_i^t), r_i^t, o_i^{t+1})$ in D
        $t \leftarrow t + 1$, $t' \leftarrow t' + 1$
      end while
    end while
    $t_{max} \leftarrow t$
    for $t = t_{max}, t_{max} - 1, \cdots, 1$ **do**
      $R_i^t = 0$ if o_i^{t+1} is terminal else $R_i^t = r_i^t + \gamma R_i^{t+1}$ using D
    end for
    Calculate $d\theta_k$ and $d\phi_k$ using D with Equation 4.8
  end for
  for thread $k = 1$ to K **do**
    Accumulate gradients: $d\theta \leftarrow d\theta + d\theta_k$, $d\phi \leftarrow d\phi + d\phi_k$
  end for
  Perform update of θ using $d\theta$, and of ϕ using $d\phi$
end for

CHAPTER 5
MULTI-AGENT COORDINATION FOR HETEROGENEOUS AGENTS

In this chapter, we design a multi-agent coordination algorithm, Heterogeneous Policy Networks (HetNet) [7], to learn efficient and diverse communication models for coordinating cooperative heterogeneous teams. Utilizing our novel formulation with multi-agent reinforcement learning, we see that we can utilize simulation to produce high-performance collaborative policies for heterogeneous teammates that may have different capabilities and/or access to different sensory information.

5.1 Introduction

High-performing human teams benefit from communication to build and maintain shared mental models to improve team effectiveness [25, 176]. Information sharing is key in building team cognition, and enables teammates to cooperate to successfully achieve shared goals [24, 176]. Typical communication patterns across human teams widely differ based on the task or role the human assumes [29]. The field of MARL [77] has sought to develop agents that autonomously learn coordination and communication strategies to emulate high-performing human-human teams [83, 177, 178, 167, 168]. Yet, these approaches have fallen short in properly modeling heterogeneity and communication overhead in teaming [82, 55, 179, 180].

Heterogeneity in robots' design characteristics and their roles are introduced to leverage the relative merits of different agents and their capabilities [181, 182, 183] We define a heterogeneous robot team as a group of cooperative agents that are capable of performing different tasks and may have access to different sensory information. We categorize agents with similar state, action, and observation spaces in the same *class*. In such a heterogeneous setting, communicating is not straightfor-

ward as agents do not speak the same "language"; we consider scenarios in which agents have different action-spaces and observation inputs from the environment (i.e., due to different sensors) or may not even have access to any observation input (i.e., lack of sensors, broken or low-quality sensors). The dependency generated via sensor-lax or sensor-void agents on agents with strong sensing capabilities makes efficient communication protocols for cooperation a requirement rather than an additional modeling technique for performance improvement.

While MARL researchers have increasingly focused on developing computational models of team communication [6], most of these prior frameworks fail to explicitly model the heterogeneity of *composite teams* and fail to explicitly quantify and reduce the team's communication overhead to support decentralized, bandwidth-limited teaming. We define a composite team as a group of heterogeneous agents that perform different tasks according to their respective capabilities while their tasks are co-dependent on accomplishing an overarching mission [184, 185, 182]. Agents in a composite team can inherently have different state, action, and observation spaces and yet, must still communicate essential information. Without a proper model for teaming, heterogeneous agents will not be able to reason about the heterogeneity in their team and share information accordingly to achieve team cognition. Therefore, communication may become unhelpful and deteriorate the MARL performance. More recent prior work, such as MAGIC [6], utilized centralized *schedulers* and focused on communication efficiency to achieve improved high team performance. In this work, we intend to push the boundaries beyond this goal and seek to significantly reduce the bandwidth needs for communication to minimize communication overhead and facilitate practical implementation of our framework by designing a *decentralized* execution paradigm.

Inspired by heterogeneous communication patterns across human teams, we propose Heterogeneous Policy Networks (HetNet) to learn efficient and diverse

communication models for coordinating cooperative heterogeneous robot teams. The key to our approach is the design of an end-to-end communication learning model with a differentiable encoder-decoder channel to account for the heterogeneity of inter-class messages, "translating" the encoded messages into a shared, intermediate language among agents of a composite team. Our empirical validation shows that HetNet's novel graph-based architecture achieves a new SOTA in learning emergent cooperative behaviors in complex, heterogeneous domains. HetNet achieves this result while also reducing communication overhead through intelligent message binarization, compressing the number of communicated bits needed by more than 200× per round of communication over the best performing baseline.

Contributions:

1. We develop a novel, end-to-end heterogeneous graph-attention architecture for MARL that facilitates learning efficient, heterogeneous communication protocols among cooperating agents to accomplish a shared task.

2. We design a differentiable encoder-decoder communication channel to learn efficient binary representations of states as an intermediate language among agents of different types to improve their cooperativity. Our binarized communication model achieves 200× reduction in the number of communicated bits per round of communication over baselines while also setting a new SOTA in team performance.

3. We develop Multi-Agent Heterogeneous Actor-Critic (MAHAC) to learn *class-wise* cooperation policies in composite robot teams. Our results show the per-class critic structure achieves better performance over a centralized critic while having fewer model parameters than a per-agent critic.

4. We present empirical evidence that show HetNet is robust to varying band-

width limitations and team compositions, setting a new SOTA in learning emergent cooperative policies by achieving at least an 8.1% to 434.7% performance improvement over baselines and across domains.

5.2 Problem Formulation

Founding on a standard Partially Observable MDP (POMDP) [186], we formulate a new problem setup termed as Multi-Agent Heterogeneous POMDP (MAH-POMDP), which can be represented by a 9-tuple $\langle C, \mathcal{N}, \{\mathcal{S}^{(i)}\}_{i\in C}, \{\mathcal{A}^{(i)}\}_{i\in C}, \{\Omega^{(i)}\}_{i\in C}, \{\mathcal{O}^{i}\}_{i\in C}, r, \mathcal{T}, \gamma\rangle$. C is set of all available agent classes in the composite robot team and the index $i \in C$ shows the agent class. $\mathcal{N} = \sum_{\langle i\in C\rangle} N^{(i)}$ is the total number of collaborating agents where $N^{(i)}$ represents the number of agents in class i. $\{\mathcal{S}^{(i)}\}_{i\in C}$ is a discrete joint set of state-spaces. For each class-dependent state-space, $\mathcal{S}^{(i)}$, we have $\mathcal{S}^{(i)} = \left[s_t^{i_1}, s_t^{i_2}, \cdots, s_t^{i_{N^{(i)}}}\right]$, where $s_t^{i_j}$ represents the state-vector of agent j of the i-th class, at time t. $\{\mathcal{A}^{(i)}\}_{i\in C}$, is a discrete joint set of action-spaces. For each state-dependent action-space, $\mathcal{A}^{(i)}$, we have $\mathcal{A}^{(i)} = \left[a_t^{i_1}, a_t^{i_2}, \cdots, a_t^{i_{N^{(i)}}}\right]$, forming the joint action-vector of agents of class i at time t. $\{\Omega^{(i)}\}_{i\in C}$ is similarly defined as the joint set of observation-spaces, including class-specific observations. $\gamma \in [0, 1)$ is the temporal discount factor for each unit of time and $\mathcal{T}$ is the state transition probability density function.

At each timestep, t, each agent, j, of the i-th class can receive (if the observation input is enabled for class i) a partial observation $o_t^{i_j} \in \Omega^{(i)}$ according to some class-specific observation function $\{\mathcal{O}^{(i)}\}_{i\in C} : o_t^{i_j} \sim \mathcal{O}^{(i)}(\cdot|\bar{s})$. If the environment observation is not available for agents of class i, agents in the respective class will not receive any input from the environment (e.g., lack of sensory inputs). Regardless of receiving an observation or not, at each time, t, each agent, j, of class i, takes an action, $a_t^{i_j}$, forming a joint action vector $\bar{a} = \left(a_t^{1_1}, a_t^{1_2}, \cdots, a_t^{i_1}, \cdots, a_t^{i_j}\right)$. When agents take the joint action $\bar{a}$, in the joint state $\bar{s}$ and depending on the next joint-state, they receive an

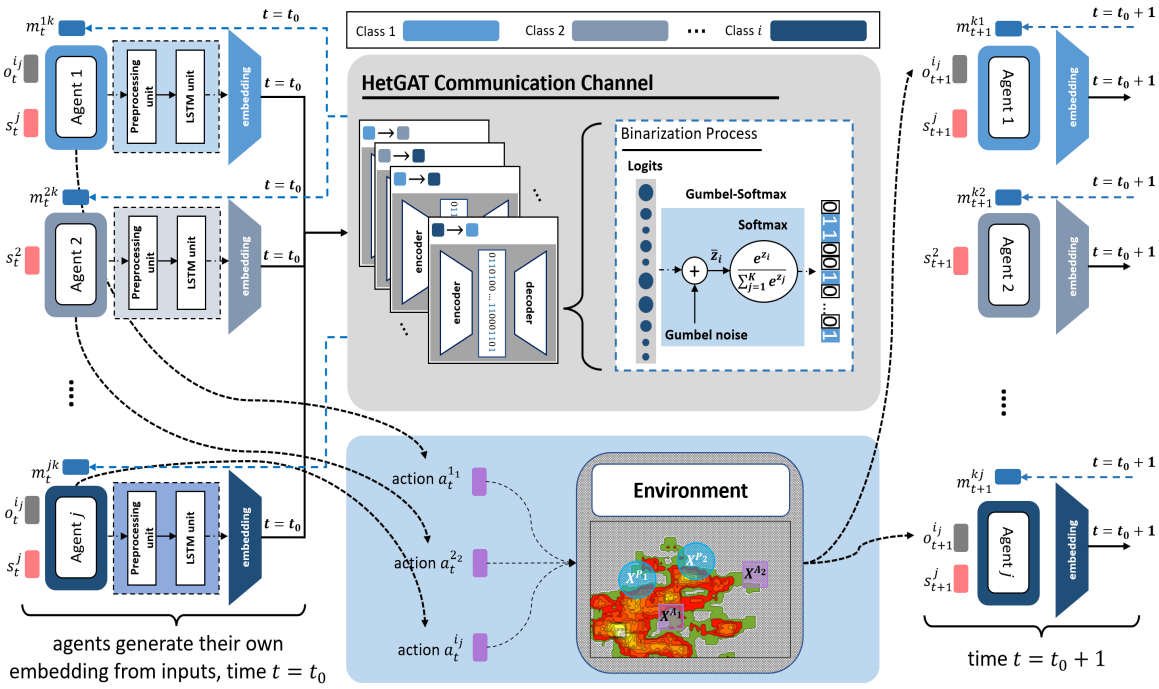


Figure 5.1: Overview of our multi-agent heterogeneous attentional communication architecture in a CTDE paradigm. At each time point $t = t_0$, each agent j of class i generates a local embedding from its own inputs, by passing its input data through class-specific preprocessing units (i.e., a CNN or a fully-connected NN) and an LSTM cell. Each agent then sends the embedding to a class-specific encoder-decoder networks to generate a binarized message, m_t^{jk}, from its local neighbor k. The message information is decoded and leveraged by the receiving agent to compute the action probabilities as its policy output.

immediate reward, $r(\bar{s}, \bar{a}) \in \mathbb{R}$, shared by all agents and regardless of their classes. Our objective is to learn optimal policies per existing agent-class to solve the MAH-POMDP by maximizing the total expected, discounted reward accumulated by agents over an infinite horizon, i.e., $\arg\max_{\pi(\bar{s})\in\Pi} \mathbb{E}_{\pi(\bar{s})}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | \pi(\bar{s})\right]$.

5.3 Method

In this section, we first present an overview of the communication problems and constraints considered in our work. We then describe how to construct a heterogeneous graph given a problem state and present the building block layer, which we refer to as Heterogeneous Graph-Attention (HetGAT) layer, and develop a bi-

narized encoder-decoder communication channel to account for the heterogeneity of messages passed among agents. Eventually, we cover the logistics of utilizing HetGAT layers to assemble our heterogeneous policy network, HetNet, of arbitrary depth.

5.3.1 Communication Problem Overview

In this work, we are concerned with the problem of coordinating a robot team via fostering direct communication among interacting agents. We consider MARL problems wherein multiple agents interact in a single environment to accomplish a task which is of a cooperative nature. We are particularly interested in scenarios in which the agents are heterogeneous in their capabilities, meaning agents can have different state, action and observation spaces in forming a composite team. To collaborate effectively, agents must share messages that express their observations under a Centralized Training and Distributed Execution (CTDE) paradigm [79, 88].

In learning an end-to-end communication model, we take a series of problems and constraints into consideration: (1) heterogeneous messages, where agents of different classes have different action and observation spaces, resulting in different interpretations of sent/received messages; (2) Attentional and scalable communication protocols, such that agents incorporate attention coefficients depending on the agent/class they are communicating with for coordinating with teammates in any arbitrary team sizes; (3) Learning communication models for Low-Size, -Weight, and -Power (Low-SWAP) systems, where due to limited communication bandwidth, agents must learn to communicate in a highly efficient shared intermediate "language" (e.g., limited-length binarized messages); (4) Limited-range communications, where agents can only exchange messages when they are within a close proximity.

5.3.2 Heterogeneous Communication Model

GNNs previously used in MARL operate on homogeneous graphs to learn a universal feature update and communication scheme for all agents [95, 91, 54, 6], which fails to explicitly model the heterogeneity among agents. We instead cast the cooperative MARL problem into a heterogeneous graph structure, and propose a novel heterogeneous graph-attention network capable of learning diverse communication strategies based on agent classes. Compared to homogeneous graphs, a heterogeneous graph can have nodes and edges of different types that can have different types of attributes. This advantage greatly increases a graph's expressivity and enables straightforward modeling of complicated, composite teams.

Given our MAH-POMDP formulation in section 5.2, we directly model each agent class $i \in \mathcal{C}$ as a unique node type. This approach allows agents to have different types of state-space content, $\mathcal{S}^{(i)}$, as input features according to their classes, as well as enabling different types of action spaces, $\mathcal{A}^{(i)}$. Communication channels between agents are modeled as directed edges connecting the corresponding agent nodes. When two agents move to a close proximity of each other such that those agents fall within communication range, we add bidirectional edges to allow message passing between them. We use different edge types to model different class combinations of the sender and receiver agents to allow for learning heterogeneous communication protocols and intermediate representations.

To form our novel architecture for modeling heterogeneous interactions, we add a State Embedding Node (SEN) into the heterogeneous graph to train a critic network. SEN serves as a central node where we aggregate all the important meta-data from the MARL environment (i.e., number of agents, $\mathcal{N}$, world size, current time step, etc.) and use the embeddings for critic training. The SEN forms a *one-way* connection to the agent nodes (i.e., from an agent to the SEN) to receive messages from them during training. The SEN's learned embeddings are used as

input of a critic network consisting of one Fully-Connected (FC) layer for state-dependent value estimation. We note that, since there are no edges pointing from the SEN to any agent nodes, during the execution phase, the SEN can be safely removed without affecting an agent's own policy output, which complies with our underlying CTDE paradigm.

Accordingly, we present an overview of our multi-agent heterogeneous attentional communication architecture in Figure 5.1. At each time, t, the features of each agent (i.e., each node of the heterogeneous graph) are generated through a class-specific feature preprocessor. We utilize separate modules to preprocess an agent's state-vector, $s_t^{i_j}$, and observation, $o_t^{i_j}$, since depending on the agent's class, the environment observation input may not be available (e.g., an action agent in a perception-action composite team). Each preprocecssing module contains one CNN or a fully-connected unit followed by one LSTM cell to enable reasoning about temporal information. As shown in Figure 5.1, the generated embeddings are then passed into a HetGAT communication channel including a class-specific encoder-decoder network and a Gumbel-Softmax [172] unit to generate a binarized message, m_t, for an agent, j.

5.3.3 Binarized Communication Channels

The feature update process in a HetGAT layer is conducted in two steps: *per-edge-type* message passing followed by *per-node-type* feature reduction. When modeling multi-agent teams, we reformulate the computation process into two phases: a *sender* phase and a *receiver* phase. Figure 5.2 shows the computation flow during the sender and receiver phases for an agent, j, of class i.

During the sender phase, the agent, j, of class $i \in C$, processes its input feature, h_j, using a class-specific weight matrix, $\omega_i \in \mathbb{R}^{d' \times d}$, where d and d' are the input and output feature dimensions, respectively. The agent also transforms h_j into the

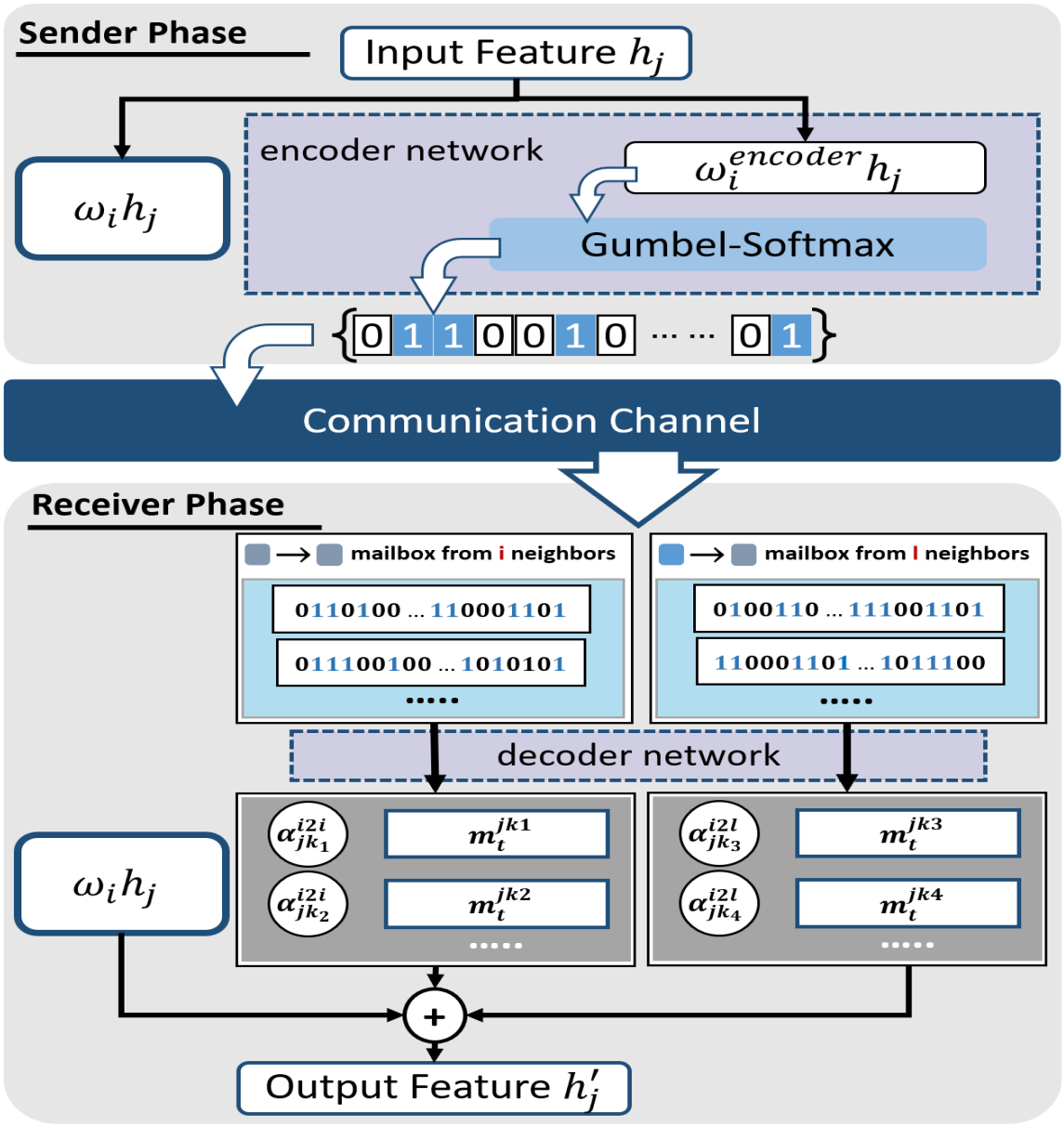


Figure 5.2: The sender and receiver phases of the feature update process in a HetGAT layer for one agent, j, of class i.

assigned message dimension using a class-specific encoder, $\omega_i^{enc} \in \mathbb{R}^{n \times d}$, where n is the communication channel band-width. Next, we leverage a universal binarization process utilizing Gumbel-Softmax to convert the message into 0s and 1s for all classes as an efficient, intermediate language. The binarized message is then sent to neighboring agents.

During the receiver phase, agent, j, of class i, processes all the received messages

using a class-specific decoder, $\omega_i^{dec} \in \mathbb{R}^{d' \times n}$. Next, for each type of communication edge that an agent is connected to, the HetGAT layer computes per-edge-type aggregation result by weighing received messages, along the same edge-type with normalized attention coefficients, $\alpha^{edgeType}$. The aggregation results are then merged with the agent's own transformed embedding, $\omega_i h_j$, to compute the output features. The feature update formula for an agent is shown in Eq. Equation 5.1, where j and k are agent indexes and, $i, l \in C$ are class indexes; such that, $i2l$ is an *edgeType* and means "from class i to class l". m_t^{jk} is the decoded message computed by Eq. Equation 5.2 and, $\Delta_l(j)$ include agent j's neighbors that belong to class l.

$$\text{Class } (i): \quad \bar{h}'_j = \sigma\Big(\omega_i \bar{h}_j + \sum_{l \in C} \sum_{k \in N_l(j)} \alpha_{jk}^{i2l} m_t^{jk}\Big) \tag{5.1}$$

$$m_t^{jk} = \omega_i^{dec}(\text{GumbelSoftmax}(\omega_l^{enc} h_k)) \tag{5.2}$$

Note that we have $l = i$ for intra-class communication. When computing attention coefficients in a heterogeneous graph, we adapt Eq. Equation 3.2 into Eq. Equation 5.3 to account for heterogeneous channels.

$$\alpha_{jk}^{i2l} = \text{softmax}_k\left(\sigma'\left(\bar{W}_{att}^T\left[\omega_i \bar{h}_j \,\|\, \omega_{i2l} \bar{h}_k\right]\right)\right) \tag{5.3}$$

As discussed in subsection 5.3.2, we add an SEN to the graph during centralized training with a state-dependent critic network. The feature update formula of the SEN is shown in Eq. Equation 5.4. Here, feature vectors from all agents are passed to the SEN after being processed with edge-specific weights, $\omega_{edgeType}$. The attention coefficients for the SEN are computed in a similar manner as in Eq. Equation 5.3.

$$\text{SEN}: \quad \bar{h}'_s = \sigma\Big(\omega_s \bar{h}_s + \sum_{i \in C} \sum_{j \in N^{(i)}} \alpha_{sj}^{i2sen} \omega_{i2sen} \bar{h}_j\Big) \tag{5.4}$$

To stabilize the learning process, we adapt the multi-head extension of the attention mechanism [164] to fit our heterogeneous setting. We use L independent HetGAT (sub-)layers to compute node features in parallel and then merge the results by concatenation operation for each multi-head sub-layer in HetNet except for the last layer which employs averaging. As a result, each type of communication channel is split into L independent, parallel sub-channels.

5.3.4 Heterogeneous Policy Network (HetNet)

At each timestep, t, a HetGAT layer corresponds to one round of message exchange between neighboring agents and feature update within each agent. By stacking several HetGAT layers, we construct the Heterogeneous Policy Network (HetNet) model that utilizes multi-round communication to extract high-level embeddings of each agent for decision-making. For the last HetGAT layer in HetNet, we set each agent's output feature dimension to the size of its action-space, specific to its class, i. Then, for each agent node, we add a Softmax layer on top of its output to obtain a probability distribution over actions that can be used for action sampling, resulting in class-wise stochastic policies. Accordingly, the computation process of each agent's policy remains local for distributed execution, and the SEN is no longer needed during execution/testing.

5.4 Training and Execution

5.4.1 Multi-agent Heterogeneous Actor-Critic

We present our modified Multi-Agent Heterogeneous Actor-Critic (MAHAC) framework for learning class-wise coordination policies. We assign one policy per existing class, $\pi^i \in \{\Pi\}^C$, each of which is parametrized by θ^i. The trained actor network on the heterogeneous graph contains one set of learnable weights per agent class, which due to the message-passing nature of GNN updates, can be

distributed to individual agents in the execution phase. Accordingly, in MAHAC, the policy for each class, π^i, is updated by a variant of the basic AC objective (see section 3.4, shown in Eq. Equation 5.5. We leverage an on-policy training paradigm for MAHAC.

$$\nabla_{\theta^i} J(\theta^i) = \frac{1}{N} \sum_{j=1}^{\mathcal{N}^{(i)}} \sum_{t=1}^{T} \nabla_{\theta^i} \log \pi^i \left(\bar{a}_t^{i_j} | \bar{o}_t^{i_j}, \bar{m}_t \right) \left(\left(\sum_{t'=t}^{T} \gamma^{t'-t} r^{i_j}(\bar{s}^{i_j}, \bar{a}^{i_j}) \right) - b(t) \right) \tag{5.5}$$

In Eq. Equation 5.5, $\bar{a}_t^{i_j}$ and $\bar{o}_t^{i_j}$ represent the joint actions taken and joint observations received (if applicable for class i) by agents at time, t. $\bar{m}_t$ represents the message-vector received by agent j from its neighbors. The term $\sum_{t'=t}^{T} \gamma^{t'-t} r^{i_j}(\bar{s}^{i_j}, \bar{a}^{i_j})$ calculates the total discounted future reward from current timestep to end of an episode. Moreover, $b(t)$ is a temporal baseline function leveraged to reduce the variance of the gradient updates in MAHAC. We utilize the value-estimates via our critic network as the baseline function [187][1].

5.4.2 Critic Architecture Design for HetNet

In this section, we propose and assess several MAHAC architectures to investigate the utility of: (1) *fully-centralized* critic, $b(t)$ (i.e., one critic signal for all), (2) *per-class* critics, $b^i(t)$ (i.e., one critic signal per class of agents) and (3) *per-agent* critics, $b^{i_j}(t)$ (i.e., individual critic signals for each agent) to learn class-wise policies.

In the fully-centralized critic implementation for HetNet, we stack an FC layer on top of SEN's output feature for critic prediction of the value estimate. The same predicted critic value is used in the policy gradient update for all agents of all classes. The target value for training the critic output is the average returns (i.e., discounted sum of future rewards) over all agents. Thus, in this architecture one centralized critic network "criticizes" the actions of all agents. Note that

[1] We provide our code at `https://github.com/CORE-Robotics-Lab/HetNet`

this approach still complies with our CTDE paradigm, since the actor network is implemented on a GNN structure.

For the per-class critic implementation, we split the critic head into one critic head per existing agent class to separate the critic estimation for different types of agents. The critic split is done while the critic is estimated based on a class-specific SEN's output feature. During training, the target value for each class of critic output is the average returns over the same class of agents. Algorithm Algorithm 2 provides a pseudocode to train HetNet with the per-class critic architecture.

In our per-agent critic implementation for HetNet, the critic network outputs one critic value for each agent. This is achieved by concatenating the SEN's output feature with each agent node's output embedding to serve as the input of class-specific critic heads. The per-agent critic estimation is used for each agent's policy update where the target value for training is the returns of that agent.

5.5 Empirical Evaluation

5.5.1 Evaluation Environments

We evaluate the utility of HetNet against several baselines in three cooperative MARL domains (a homogeneous and two heterogeneous) that require learning collaborative behaviors.

Predator-Prey (PP) [51] – For the homogeneous domain, we adopt the Predator-Prey (PP) [51] in which the goal is for N predator agents with limited vision to find a stationary prey and move to its location. The agents in this domain are homogeneous in their state, observation, and action spaces, and thus, all agents are of the same *class*.

Predator-Capture-Prey (PCP) – For the first heterogeneous domain, we modify the PP to create a new environment, which we refer to as Predator-Capture-Prey (PCP), to include a composite team. In PCP, we have two classes of agents: *predator*

Algorithm 2 The *Per-class* training procedure for HetNet.

Input: Agent classes, $i \in C$, number of agents in each class, $\mathcal{N}^{(i)}$, number of episodes per epoch K, maximum allowed steps for each episode, T, learning rate, η.
Initialize: Per-class policy parameters $\{\theta^i\}$ for $\{\pi^i\}$ and per-class critic parameters $\{\phi^i\}$ for $\{V^i\}$, $i \in C$
while not converged **do**
  Sample a random environment instance
  for $k = 1$ to K **do**
    Get initial observations $\{o_1^{1_1}, o_1^{1_2}, ..., o_1^{i_j}\}$, $i \in C$, $j \in \mathcal{N}^{(i)}$
    for $t = 1$ to T **do**
      Perform message passing and feature reduction
      Store critic predictions $\{V_t^i\}$, $i \in C$
      Sample actions: $a_t^{i_j} \sim \pi^i(* \mid o_t^{i_j})$, $i \in C$, $j \in \mathcal{N}^{(i)}$
      Step through environment using $\{a_t^{1_1}, a_t^{1_2}, \cdots, a_t^{i_j}\}$, receive next observations and rewards: $\{o_{t+1}^{1_1}, o_{t+1}^{1_2}, ..., o_{t+1}^{i_j}\}$, $\{r_t^{1_1}, r_t^{1_2}, ..., r_t^{i_j}\}$
      if *environment_solved* **then**: Terminate early **end if**
    end for
  end for
  for $i \in C$ **do**
    Compute rewards-to-go R_t^i and GAE advantages A_t^i
    $\nabla J(\theta^i) = \frac{1}{N} \sum_{j=1}^{\mathcal{N}^{(i)}} \sum_{t=1}^{T} \nabla \log \pi^i \left(a_t^{i_j} | o_t^{i_j}\right) A_t^i$
    Critic loss: $L(V^i) = \frac{1}{N} \sum_{j=1}^{\mathcal{N}^{(i)}} \sum_{t=1}^{T} \left(V_t^i - R_t^i\right)^2$
    Joint update: $\theta^i = \theta^i + \eta \nabla J(\theta^i)$, $\phi^i = \phi^i - \eta \nabla L(V^i)$
  end for
end while

agents and *capture* agents. The first class of agent, called the *predator* agents, have the goal of discovering the prey and have an action-space of dimension five, including cardinal movements and a null (stay) action. *Predator* agents have an observation space similar to the agents in PP domain. The second class of agents, called the *capture* agents, have the objective of locating the prey *and* capturing it. Capture agents differ from the predator agents in both their observation and their action spaces. Capture agents do not receive any observation inputs from the environment (i.e., no scanning sensors) and have an additional action of *capture-prey* in their action-space. This additional action must be used at a prey's location to capture

the prey. Note that this domain is an explicit example of the perception-action composite teams, as introduced in Section 6.1. An episode is deemed successful once all agents have completed their class-specific objectives. Each *predator* agent is penalized with -0.05 reward every timestep until it has discovered the prey. Each *capture* agent is also penalized with -0.05 every timestep until it has captured (i.e., find the prey and then capture it) the prey. Note the difference in reward scheme, a *capture* agent may have discovered the prey but will receive a negative reward until the *capture-action* is utilized. We utilize PCP as a testbed for several test heterogeneous interactions. In our head-to-head evaluation against baselines, we utilize a problem with two *predator* agents and one *capture* agent within a 5x5 grid. We set the maximum steps for an episode to be 80.

FireCommander (FC) [188] – In the second heterogeneous domain, the FireCommander [188], two classes of *perception* and *action* agents must collaborate as a composite team to extinguish a propagating firespot. FireCommander can be categorized as a strategic game, in which a composite team of robots (i.e., UAVs) must collaboratively find hidden areas of propagating wildfire and extinguish the fire in such areas as fast as possible. At each timestep, the firespot propagates to a new location according to the FARSITE [189] model, while the previous location is still on fire. All firespots are initially hidden to agents and need to be discovered before being extinguished. The robot team in FC is composed of two classes of agents: (1) *perception* agents (class P), which can only sense the environment and detect areas of fire and, (2) *action* agents (class A), which can only manipulate the environment by extinguishing a firespot which has already been detected by class P agents. Neither class P, nor class A agents are capable of accomplishing the task on their own, and therefore must communicate and collaborate. Under the notations in our problem formulation in section 5.2, we have $C = \{\mathrm{P}, \mathrm{A}\}$ where, $\mathcal{A}^{(\mathrm{P})} = \{1, 2, \cdots, 4\}$ representing the four primitive motions and $\mathcal{A}^{(\mathrm{A})} = \{1, \cdots, 5\}$, representing the four

Table 5.1: Reported results are Mean (± Standard Error (SE)) from 50 evaluation trials. For all tests, the final training policy at convergence is used for each method. As shown, HetNet outperforms all baselines in all three domains.

Method	Homogeneous Domain (PP)		Heterogeneous Domain #1 (PCP)		Heterogeneous Domain #2 (FC)	
	Avg. Cumulative $\mathcal{R}$	Avg. Steps Taken	Avg. Cumulative $\mathcal{R}$	Avg. Steps Taken	Avg. Cumulative $\mathcal{R}$	Avg. Steps Taken
TarMAC [55]	-0.563 ± 0.030	18.4 ± 0.46	-0.548 ± 0.031	17.0 ± 0.80	-109.2 ± 6.26	248.1 ± 6.97
IC3Net [51]	-0.342 ± 0.015	9.69 ± 0.26	-0.411 ± 0.019	11.5 ± 0.37	-187.2 ± 0.79	276.0 ± 5.51
CommNet [52]	-0.336 ± 0.012	8.97 ± 0.25	-0.394 ± 0.019	11.3 ± 0.34	-253.2 ± 1.01	292.7 ± 3.07
MAGIC [6]	-0.386 ± 0.024	10.6 ± 0.50	-0.394 ± 0.017	10.8 ± 0.45	-267.6 ± 10.9	298.1 ± 23.3
HetNet [Ours]	**-0.232 ± 0.010**	**8.30 ± 0.25**	**-0.364 ± 0.017**	**9.98 ± 0.36**	**-9.862 ± 2.77**	**46.40 ± 2.90**

primitive motions and an extra action which corresponds to extinguishing fire by dousing water. Agents of class P are equipped with fire detection sensors and can observe the environment, receiving an input vector of length 29 for each grid within their FOV. Agents of class A, do not receive any observation from the environment. The reward scheme in this domain includes a small temporal penalty of -0.1 per timestep for all agents, a false water-drop penalty of -0.1 for *action* agents, a -0.1 penalty per new firespot for all agents, and a positive reward of +10 for all agents per each extinguished firespot. In our head-to-head evaluation against baselines, we utilize a problem with two *perception* agents and one *action* agent within a 5x5 grid and one initial firespot that propagates to a new location at each timestep, leaving the previous grid on fire. We set the maximum steps for an episode to be 300. An episode of the game is marked as successful only if all the active firespots within the map are discovered and extinguished.

5.5.2 Baselines

We benchmark two variants of our framework, i.e. HetNet-Binary and HetNet-Real, against four end-to-end communicative MARL baselines: (1) CommNet [52], (2) IC3Net [51], (3) TarMAC [55] and, (4) MAGIC [6]. For our HetNet-Real variant, we remove the binarization process (i.e., Gumbel-Softmax) and the encoder-decoder network from the communication channel. Accordingly, agents directly

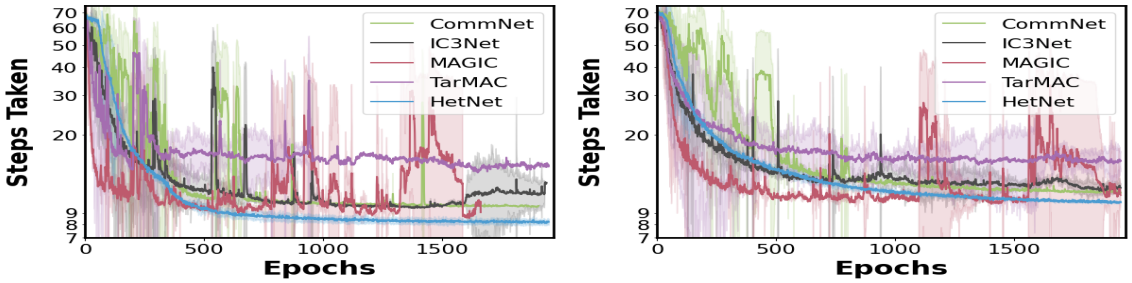
send their generated embeddings (i.e., the LSTM cell output) to a class-specific communication edge in HetGAT layers. The HetNet-Real utilizes continuous, agent-specific embeddings to generate limited-length, real-valued numbers which allow for greater expressivity in the message-space. The real-valued numbers require more communication band-width and higher memory storage as compared to HetNet-Binary (see subsection 5.5.3). We note that, for all four baselines, i.e. CommNet [52], IC3Net [51], TarMAC [55] and, MAGIC [6], we directly pulled the respective authors' publicly available code-bases and hyperparameters for training. Note that we observed some performance discrepancies while directly using MAGIC's public repository (i.e., `github.com/MAGIC`).

5.5.3 Results, Ablation Studies, and Discussion

Here, we empirically validate the performance of our frameworks, across homogeneous and heterogeneous teaming domains and against the introduced baselines. Next, we present an ablation study to investigate the required communication overhead for each method (subsubsection 5.5.3). We then present evidence to support the effects of communication on collaboration performance (subsubsection 5.5.3) as well as to determine the sensitivity of HetNet to key variables such as number of agents (subsubsection 5.5.3). Additionally, we investigate the effects of the critic structures proposed in subsection 5.4.2 on HetNet's performance (subsubsection 5.5.3).

Baseline Comparison

Figure 5.3 depicts the average steps taken (± standard error) by each method across episodes as training proceeds in PP and PCP domains. In both domains, PP and PCP, HetNet outperforms all baselines by converging to a more efficient coordination policy (i.e., fewer steps taken). We also tested the learned coordination policies



(a) Homogeneous Domain (PP) (b) Heterogeneous Domain (PCP)

Figure 5.3: Average steps taken (± SE) by each method across episodes and three different random seeds as training proceeds. HetNet outperforms all baselines in both domains.

at convergence by each of the baselines in PP, PCP and FC domains. The results of this test are presented in Table 5.1 where the reported results are mean (± Standard Error (SE)) from 50 evaluation trials with different random-seed initializations. As shown, HetNet outperforms all baselines in all three domains. Additionally, in the same experiment, the coordination policy learned by our HetNet-Binary with 64-bits message dimensionality achieved 9.90±0.58 average steps taken in the PCP domain; showing better performance than all baselines while significantly compressing the communication bandwidth (see Figure 5.4). The heterogeneous policies learned by our model set the SOTA for learning challenging cooperative behaviors for composite teams.

Ablation Study #1: Communication Bandwidth

In this experiment, we compute the Communication Bandwidth (CB) for each baseline as the number of bits required to communicate messages per round of communication during evaluation (i.e., converged policies deployed for test). As shown in Figure 5.4, HetNet facilitates binarized communication among agents which requires significantly less CB as compared to real-valued baselines (i.e., one bit per binary value vs. 64 bits in single-precision floating-point format [190]). HetNet-Binary with 64 and 32-bits messages, respectively, achieve more than 100×

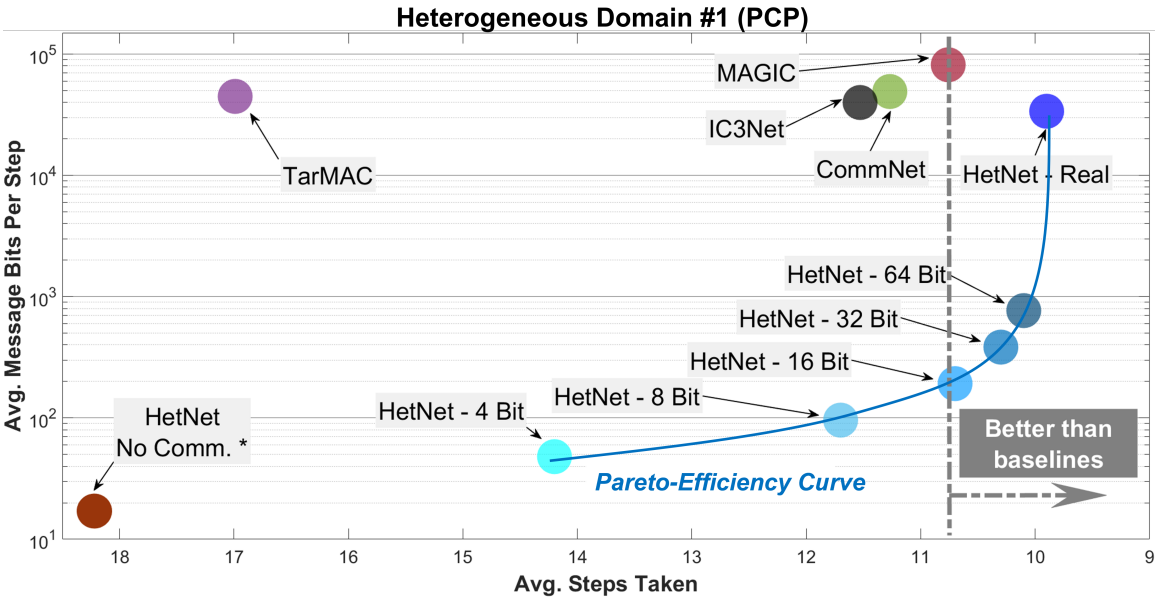
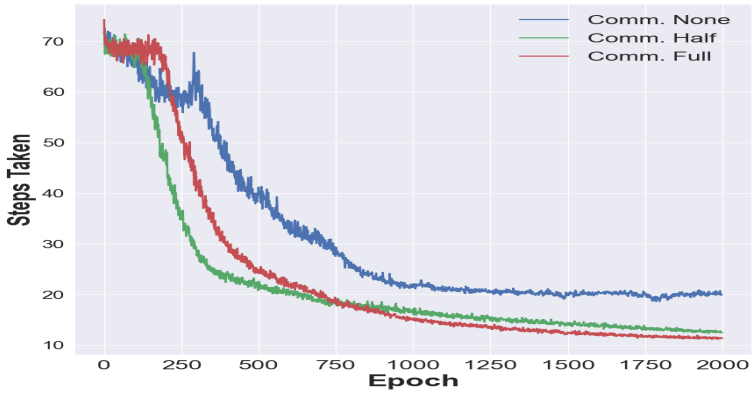


Figure 5.4: Communicated bits per round of communication vs. performance in PCP for different methods. HetNet facilitates binarized messages among agents which requires significantly less CB as compared to real-valued baselines.

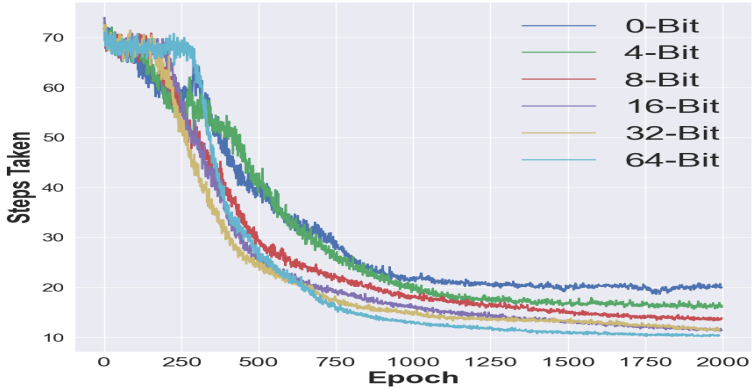
and 200× lower CB while showing better performance than real-valued baselines.

Ablation Study #2: Effects of Communication

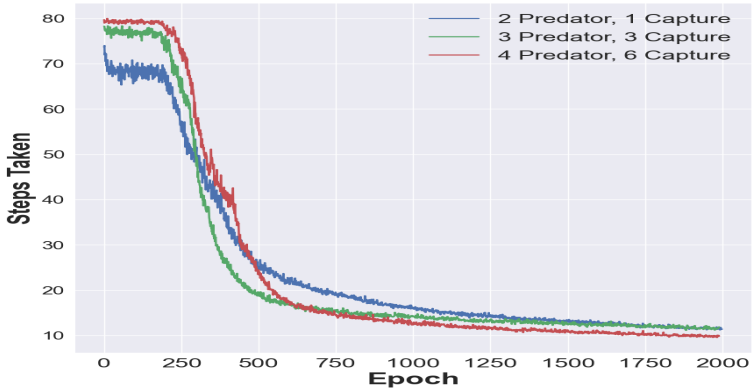
We assess the impact of the communication on cooperation performance of the composite team. We present two experiments in the PCP domain for comparing HetNet's performance: (1) with *Full*, *Half* and *No* communication among agents and (2) with different binary message dimensions (number of bits). As depicted in Figure 5.5a, HetNet performs significantly better with full communication while the performance drop for half-communication (i.e., limited range) is not considerable. As such, the results show that our model, HetNet has robustness to degradation in communication range. Additionally, as shown in Figure 5.5b, a gradual degradation in performance is observed by decreasing message dimensionality rather than a sharp drop-off. HetNet's performance improves with longer messages as the learned intermediate language will have greater expressivity.



(a) Communication range.



(b) Message dimensionality.



(c) Scalability to num. of agents.

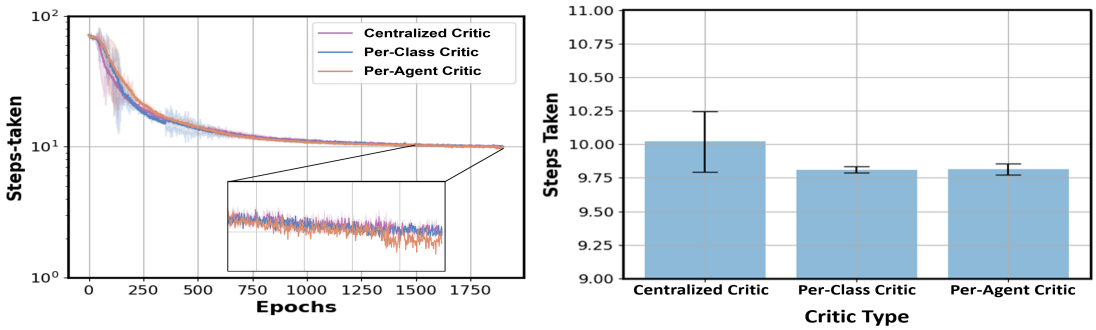
Figure 5.5: Analyzing HetNet's performance with and without communication (Figure 5.5a) and across different binary message dimensions (Figure 5.5b) in the PCP domain. Communication policy learned by HetNet improves the cooperativity among agents and the performance improves with larger message sizes. Figure 5.5c depicts results for analyzing HetNet's ability to scale to different number of agents. As shown, HetNet-Binary can successfully scale to different sizes of the composite team.

Ablation Study #3: Scalability to Number of Agents in the Composite Team

In this experiment, we evaluated the scalability of our HetNet-Binary to different number of agents in the composite team. Specifically, we tested HetNet-Binary in PCP domain with (2P, 1C), (3P, 3C) and (4P, 6C) team compositions, where *P* and *C* represent *predator* and *capture* agents, to evaluate the scalability to different team sizes. The results of this experiment are presented in Figure 5.5c. As shown, HetNet's GNN-based architecture can successfully scale to different combinations of the composite team by approximately converging at the same rates.

Ablation Study #4: Effects of the Critic Structures

Finally, we investigate the utility and performance of the three critic structures proposed in subsection 5.4.2 on HetNet's performance in the PCP domain. We utilized our HetNet-Real variant for this experiment. Figure 5.6a shows the learning curves during training for centralized, per-class, and per-agent critic structures in the PCP domain. The test results for coordination policies learned by each of the critic architectures are presented in Figure 5.6b, showing the average number of steps taken to win the game by deploying the converged policies by each critic design. As depicted, HetNet-Real shows similar performance with per-class and per-agent critics, both having better results than the centralized critic, decreasing the number of steps of episode completion by 0.20 ($10.01 \rightarrow 9.81$). The performance benefit can be attributed to the ability to utilize individual and class-wise rewards, both of which help to capture the heterogeneity in the received feedback from the environment.



(a) Training Performance. (b) Converged Performance.

Figure 5.6: Learning curves during training as well as the test results (average number of steps taken) for final policies learned by centralized, per-class and per-agent critic architectures in the PCP domain.

5.6 Conclusion

Motivated by the diverse communication patterns across collaborating human teams, we present a communicative, cooperative MARL framework for learning heterogeneous cooperation policies among agents of a *composite* team. We propose Heterogeneous Policy Network (HetNet), a heterogeneous graph-attention based architecture, and introduce the Multi-Agent Heterogeneous Actor-Critic (MAHAC) learning paradigm for training HetNet to learn class-wise cooperation policies. We push the boundaries beyond performance considerations as in prior work by equipping HetNet with a binarized encoder-decoder communication channel to facilitate learning a new and highly efficient encoded language for heterogeneous communication. We empirically show HetNet's superior performance against several baselines in learning both homogeneous and heterogeneous cooperative policies. We provide empirical evidence that show: (1) our binarized model achieves more than 200× reduction in communication overhead (i.e., message bits) per round of communication while also outperforming baselines in performance, (2) HetNet is robust to varying bandwidth limitations and team compositions.

In later extended work, we create a scalable MARL architecture that can support variable-length input representations [191] and an updated, more sample-efficient

learning algorithm based on MA-PPO [192] to support heterogeneous robot team coordination. We provide several enhancements to the aforementioned, including 1) the development of a novel Multi-Agent Heterogeneous Proximal Policy Optimization (MAH-PPO) algorithm to support sample-efficient learning of coordination policies for heterogeneous robots, 2) an architecture enhancement to support transferability to different environment sizes, resulting in non-parametricity in the number of agents, and 3) a noise-degradation channel with three different paradigms of communication loss, allowing us to test the robustness of our algorithm. Overall, we find that HetNet can outperform baselines across several domains, scale to domain configurations in which baselines fail to learn, and develop robustness to noise-degraded communication channels.

CHAPTER 6
INFERRING DECISION-MAKING BEHAVIOR ACROSS HETEROGENEOUS USERS

In this chapter, we begin to enable the previously found vital characteristics in agent-agent collaboration, *utilizing communication* and *accounting for heterogeneity*, to human-robot collaboration. Here, we specifically look to develop an *interpretable* architecture that can infer latent characteristics of human behavior (i.e., capture diversity). We utilize resource coordination as a test domain, as schedulers maintain a wide variety of heuristics (developed through experience and apprenticeship) across their decision-making.

6.1 Introduction

Coordinating resources in time and space is a challenging and costly problem worldwide, affecting everything from the medical supplies we need to fight pandemics to the food on our tables. The manufacturing and healthcare industries account for a total of $35 trillion [193] and $8.1 trillion [194] USD, respectively. In manufacturing, scheduling workers – whether they be humans or robots – to complete a set of tasks in a shared space with upper- and lower-bound temporal constraints (i.e., deadline and wait constraints) is an NP-hard optimization problem [195], typically approaching computational intractability for real-world problems of interest.

Human domain experts efficiently, if sub-optimally, solve these NP-hard problems to coordinate resources using heterogeneous rules-of-thumb and strategies honed over decades of apprenticeship, creating unique heuristics depending on experts' varied experiences and personal preferences [56, 57]. Each expert has her

own strategies, and it is common for factories and hospital wards to be run completely differently – yet effectively – across different shifts based upon the person in charge of coordinating the workers' activities [196, 197, 198]. The challenge we pose is to develop new *apprenticeship learning* techniques for capturing these heterogeneous rules-of-thumb in order to scale beyond the power of a single expert. However, such heterogeneity is not readily handled by traditional apprenticeship learning approaches that assume demonstrator homogeneity. A canonical example of this limitation is of human drivers teaching an autonomous car to pass a slower-moving car, where some drivers prefer to pass on the left and others on the right. Apprenticeship learning approaches assuming homogeneous demonstrations either fit the mean (i.e., driving straight into the car ahead of you) or fit a single mode (i.e., only pass to the left).

The field of apprenticeship learning has recently begun working to relax the assumption of homogeneous demonstrations by explicitly capturing modes in heterogeneous human demonstrations [62, 64, 63, 199, 105]. One such approach, InfoGAIL [62], uses a generative adversarial setting with variational inference to learn discrete, latent codes to describe multi-modal decision-making. However, InfoGAIL requires access to an environment simulator, relies on a ground-truth reward signal, and is ill-suited to reasoning about resource scheduling and optimization problems, as we show in section 6.4. Further, modern imitation learning frameworks lack interpretability, hampering adoption in safety-critical and legally-regulated domains [34, 35, 36, 37].

Contributions – Overcoming these key limitations of prior work, we develop a novel, data-efficient apprenticeship learning framework for learning from heterogeneous scheduling demonstration. The key to our approach is a neural network architecture that serves as a function approximator specifically designed for sparsity to afford easy "discretization" into a Boolean decision tree after training as

well as the ability to leverage variational inference to tease out each demonstrator's unique decision-making criteria. In section 6.4, we empirically validate that our approach, "Personalized Neural Trees", outperforms baselines even after discretization into a decision tree. Our contributions are as follows:

1. Formulate a personalized and interpretable apprenticeship scheduling framework for heterogeneous LfD that outperforms prior state-of-the-art approaches on both synthetic and real-world data across several domains (+51% and +11%, respectively) through the use of personalized embeddings without constraining the number of demonstrator types.

2. Develop a methodology for converting a personalized neural tree into an interpretable representation that directly translates decision-making behavior. Our discretized trees also outperform previous benchmarks on synthetic and real-world data across several domains.

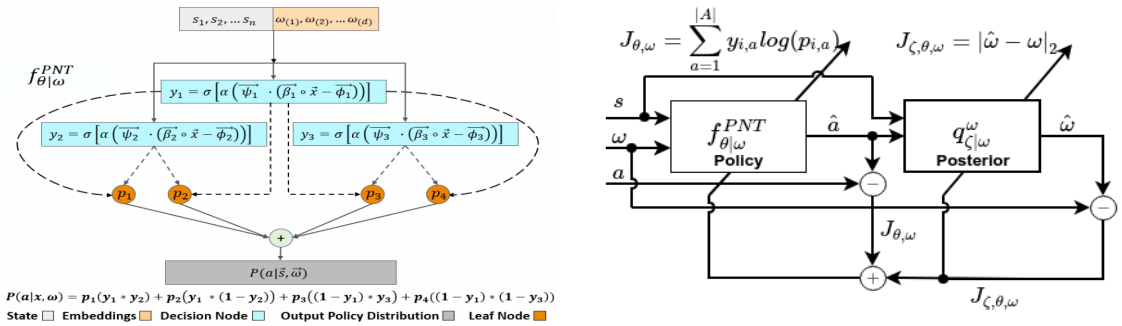
3. Conduct a user study that shows our post-processed interpretable trees are more interpretable ($p < 0.05$), easier to simulate ($p < 0.01$), and quicker to validate ($p < 0.01$) than their black box neural network counterparts.

6.2 Personalized and Interpretable Neural Trees

In this section, we present our apprenticeship framework that utilizes person-specific (personalized) embeddings, learned through backpropagation, which enables the apprenticeship learner to automatically adapt to a person's unique characteristics while simultaneously leveraging any homogeneity that exists within the data (e.g., uniform adherence to hard constraints).

6.2.1 Algorithm Overview

To effectively learn from heterogeneous decision-makers, we must capture the homo- and heterogeneity in their demonstrations, allowing us to learn a gen-



(a) PNT Architecture (b) Algorithm Overview

Figure 6.1: The PNT architecture (left) displaying decision nodes, y_i, with evaluation equations, leaf nodes, k, with respective weights p_k, and output equation describing the calculation of the action probability mass function. An overview of our training algorithm (right) displaying the input/output flow of the policy and the posterior alongside their respective update equations.

eral behavior model accompanied by personalized embeddings that fit distinct behavior modalities. We contribute a novel apprenticeship learning model for resource scheduling, "Personalized Neural Trees", by extending DDTs in four important ways: 1) Personalized embeddings, ω, as a latent variable representing person-specific modality (subsection 6.2.2), 2) Variational inference mechanism to maximize the mutual information between the embedding and the modeled decision-maker (subsubsection 6.2.2), 3) Counterfactual reasoning to increase data-efficiency (subsubsection 6.2.2), and 4) Novel feature selector, $\vec{\psi}_i$, for each decision node to enhance interpretability (subsection 6.2.3).

A Personalized Neural Tree (PNT) learns a model, $f^{PNT}_{\theta|\omega} : S \times \Omega \rightarrow [0,1]^{|A|}$, of human demonstrator decision-making policies, where $\theta \in \Theta$ are the policy weights and $\omega \in \Omega$ ($\Omega \subset \mathbb{R}^d$) is the demonstrator-specific personalized embedding of length d, which is a tunable hyperparameter. Here, $\Omega = \{\omega_1, \omega_2, \ldots, \omega_P\}$ represents the set of all demonstrator personalized embeddings. The person-specific features ω identify the latent pattern of thinking for the current decision-maker. We note that the policy weights, $\theta \in \Theta$, are specifically defined as $\Theta = \alpha \times \Psi \times B \times \Phi \times \mathcal{P}$, where α, B, and Φ are the parameters of the decision nodes (subsection 6.2.2), $\mathcal{P}$ are the

leaf parameters (subsection 6.2.2), and Ψ are a new set of parameters we introduce in subsection 6.2.3 to enhance interpretability during post-training discretization.

Alongside learning a demonstrator's decision-making policy $f^{PNT}_{\theta|\omega}$, we introduce (subsubsection 6.2.2) an information theoretic regularization model, similar to [200], to maximize mutual information between latent embeddings, ω, and trajectories, τ, by learning a model, $q^{\omega}_{\zeta|\theta} : S \times [0,1]^{|A|} \rightarrow \mathcal{N}_{\Omega}$ represented by a neural tree (PNT$\backslash\omega$) with weights ζ, that approximates the true posterior $P(\omega|\tau)$. This induces the latent personalized embeddings to capture modality within demonstrator trajectories.

6.2.2 Personalized Neural Tree

We present architecture of $f^{PNT}_{\theta|\omega}$ as shown in Fig. Figure 6.1a. First, a demonstrator-specific embedding (represented by $\omega \in \Omega$) is concatenated with state $\vec{s} \in S$ and routed directly to each decision node as $\vec{x} = [\vec{s}, \vec{\omega}]$). Each decision node in the PNT is conditioned on three differentiable parameters: weights $\vec{\beta} \in B$, comparison values $\vec{\phi} \in \Phi \subset [0,1]^{n+d}$, and selective importance vectors $\vec{\psi} \in \Psi$. When input data $\vec{x}$ is passed to a decision node, i, the data is weighted by $\vec{\beta_i}$ and compared against $\vec{\phi_i}$ as shown in Equation 6.1, where $\circ$ is the Hadamard product. The PNT algorithm uses its selective importance vector $\vec{\psi_i} \in [0,1]^{|\vec{x}|}$ before weighting by α and passing through a sigmoid to decide which "rule" is the most helpful to apply for this node; y_i is the probability of decision node i evaluating to TRUE.

$$y_i = \sigma[\alpha(\vec{\psi_i} \cdot (\vec{\beta_i} \circ \vec{x} - \vec{\phi_i}))] \tag{6.1}$$

subsection 6.2.3 describes how this extension of the original formulation [67] enhances interpretability.

Leaf nodes, k, in the PNT maintain a set of weights over each output class denoted $\vec{p_k} \in \mathcal{P}$. Each decision node, i, along a path from the root to a leaf (i.e., a branch) output probabilities, y_i, for each such decision node. The branch's prob-

abilities are multiplied to produce a joint probability of reaching the leaf in that branch given state $\vec{s}$ and the current demonstrator embedding ω_p. Each leaf, k contains a probability mass function (PMF), $\vec{p}_k$, where $\vec{p}_{k,a}$ is the probability of applying action a for the branch leading to leaf node, k. This probability distribution, $\vec{p}_k$, for leaf, k, is multiplied by its corresponding branches' joint probability. Finally, the products of all leaf vectors with their branch's joint probability is summed to produce the final network output, a PMF for actions given state, s, and embedding, ω_p. An example is shown in Fig. Figure 6.1a complete with an equation summarizing the output.

Maximizing Mutual Information

The parameters, ζ, θ, and ω, are updated via a cross-entropy loss and mutual information maximization loss, as discussed in subsection 6.2.4. Maximizing mutual information encourages ω to correlate with semantic features within the data distribution (i.e., mode discovery) [200, 62]. Yet, maximizing mutual information between the trajectories and latent code, $G(\omega;\tau)$, is intractable as it requires access to the true posterior, $P(\omega|\tau)$. Therefore, researchers employ the evidence lower bound (ELBO) of the mutual information $G(\omega;\tau)$, as shown in Equation 6.2. Maximizing $G(\omega;\tau)$ incentivizes the policy, $f^{PNT}_{\theta|\omega}$, to utilize the latent embedding ω as much as possible.

$$
\begin{aligned}
G(\omega;\tau) &= H(\omega) - H(\omega|\tau) \\
&= \mathbb{E}_{\omega\sim P(\omega), a^t_p\sim f^{PNT}_{\theta|\omega}}[logP(\omega|s^t_p, a^t_p)] + H(\omega) \\
&= \mathbb{E}_{a\sim f^{PNT}_{\theta|\omega}}[D_{KL}(log(P(\omega_p|s^t_p, a^t_p))||log(q^{\omega}_{\zeta|\theta}(s^t_p, a^t_p))) + \mathbb{E}_{\omega\sim P(\omega)}log(q^{\omega}_{\zeta|\theta}(s^t_p, a^t_p))] + H(\omega) \\
&\geq \mathbb{E}_{\omega_p\sim\mathcal{N}(\vec{\mu}_p,\vec{\sigma}^2_p), a\sim f^{PNT}_{\theta|\omega}}[log(q^{\omega}_{\zeta|\theta}(\omega_p|s^t_p, a^t_p))] + H(\omega) = L_G(f^{PNT}_{\theta|\omega}||q^{\omega}_{\zeta|\theta})
\end{aligned}
\tag{6.2}
$$

In our approach, we make use of continuous personalized embeddings which allow for greater expressivity in the embedding space, Ω. As such, we utilize a mean-squared error (MSE) loss between a sample from the approximate posterior (modeled as a normal distribution with constant variance) and the current embedding. A derivation of the equivalence between using the MSE and log-likelihood loss to maximize the posterior is attached below.

We present the approximate normal distribution, $\mathcal{N}_{q^{\omega}_{\zeta|\theta}}$, in Equation 6.3, where ω is the mean outputted by the posterior network, and σ is the standard deviation.

$$\mathcal{N}_{q^{\omega}_{\zeta|\theta}} = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\frac{(x-\omega)^2}{\sigma^2}} \tag{6.3}$$

Theorem 6.2.1 *Minimizing the mean-squared error between a sample from the approximate posterior and the current embedding is equivalent to maximizing the log-likelihood and therefore, the evidence lower bound.*

Proof: The mean-squared error (MSE) loss is $(x-\omega)^2$, where ω is the sample from the approximate posterior, and x is the current personalized embedding used to generate the predicted action. This is equivalent to the exponent numerator in $\mathcal{N}_{q^{\omega}_{\zeta|\theta}}$. With constant variance, the exponential function is monotonic, and thus, minimizing the exponent will maximize the likelihood of the posterior. Thus, minimizing the MSE is equivalent to maximizing the likelihood of the posterior. This naturally extends to the multivariate case. □

Counterfactual Reasoning

We further enhance our model's learning capability through counterfactual reasoning [201, 65, 79, 202, 203, 204, 205]. Based upon the insight in prior work in homogeneous apprenticeship scheduling [65] that counterfactual reasoning was critical for learning scheduling strategies from demonstration, we adopt counter-

factual reasoning through pairwise comparisons. We present a novel extension to construct counterfactuals in Equations Equation 6.4-Equation 6.5 that leverages person-specific embeddings as pointwise terms.

$$z^{t,p}_{a,a'} \coloneqq [\omega_p, \bar{x}^t, x^t_a - x^t_{a'}], y^t_{a,a'} = 1 \text{ for } \forall a' \in A \setminus a \tag{6.4}$$

$$z^{t,p}_{a',a} \coloneqq [\omega_p, \bar{x}^t, x^t_{a'} - x^t_a], y^t_{a',a} = 0 \text{ for } \forall a' \in A \setminus a \tag{6.5}$$

At each timestep, we observe the decision, a, that person, p, made at time t. From each observation, we then extract 1) the feature vector describing that action, x^t_a, from state s^t, 2) the corresponding feature, $x^t_{a'}$, for an alternative action $\forall a' \in A \setminus a$, 3) a contextual feature vector capturing features common to all actions, $\bar{x}^t$, and 4) the person's embedding, ω_p. We note that each demonstrator has their own embedding which is updated through backpropagation.

Given this dataset, the apprentice is trained to output a pseudo-probability, $f(a, a', p)$ of action a being taken over action a' at time t by the human decision-maker p described by embedding ω_p, using features $z^{t,p}_{a,a'}$. To predict the probability of taking action a at timestep t, we marginalize over all other actions, as shown in Equation 6.6. Finally, the action prediction is the argument max of this probability, $\hat{a} = \arg\max_{a \in A} \hat{P}(a|t, p)$. We term models that use counterfactual reasoning as pairwise models.

$$\hat{P}(a|t, p) \sim \sum_{a' \in A} f(a, a', p) \tag{6.6}$$

Algorithm 3 PNT Training

Input: data $\vec{s} \in S$, labels $a \in A$, embeddings $\omega \in \Omega$

Output: $f^{PNT*}_{\theta|\omega}$

Initialize $f^{PNT}_{\theta|\omega}$, $q^{\omega}_{\zeta|\theta}$, Ω
for i epochs **do**
  Sample data of person p at time t : $\vec{s}^t_p, a^t_p$
  $\vec{x}^t_p \leftarrow [\omega^{(i)}_p, \vec{s}^t_p]$
  $\hat{a}^t_p \leftarrow f^{PNT}_{\theta|\omega}(\vec{x}^t_p)$
  $\mu_p, \sigma_p \leftarrow q^{\omega}_{\zeta|\theta}(\vec{s}^t_p, \hat{a}^t_p)$
  $\hat{\omega}^{(i)}_p \sim \mathcal{N}(\vec{\mu_p}, \vec{\sigma_p}^2)$
  $J_{\zeta,\theta,\omega} = |\hat{\omega}^{(i)}_p - \omega^{(i)}_p|$
  $J_{\theta,\omega} = CrossEntropy(\hat{a}^t_p || a^t_p)$
  $J \leftarrow J_{\theta,\omega} + J_{\zeta,\theta,\omega}$
  $[\omega_p, \theta, \zeta]^{(i+1)} \leftarrow [\omega_p, \theta, \zeta]^{(i)} - \eta \nabla_{\omega_p,\theta,\zeta} J$
end for

Algorithm 4 PNT Run-time Adaptation

Input: data $\vec{s}^t_{p^*} \in S$, $f^{PNT^*}_{\theta|\omega}$, training embeddings mean $\bar{\Omega}$

Output: Demonstrator p^*'s action: $\hat{a}^t_p$

$\omega^t_{p^*} = \bar{\Omega}$
for t in range(1,T) **do**
  $\vec{x}^t_{p^*} = [\omega^t_{p^*}, \vec{s}^t_{p^*}]$
  $\hat{a}^t_{p^*} \leftarrow f^{PNT^*}_{\theta|\omega}(\vec{x}^t_{p^*})$
  $a^t_{p^*} \leftarrow ObserveAction()$
  $J_{\theta,\omega} = CrossEntropy(\hat{a}^t_{p^*} || a^t_{p^*})$
  $\omega^{(t+1)}_p \leftarrow \omega^{(t)}_p - \eta \nabla_{\theta,\omega} J_{\theta,\omega}$
end for

Nota Bene: *While counterfactuals have been exploited in prior work, ours is the first to our knowledge that incorporates variational inference for counterfactual learning from heterogeneous demonstration.*

6.2.3 Interpretability via Discretization

In our work, the PNT is able to learn over datasets from heterogeneous demonstrators with high performance while still being able to convert back into a simple, interpretable decision tree post-training. Our formulation, as shown in Equation 6.1, includes two important augmentations to the original DDT formulation to allow for discretization post-training: 1) The per-node feature selector vector, $\vec{\psi}_i$,

that learns the relative importance of each candidate splitting rule, $(\beta_{i,j} x_{i,j} - \phi_{i,j})$ for each feature, j, and node, i, and 2) a per-feature splitting criterion, $\vec{\phi}_{i,j}$, that enables us to simultaneously curate per-node and per-feature splitting criteria.

During discretization of the PNT to its interpretable form, we apply the following operations to each decision node, i: 1) Set the argument max of $\vec{\psi}_i$ to 1 and all other elements to zero; 2) Set $\alpha \leftarrow \infty$. For each leaf node, i', we likewise set the argument max of $\vec{p}_{i'}$ to one and all non-maximal elements zero. The result of these operations is that each decision node has a single, Boolean splitting rule as per a standard decision tree and each leaf node dictates a single action to be taken. This procedure produces a simple yet powerful decision tree, which we show in section 6.4 outperforms all baselines even in its interpretable form.

6.2.4 Training and Runtime Procedure

Offline – At the start of training (Algorithm 3), each ω_p is a vector of uniform values. A state is sampled, s_p^t at time t, for demonstrator p, as well as the person-specific embedding, $\omega_p^{(i)}$at training iteration i, to produce a concatenated input, $\vec{x}_p^t$ as shown in lines 3 and 4. Policy $f_{\theta|\omega}^{PNT}$ uses input $\vec{x}_p^t$ to predict the demonstrator's action in that state, $\hat{a}_p^t$, as shown in line 5. The predicted action, $\hat{a}_p^t$, and state, s_p^t, are then used to recover a normal distribution, $\mathcal{N}(\vec{\mu}_p, \vec{\sigma}_p^2)$, representing that user's personalized embedding $\omega_p^{(i)}$. By sampling from this distribution, $\hat{\omega}_p^{(i)} \sim \mathcal{N}(\vec{\mu_p}, \vec{\sigma_p}^2)$, we can estimate the accuracy of our approximate posterior by computing the difference between the current embedding and the sampled embedding shown in lines 6 and 7. The learning from demonstration loss is then computed as the cross entropy loss between the true action a_p^t and the predicted action $\hat{a}_p^t$. Summed together, we have a total loss J that is dependent on ζ, θ, and ω, as shown in lines 7-10. This loss is then used to update model parameters θ, personalized embedding ω, and embedding regularization parameters ζ via SGD [206], as shown in line 11.

This process is repeated until a convergence criterion is satisfied. An overview of this training procedure is displayed in Fig. Figure 6.1b.

Online – When applying the algorithm during runtime for a new human demonstrator, p', the model updates the embedding, $\omega_{p'}$; however, θ remain static. This online update utilizes the information provided after every timestep (i.e., the true action) to converge on the type of current demonstrator in embedding space. The personalized embedding $\omega_{p'}$ for a new human demonstrator is initialized to the mean of the embeddings of demonstrators in the training set and updated as we gain more information, as shown in Algorithm Algorithm 4. In other words, we start by assuming a new expert is performing the planning task; over time, we infer how she is acting differently and update our personalized embedding accordingly (repetition of steps 3-8). Note that during deployment of our framework, the embedding regularization model $q^{\omega}_{\zeta|\theta}$ is not utilized.

Interpretable Policy – Once this tuning process has finished, the person-specific policy can be converted into an interpretable tree, through discretization of our PNTs. An interpretable model of resource allocation or planning tasks would be highly useful for a variety of reasons, from decision explanations to training purposes.

Covariate Shift – Typically, policy-based apprenticeship suffers from covariate shift. This refers to the case where the policy degrades as it enters unknown regions of the state space during deployment. To remedy the co-variate shift typically encountered with policy-based apprenticeship learning, DAgger [207] was proposed for problems where there is access to the environment model. In section 6.4, we show that pre-training with PNTs leads to a significant increase in performance for DAgger-based training while also reducing the number of environment samples DAgger requires.

6.3 Evaluation Environments

We utilize three environments to evaluate the utility of our personalized apprenticeship scheduling framework. Additional details about each domain are provided in the supplementary material.

1) Synthetic Low-Dimensional Environment The synthetic low-dimensional environment represents a simple domain where an expert will choose an action based on the state and one of two hidden heuristics. This domain captures the idea that we have homogeneity in conforming to constraints z and strategies or preferences (heterogeneity) in the form of λ. Demonstration trajectories are given in sets of 20 (which we denote a complete schedule), where each observation consists of $x^t \in \{0, 1\}$ and $z^t \in \mathcal{N}(0, 1)$, and the binary output is y^t. Exact specifications for the computation of the label are given by the observation of $y = x * \mathbb{1}_{(z>=0 \wedge \lambda=1) \vee (z<0 \wedge \lambda=2)}$, where $\mathbb{1}$ is the indicator function.

2) Synthetic Scheduling Environment The second environment we use is a synthetic environment that we can control, manipulate, and interpret to empirically validate the efficacy of our proposed method. For our investigation, we leverage a jobshop scheduling environment built on the **XD[ST-SR-TA]** scheduling domain defined by [58], representing one of the hardest scheduling problems. In this environment, two agents must work together to complete a set of 20 tasks that have upper- and lower-bound temporal constraints (i.e., deadline and wait constraints), proximity constraints, and travel-time constraints. Schedulers have a randomly-generated task-prioritization scheme dependent upon task deadline, distance, and index. The decision-maker must decide the optimal sequence of actions according to the decision-maker's own criteria. For this environment, we construct a set of

heterogeneous, mock decision-makers that schedule according to Equation 6.7.

$$\tau_i^* = \arg\max_{\tau_j \subset \boldsymbol{\tau}_S} (\rho_1 H_{EDF}(\tau_j) + \rho_2 H_{dist}(\tau_j) + H_{ID}(\tau_j, \rho_3)) \tag{6.7}$$

In this equation, our decision-maker selects a task τ_i^* from the set of tasks $\boldsymbol{\tau}_S$. The task-prioritization scheme is based on three criteria: H_{EDF} prioritizes tasks according to deadline (i.e., "earliest-deadline first"), H_{dist} prioritizes the closest task, and H_{ID} prioritizes tasks according to a user-specified highest/lowest index or value based on ρ_3 (i.e., $\rho_3(j) + (1 - \rho_3)(-j)$). The heterogeneity in decision-making comes from the latent weighting vector $\vec{\rho}$. Specifically, $\rho_1 \in \mathbb{R}$ and $\rho_2 \in \mathbb{R}$ weight the importance of H_{EDF} and H_{dist}, respectively. $\rho_3 \in \{0, 1\}$ is a mode selector in which the highest/lowest task index is prioritized. By drawing $\vec{\rho}$ from a multivariate random distribution, we can create an infinite number of unique demonstrator types. This adapted environment differs from the synthetic, low-dimensional environment in that there are a rich set of temporal, spatial, and agent-based constraints modeling the job-shop scheduling problem; furthermore, the parameters of the demonstrator's decision-making process is hidden and comprised of one discrete factor and two continuous factors. In this domain, counterfactuals are generated by consider specific task information such as availability, distance from agent, prerequisites satisfied. We note that this domain is a more complex variant of the domain in [65] as we have demonstrations of heterogeneous scheduling strategies.

3) Real-world Data: Taxi Domain We evaluate our algorithm with actual human decision-making behavior collected in a variant of the Taxi Domain in [59]. This domain describes a **ND[ST-SR-TA]** scheduling domain as defined by [58]. Our environment has three locations: the village, the airport, and the city. The taxi driver has the objective of picking up a passenger from the city or village. A dataset of 70 human-collected tree policies to solve this task (given with leaf node

Table 6.1: A comparison of heterogeneous LfD approaches. Our method achieves superior performance. Interpretable approaches are shown in the right-hand table.

Method	Low-Dim	Scheduling	Taxi
Our Method	**97.30 ± 0.3%**	**96.13 ± 2.3%**	**88.22 ± 0.6%**
Sammut et al.	55.36 ± 1.2%	5.00 ± 0.0%	76.16 ± 0.3%
Nikolaidis et al.	54.23 ± 2.5%	5.00 ± 0.0%	76.16± 0.3%
Tamar et al.	55.83 ± 0.6%	9.78 ± 0.3%	60.93± 2.8%
Hsiao et al.	56.06 ± 1.1%	11.25 ± 0.1%	76.19 ± 0.4%
InfoGAIL	54.66 ± 3.4%	25.72 ± 5.5%	75.51 ± 0.8%
DDT	55.28 ± 1.8%	52.35 ± 0.7%	76.70 ± 0.7%

Method	Low-Dim	Scheduling	Taxi
Our Method (Interpretable)	**96.13 ± 2.5%**	**99.66 ± 0.5%**	77.73 ± 1.9%
Our Method (DT + ω)	53.66 ± 2.4%	32.85 ± 0.1%	**87.85 ± 0.5%**
Gombolay et al.	55.76 ± 1.4%	45.50 ± 2.0%	75.88 ± 0.7%
Vanilla DT	55.76 ± 1.4%	32.4 ± 0.7%	74.90 ± 0.2%

actions such as "Drive to X" and "Wait for Passenger", and decision node criterion depending on the amount of wait time, traffic time, and current location) [208] are used to generate heterogeneous trajectories.

6.4 Results and Discussion

We benchmark our approach against a variety of baselines [60, 61, 62, 63, 64, 65][2]. Accuracy is the *k*-fold cross-validation, multi-class, classification accuracy for state-action pairs.

1) Synthetic Low-Dimensional Environment – Table 6.1 shows that our method for learning a continuous, personalized embedding sets the state-of-the-art (95.30%± 0.3%) for solving this latent-variable classification problem. Even after discretizing to an interpretable form, our method is still able to outperform all baselines, achieving a 96.13% ± 2.49% accuracy. A graphical depiction of the interpretable PNT model, generated through discretization is provided in the supplementary.

2) Synthetic Scheduling Environment – Table Table 6.1 shows that our personalized apprenticeship learning framework outperforms all other approaches,

[1]To infer the embeddings for the interpretable form of our PNT model, we utilize a pre-discretized version of the PNT to learn a demonstrator's embeddings, which is run prior or concurrently with the discretized version.

[2]An offline version of InfoGAIL [62] is used, as access to a simulator and ground truth reward signal, *R*, is not available in many real-world domains.

achieving 96.13% ± 2.3% accuracy in predicting demonstrator actions before conversion to an interpretable policy. After discretizing to an interpretable form, our method is able to achieve 99.66% ± 0.5% accuracy[1]. Furthermore, benchmarks that seek to handle heterogeneity [62, 63, 64] are unable to handle the complexity associated with resource coordination problems, with InfoGAIL achieving only 25.72% ± 5.5% accuracy. We provide a sensitivity analysis for this domain within our supplementary material by considering noisy demonstrations and varying the amount of data available to train the algorithm.

3) Real-world Data: Taxi Domain – As seen in Table Table 6.1, our personalized apprenticeship learning framework outperforms all other benchmarks and is the only method to achieve over 80%. Even after discretizing to an interpretable form, our method again outperforms all baselines from prior work. We posit that all other methods overfit to the most prevalent behavior, and are unable to tease out the heterogeneity represented within the training dataset.

Interpretability – We validate the effectiveness of using the discretized PNT architecture versus several interpretable architectures in Table Table 6.1. In two out of three of our environments, using our discretized PNT architecture results in a large performance gain (42.57% in the low-dim and 38.01% in the scheduling environment). In one domain, our method for distilling a DT using our PNT architecture's learned embeddings appended to the states for training a DT policy was better.

Understanding Performance of Baselines – Each baseline has particular flaws that results in its low performance on scheduling problems. [60] simply assumes homogeneity and serves as a lowerbound. [61] has two-step approach that first clusters to find modes and then trains policies; thus, there is no feedback symbol to adjust clustering-established modes. [63] utilizes a sampling-based approach which is acknowledged to require larger data sets. [64] uses a categorical vari-

able for modes results in limited expressively. InfoGAIL [62] does not have access to a ground-truth reward signal nor the environment. Our approach is the only method that both includes a decision tree-like architecture, helpful for apprenticeship scheduling [65], while also allowing for variational inference.

Performance of Pretrained Policies with DAgger – In deploying our pretrained policies within the scheduling environment, our metrics to verify whether our pre-trained PNT (PT-PNT) policies with DAgger are able to outperform randomly-initialized PNT (RI-PNT) policies with DAgger are 1) to maximize the number of tasks scheduled before a terminal state and 2) maximize the number of successful schedules. Our PT-PNT is trained on a set of 150 schedules and then with 500 episodes of DAgger. Our RI-PNT is given 650 episodes of DAgger. We find that our PT-PNT outperforms a RI-PNT by 28.57% in successful schedule completion and 12.5% in the number of tasks completed before failure. This result shows the benefits of pre-training using our framework and that our approach is amenable to DAgger-based training.

6.5 Hyperparameters and Architecture Details

Throughout this section, we will discuss the architecture, implementation details, and learning rates for all baselines and our algorithm in each domain. The runtime mentioned is in respect to a desktop with a NVIDIA RTX 2080Ti GPU and an Intel i7 processor.

6.5.1 Synthetic Low-Dimensional Environment

Each apprenticeship learning algorithm below is given 50 schedules to learn from and tests on a set of 50 unseen demonstrations.

- For the method of [60], we utilize an multi-layer perceptron (MLP) with 3 linear layers connected by ReLU activation functions. After the last linear

layer, we utilize a log softmax function to compute the log probability of which task to schedule. Each linear layer has 10 hidden units. We utilize the Adam optimizer with a learning rate of $1e^{-}3$. The runtime for training and verifying this model is under 30 minutes.

- For the method of [61], we utilize k-means clustering to separate the data into two clusters. Two neural networks (one for each cluster) are trained to imitate demonstrator data within the cluster. Each network utilizes the same architecture and learning rate used in the baseline of [60]. The runtime for training and verifying this model is under 30 minutes.

- For the method of [62], we utilize an simulator-free version of infoGAIL. The policy, discriminator, and approximate posterior are modeled by MLPs with 2 linear layers (32 hidden units) connected by a ReLU activation function, and an output activation function of a softmax, sigmoid, and softmax respectively. We initialize the number of discrete modes to 2. We utilize learning rates of $1e^{-}4, 1e^{-}3, 1e^{-}4$ respectively. For the hyperparameters of infoGAIL, we initialize λ_1 to 1, γ to 0.95, and λ_2 to 0. The runtime for training and verifying this model is under 30 minutes.

- For the method of [63], we utilize a neural network with 3 linear layers (10, 2, 2 hidden units, respectively) connected by ReLU activation functions. We use N=5 samples as our hyperparameter to estimate the intention probability distribution $\mathcal{P}(z)$. We utilize a learning rate of $1e^{-}3$ alongside Stochastic Gradient Descent (SGD). The runtime for training and verifying this model is under 30 minutes.

- For the method of [64], we utilize a bidirectional LSTM with attention followed by a linear layer as specified in their paper. For the decoder, we utilize three linear layers connected by ReLU activation functions. We utilize a learn-

ing rate of $1e^{-}3$ alongside Stochastic Gradient Descent (SGD). The runtime for training and verifying this model is under 30 minutes.

- For the method of [65], we utilize a standard decision tree (counterfactuals are not possible when $|A| \leq 2$) of depth 10. The runtime for training and verifying this model is under 30 minutes.

- For our Personalized Neural Trees, we utilize a max depth of 6 (32 leaves) and embedding dimension of 2 ($d = 2$). We set learning rates of θ to $1e^{-}3$, ω to $1e^{-}2$, and ζ to $1e^{-}3$. We find empirically that setting the learning rate of ω slightly higher allows for better LfD accuracy. For our approximate posterior, $q^{\omega}_{\zeta|\theta}$, we set the value of σ_p to zero. The runtime for training and verifying this model is under 30 minutes.

6.5.2 Synthetic Scheduling Environment

Each apprenticeship learning algorithm below is given 150 schedules to learn from and tests on a set of 100 unseen demonstrators.

- For the method of [60], we utilize an multi-layer perceptron (MLP) with six linear layers connected by ReLU activation functions. After the last linear layer, we utilize a log softmax function to compute the log probability of which task to schedule. Each linear layers have 128, 128, 32, 32, 32, and 20 hidden units, respectively. We utilize the Adam optimizer with a learning rate of $1e^{-}4$. The runtime for training and verifying this model is approximately 30 minutes.

- For the method of [61], we utilize k-means clustering to separate the data into three clusters. Three neural networks (one for each cluster) are trained to imitate demonstrator data within the cluster. Each network utilizes the same

architecture and learning rate used in the baseline of [60]. The runtime for training and verifying this model is approximately 30 minutes.

- For the method of [62], we again utilize a simulator-free version of infoGAIL. The policy follows the same network structure used in the [60] baseline. The discriminator and approximate posterior are modeled by MLPs with six linear layers (128, 128, 128, 32, 32, 32 hidden units, respecitively) connected by a ReLU activation function, and an output activation function of a sigmoid, and softmax respectively. We initialize the number of discrete modes to 3. We utilize learning rates of $1e^{-}4, 1e^{-}3, 1e^{-}4$ respectively. For the hyperparameters of infoGAIL, we initialize λ_1 to 1, γ to 0.95, and λ_2 to 0. The runtime for training and verifying this model is approximately 24-48 hours.

- For the method of [63], we utilize a neural network with 5 linear layers (128, 32, 32, 32, 32, 20, 2, 2 hidden units, respectively) connected by ReLU activation functions. We use N=5 samples as our hyperparameter to estimate the intention probability distribution $\mathcal{P}(z)$. We utilize a learning rate of $1e^{-}3$ alongside Stochastic Gradient Descent (SGD). The runtime for training and verifying this model is approximately 3 hours.

- For the method of [64], we utilize a bidirectional LSTM with attention followed by a linear layer as specified in their paper. For the decoder, we utilize six linear layers connected by ReLU activation functions. We utilize a learning rate of $1e^{-}3$ alongside Stochastic Gradient Descent (SGD). The runtime for training and verifying this model is approximately 3 hours.

- For the method of [65], we utilize a pairwise decision tree of depth 10. The counterfactuals are set to one-hot encodings of each action, as done in the original paper. The runtime for generating and verifying this model is approximately 5 minutes.

- For our Personalized Neural Trees, we utilize a max depth of six (32 leaves) and embedding dimension of 3 ($d = 3$). We set learning rates of θ to $1e^{-}2$, ω to $1e^{-}2$, and ζ to $1e^{-}2$. We find empirically that pretraining the policy network first and then adding in the posterior at a later epoch results in both good performance and mutual information maximization. This is opposed to training both models at once from scratch. For our approximate posterior, $q^{\omega}_{\zeta|\theta}$, we set the value of σ_p to zero. The runtime for training and verifying this model is approximately 24 hours.

6.5.3 Taxi Domain

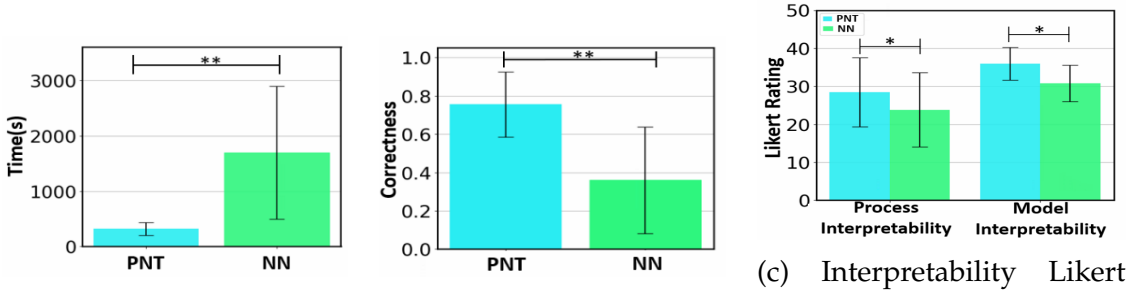
Each apprenticeship learning algorithm below is given 25 successful trajectories from each user and tested on a set of 25 unseen trajectories from each demonstrator.

- For the method of [60], we utilize the same architecture and learning rate as that of the synthetic scheduling environment. The runtime for training and verifying this model is approximately 30 minutes.

- For the method of [61], we utilize k-means clustering to separate the data into three clusters. Three neural networks (one for each cluster) are trained to imitate demonstrator data within the cluster. Each network utilizes the same architecture and learning rate used in the baseline of [60]. The runtime for training and verifying this model is approximately 30 minutes.

- For the method of [62], we utilize the same architecture and learning rate as that of the synthetic scheduling environment. The runtime for training and verifying this model is approximately 24-48 hours.

- For the method of [63], we utilize the same architecture and learning rate as that of the synthetic scheduling environment. The runtime for training and verifying this model is approximately 3 hours.



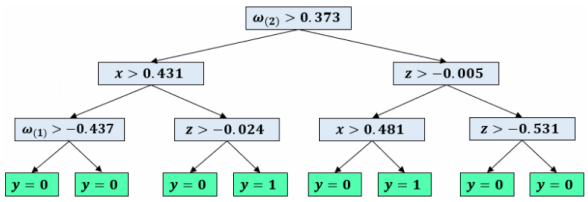
(a) Time to Simulate (b) Validation Accuracy (c) Interpretability Likert Ratings

Figure 6.2: The findings of our user study. We find significance for hypotheses **H1**, **H2**, and **H3**.

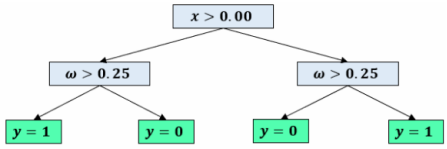
- For the method of [64], we utilize the same architecture and learning rate as that of the synthetic scheduling environment. The runtime for training and verifying this model is approximately 3 hours.

- For the method of [65], we utilize a pairwise decision tree of depth 13. The counterfactuals are set to one-hot encodings of each action, as done in the original paper. The runtime for generating and verifying this model is approximately 5 minutes.

- For our Personalized Neural Trees, we utilize a max depth of 8 (128 leaves) and embedding dimension of 3 ($d = 3$). As counterfactual task information is not readily available, we utilize one-hot encodings for each action. We set learning rates of θ to $1e^{-}2$, ω to $1e^{-}1$, and ζ to $1e^{-}2$. We find empirically that pretraining the policy network first and then adding in the posterior at a later epoch results in both good performance and mutual information maximization. For our approximate posterior, $q^{\omega}_{\zeta|\theta}$, we set the value of σ_p to zero. The runtime for training and verifying this model is approximately 12 hours.



(a) Low-dim Environment



(b) Survey Environment (counterfactual)

Figure 6.3: This figure depicts the learned PNT model after translation to an interpretable form.

6.6 Interpretable Models

As machine learning is being increasingly deployed into the real world, interpretability is required for these systems to gain human trust [209, 210, 211]. Interpretability refers to attempts that help the user understand why a machine learning model behaves the way it does. A clear visualization of a policy is one way to help a human form an accurate representation of its capabilities [212]. Furthermore, an interpretable model of resource allocation or planning tasks would be highly useful for a variety of reasons, from decision explanations to training purposes. In Figure 6.3, we display interpretable models generated through discretization for the low-dimensional environment and survey scheduling environment.

6.7 Interpretability User Study

Thus far, we have shown across a variety of datasets that our counterfactual PNT algorithm is able to achieve SOTA performance in learning from heterogeneous decision-makers. To show that our models are interpretable, we assess whether the counterfactual PNT is useful in the hands of end users. Accordingly, we conducted a novel user study to assess the interpretability of our framework. We design an online questionnaire that asks users to make predictions following each a counterfactual decision tree (PNT) and a neural network (NN). Detail about the generation of these models is in the supplementary material. We explore

three hypotheses: tree-based decision-making models are more interpretable **(H1)**, quicker to validate **(H2)**, and are easier to simulate **(H3)** than neural networks. To test **(H1)**, we ask users to answer a 13-item Likert questionnaire assessing whether the user understands the components of the decision-making model (i.e., model interpretability) and how to translate an input to an output (i.e., process interpretability) after utilizing each decision-making framework. These subjective measurements provide a practical gauge of how interpretable the decision-making models are in the hands of end-users in a **XD[ST-SR-TA]** scheduling domain as defined by [58]. To test **(H2)** and **(H3)**, we record the time required for a user to compute the model's output given a set of inputs, and measure the user's ability to correctly determine the model's output given a set of inputs, respectively.

6.7.1 User Study Results and Discussion

Our IRB-approved anonymous survey was completed by twenty adult university students. Fig. Figure 6.2 depicts the results testing **H1-H3**. The complete analysis is located in the supplementary material. **H1:** We test for normality and homoscedasticity and do not reject the null hypothesis in either case, using Shapiro-Wilk ($p > 0.3$ and $p > 0.7$) and Levene's Test ($p > 0.5$ and $p > 0.1$). We perform a paired t-test and find that tree-based models were rated statistically significantly higher than neural networks on users' Likert scale ratings for model interpretability and overall process interpretability ($p < 0.05$ and $p < 0.05$). **H2:** We perform a Wilcoxon signed-rank test on the per-model time to determine an output and find that tree-based models were statistically significantly quicker to validate than neural networks ($p < 0.01$). **H3:** We test for normality and homoscedasticity and do not reject the null hypothesis in either case, using Shapiro-Wilk ($p > 0.2$) and Levene's Test ($p > 0.4$). We perform a paired t-test and find that users using tree-based models statistically significantly achieved higher overall correctness scores

compared to NN based models ($p < 0.01$), supporting **H3**. Given these positive results, we believe our model sets a new state-of-the-art in accuracy for heterogeneous LfD (Table 6.1) and also a strong step towards making such models more interpretable.

6.8 Sensitivity Analysis of PNTs

To analyze the sensitivity of our framework, we use our synthetic scheduling environment and perturb the amount of data available to the PNT and the amount of noise (correctness) within the data. To provide a thorough analysis, we validate our approach using *k*-fold cross-validation. This entails both choosing a different subset of data to learn from and perturbing different truth-values of state-actions pairs each fold.

As shown in Figure 6.4, our PNT is reasonably robust to noise for 2, 5, and 15 schedules as there is not a steep drop in accuracy. We do not see the typical trend where the effect of noise deteriorates as the amount of data increases. We posit the cause of this deviation as follows: As the number of demonstrators increases, the embedding space Ω of the PNT tends to represent a richer distribution. While the heterogeneity among the demonstrators may remain constant (represent the same number of modes), cases in which the PNT is unable to tease out the demonstrator mode from a single schedule are more likely (due to the increase in the number of schedules), leading to an embedding distribution with higher variance. Without noise, the PNT is able to make sense of the embedding space and learn with high performance; as the amount of noise increases, it is likely more difficult to represent demonstrators compactly within the embedding space. We posit that this increased variance within the embedding space caused by the combined effect of an increased number of demonstrators and noise leads to a reduction in performance when noise is held constant and the amount of data increases.

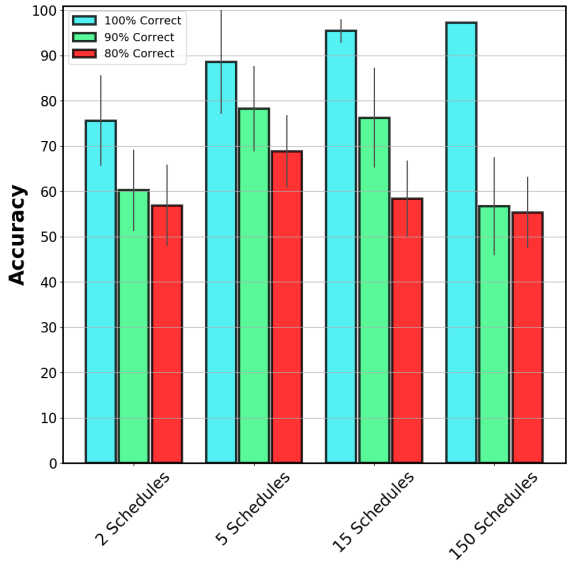


Figure 6.4: Sensitivity analysis in the synthethic scheduling environment.

As expected, as the number of schedules increase, the PNTs have higher accuracy. However, from 15 to 150 schedules (a 10x magnitude increase in data), for the case of 100% correct data, there is only a $\sim 2\%$ increase in accuracy. This result provides support to the claim of data-efficiency in our apprenticeship scheduling framework.

6.9 Conclusion

We present an apprenticeship scheduling framework for learning from heterogeneous demonstrators, leveraging a Personalized Neural Tree that is able to capture the homo- and heterogeneity in scheduling demonstrations through the use of personalized embeddings. The design of our PNT allows for translation into an interpretable form while maintaining a high level of accuracy. We demonstrate that our approach is notably superior to standard apprenticeship learning models and several approaches used in multi-modal behavior learning on synthetic and real-world data across three domains. Finally, we conduct a novel user study to assess the interpretability between our discretized trees and neural networks and

find that our discrete trees are more interpretable ($p < 0.05$), easier to simulate ($p < 0.01$), and quicker to validate ($p < 0.01$).

6.10 Broader Impact

Our interpretable apprenticeship scheduling framework has broad impacts on society and the learning from demonstration community. Our interpretable trees can give key insight into the behavior of a machine-learning-based agent, allowing a human to verify that safety constraints are being met and increasing the trustworthiness of the autonomous agent. Furthermore, these trees allow human operators to follow the decision step-by-step, allowing for verification [213, 214], and holding machines accountable [215].

Beneficiaries – Our work has the potential to benefit all human-machine collaborations, providing improved transparency, and strengthening the trustworthiness of machine teammates through policy verification. Our research contributions additionally benefit research and laboratories pursuing learning from diverse human data (which commonly contains heterogeneity).

Negatively affected parties – With any model, we believe it is important to gather consent before utilizing one's data. As our model can be used by humans as both a forward model to understand decision-maker behavior and as an inverse model to infer modality, there is a possibility of discovering latent characteristics about individuals that may reflect negatively upon them.

Implications of failure – Failure of our approach to produce high-accuracy behavior during deployment will result in a lack of trust towards the system. In the worst case, careless application may contribute to misunderstandings causing damage from a deployed robot.

Bias and Fairness – The learned behavior of our PNTs will be biased towards demonstrators within the training set. If the collected set excludes certain persona, the behavior of these persona will not be represented by our PNT. However, it should be noted that as our approach is able to better take into account heterogeneity within the training data compared to other apprenticeship learning approaches. In other words, our framework is better able to represent the entire population rather than overfitting to the most prevalent demonstrator behavior than previous approaches.

Impact on LfD community – Personalized Neural Trees can easily be extended to a variety of domains, increasing the data-efficiency, accuracy, and utility of learning-from-demonstration with multiple human demonstrators. We demonstrate this by using a PNT to learn kinesthetic robot table tennis demonstrations. We provide details about this domain, the collection process, and the results in the supplementary material.

Reproducibility – Following the NeurIPS Reproducability Checklist, we upload all code here. Within this repository, we provide collected real-world datasets, code to generate synthetic data, and code to run all benchmarks. Alongside this, we attach trained models for each domain. Further in the supplementary material, we provide specifications of our hyperparameters, descriptions of our computing infrastructure, and other details regarding runtime.

CHAPTER 7
GENERATING COBOT POLICIES VIA INTERPRETABLE REINFORCEMENT LEARNING

In this chapter, we further extend our use of DDT-based architectures to enable true policy interpretability for robots, specifically those functioning in continuous control spaces (e.g., reasoning over torque motors to rotate a joint on a robot arm or acceleration for an autonomous vehicle). This is a vital step in enabling a form of transparency for robots, which will facilitate a sense of communication from the robot to the human (discussed further in Chapter 8 and Chapter 9).

7.1 Introduction

Reinforcement learning (RL) with deep function approximators has enabled the generation of high-performance continuous control policies across a variety of complex domains, from robotics [120] and autonomous driving [71] to protein folding [216] and traffic regulation [217]. These approaches hold tremendous promise in real-world applicability and have the potential to increase traffic safety [218], decrease traffic congestion, increase average traffic speed in human-driven traffic [217], reduce CO_2 emissions, and allow for more affordable transportation [219]. However, while the performance of these controllers opens up the possibility of real-world adoption, the conventional deep-RL policies used in prior work [120, 71, 217] lack *interpretability*, limiting deployability in safety-critical and legally-regulated domains [34, 35, 36, 37].

White-box approaches, as opposed to typical black-box models (e.g., deep neural networks) used in deep-RL, model decision processes in a human-readable representation. Such approaches afford interpretability, allowing users to gain in-

sight into the model's decision-making behavior. In autonomous driving, such models would provide insurance companies, law enforcement, developers, and passengers with insight into how an autonomous vehicle (AV) reasons about state features and makes decisions. Utilizing such white-box approaches within machine learning is necessary for the deployment of autonomous vehicles and essential in building trust, ensuring safety, and enabling developers to inspect and verify policies before deploying them to the real world [209, 210, 211]. In this work, we present a novel tree-based architecture that affords gradient-based optimization with modern RL techniques to produce high-performance, interpretable policies for continuous control applications. We note that our proposed architecture can be applied to a multitude of continuous control problems ranging from robotics [120], protein folding [216], and traffic regulation [217] to high-speed autonomous driving [71] and autopilots for landing spacecraft [220].

Prior work [209, 221, 210] has attempted to approximate interpretability via explainability, a practice that can have severe consequences [114]. While the explanations produced in prior work can help to partially explain the behavior of a control policy, the explanations are not guaranteed to be accurate or generally applicable across the state-space, leading to erroneous conclusions and a lack of accountability of predictive models [114]. In autonomous driving, where understanding a decision-model is critical to avoiding collisions, local explanations are insufficient. An *interpretable model*, instead, provides a transparent *global* representation of a policy's behavior. This model can be understood directly by its structure and parameters [222] (e.g., linear models, decision trees, and our ICCTs), offering verifiability and guarantees that are not afforded by post-hoc explainability frameworks. Few works have attempted to learn an interpretable model directly; rather, prior work has attempted policy distillation to a decision tree [223, 133, 224] or imitation learning via a decision tree across trajectories generated via a deep model

[225], leaving much to be desired. Interpretable RL remains an open challenge [66]. In this work, we directly produce high-performance, interpretable policies represented by a minimalistic tree-based architecture augmented with low-fidelity linear controllers via RL, providing a novel interpretable RL architecture. Our Interpretable Continuous Control Trees are human-readable, allow for closed-form verification (associated with safety guarantees), and parity or outperform baselines by up to 33% in autonomous driving scenarios. In this work, our key contributions are:

1. We propose Interpretable Continuous Control Trees (ICCTs), a novel tree-based architecture that can be optimized via gradient descent with modern RL algorithms to produce high-performance, interpretable continuous control policies. We provide several extensions to prior differentiable decision tree (DDT) frameworks to increase expressivity and allow for direct optimization of a sparse decision-tree-like representation.

2. We show that our ICCTs are universal function approximators and can thus be utilized to learn continuous control policies in any domain, assuming that the ICCT has a reasonable depth.

3. We empirically validate ICCTs across six continuous control domains, including four autonomous driving scenarios. Further, we demonstrate ICCT's ability to learn driving policies in complex domains grounded in realistic real-world lane geometries, including the I-94 highway in Michigan, USA, and the I-280 highway in California, USA.

4. We show that our ICCTs can be verified in linear time, a vital characteristic in assessing and understanding a model's behavior under a set of inputs. Whereas black-box approaches are difficult to verify, our ICCTs can be verified

quickly, providing the possibility of safety guarantees and opening the door for safe, real-world adoption.

5. We demonstrate the utility of our ICCTs with end-users through a human-subjects study (N=34) and show that the ICCT is rated by users as easier to simulate, quicker to validate, and more interpretable than neural networks.

Recently, [66] presented a set of grand challenges in interpretable machine learning to guide the field towards solving critical research problems that must be solved before machine learning can be safely deployed within the real world. In this work, we present a solution to directly assess two challenges: (1) Optimizing sparse logical models such as decision trees and (10) Interpretable reinforcement learning. We propose a novel high-performing, sparse tree-based architecture, Interpretable Continuous Control Trees (ICCTs), which allows end-users to directly inspect the decision-making policy and developers to verify the policy for safety guarantees.

This paper presents our work in the field of Interpretable Reinforcement Learning and Explainable AI. In Section 3, we introduce the necessary preliminary work on Differentiable Decision Trees and Reinforcement Learning. Our Methodology, covered in Section 4, outlines the ICCT architecture and the Differentiable Crispification technique used for enabling policy updates via gradient-based techniques. Section 5 establishes ICCTs as universal function approximators, and Section 6 analyzes the time complexity for model verification. In Sections 7 and 8, we introduce and evaluate our ICCT across six continuous control domains. In Section 9, we offer a qualitative example of a learned policy within the Lunar Lander domain to showcase the interpretability of our model. Section 10 explores the tradeoff between performance, leaf controller sparsity, and tree depth. We compare our differential crispification method to the Gumbel-Softmax procedure in Section 11. Section 12 demonstrates the interpretability and utility of ICCTs through a 14-car physical robot demonstration. In Section 13, we introduce two realistic driving do-

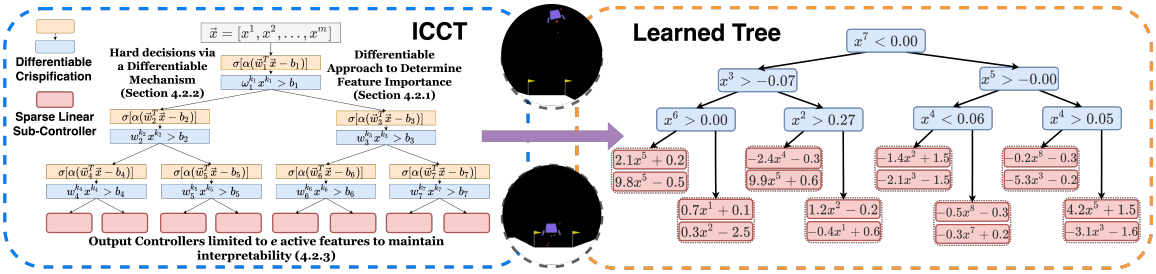


Figure 7.1: The ICCT framework (left) displays decision nodes, both in their fuzzy form (orange blocks) and crisp form (blue blocks[1]), and sparse linear leaf controllers with pointers to sections discussing our contributions. A learned representation of a high-performing ICCT policy in Lunar Lander (right) displays the interpretability of our ICCTs. Each decision node is conditioned upon only a single feature and the sparse linear controllers (to control the main engine throttle and left/right thrusters) are set to have only **one** active feature.

mains based on real-world lane geometries and find that trained ICCTs can produce safe, high-performance behavior that follows traffic regulations. Finally, Section 14 presents a user study evaluating the interpretability of our model.

7.2 Weaknesses of Prior Work with Differentiable Decision Trees

In section 3.6, we introduce Differentiable Decision Trees. Here, we discuss the mechanism used to achieve post-hoc interpretability.

7.2.1 Conversion of a DDT to a DT

DDTs with decision nodes represented in the form of Equation 3.3 are not interpretable. As DDTs maintain a one-to-one correspondence to DTs with respect to their structure, prior work [131, 9] proposed a methodology to convert a DDT into an interpretable decision tree (a process termed "crispification"). To create an interpretable, "crisp" tree from a differentiable form of the tree, prior work adopted a simplistic procedure. Starting with the differentiable form, prior work first converts each decision node from a linear combination of all variables into a single

[1]For figure simplicity, when displaying the crisp node (blue block), we assume $\alpha > 0$ in the fuzzy node (orange block). If $\alpha < 0$, the sign of the inequality would be flipped (i.e., $w_i^{k_i} x^{k_i} < b$).

feature check (i.e., a 2-arity predicate with a variable and a threshold). The feature reduction is accomplished by considering the feature dimension corresponding to the weight with the largest magnitude (i.e., most impactful), $k = \arg\max_j |w_i^j|$, where j represents the feature dimension, resulting in the decision node representation $y_i = \sigma(\alpha(w_i^k x^k - b_i))$. The sigmoid steepness, α, is also set to infinity, resulting in a "hard" decision (branch left OR right) [131, 9]. After applying this procedure to each decision node, decision nodes are represented by $y_i = \mathbb{1}(w_i^k x^k - b_i > 0)$. As each leaf node is represented as a probability mass function over output classes in prior work, each leaf node, l_d, indexed by d, must be modified to produce a single output class, o_d, during crispification. As such, we can apply an argument max, $o_d = \arg\max_a l_d^a$, where a denotes the action dimension, to find the maximum valued class within the d-th leaf distribution.

Deficiency of direct conversion from DDT to DT: This simplistic crispification procedure results in an interpretable crisp tree that is inconsistent with the original DDT (model differences arise from each *argmax* operation and setting the signmoid steepness, α, to infinity). These inconsistencies can lead to performance degradation of the interpretable model, as we show in section 7.7, and results in an interpretable model that is not representative of and inconsistent with the model learned via reinforcement learning.

In our work, we address these limitations by designing a novel architecture that updates its parameters via gradient descent while maintaining an interpretable decision-tree-like representation, thereby avoiding any inconsistencies generated through a post-hoc crispification procedure. To the best of our knowledge, we are the first work to deploy an interpretable tree-based framework for continuous control.

In this work, while ICCTs are framework-agnotistic (i.e., ICCTs will work with any RL update rule), we proceed with Soft Actor-Critic (SAC) [81] as our RL

algorithm due to its learning stability and sample efficiency. The actor objective within SAC is given in Equation 7.1, where $Q_w(s_t, a_t)$ is expected, future discounted reward parameterized by ω and α_τ is a temperature parameter that determines the relative importance of the stochastic policy entropy versus the reward.

$$J_\pi(\theta) = \mathbb{E}_{s_t \sim \mathcal{D}}[\mathbb{E}_{a_t \sim \pi_\theta}[\alpha_\tau \log(\pi_\theta(a_t|s_t)) - Q_\omega(s_t, a_t)]] \tag{7.1}$$

7.3 Method

In this section, we introduce our ICCTs, a novel interpretable reinforcement learning architecture. ICCTs are able to maintain interpretability while representing high-performance continuous control policies, making them suitable for applications that require trust and accountability, such as robotic manipulation and autonomous vehicle control. We provide several extensions to prior DDT frameworks within our proposed architecture, including 1) a differentiable crispification procedure allowing for optimization in a sparse decision-tree-like representation and 2) the addition of sparse linear leaf controllers to increase expressivity while maintaining legibility.

7.3.1 ICCT Architecture

Our ICCTs are initialized to be a symmetric decision tree with N_l decision leaves and $N_l - 1$ decision nodes. A depiction of our ICCT can be seen in Figure 7.1, with decision leaves shown in red and decision nodes shown in blue. The tree depth can be determined by $\log_2(N_l)$. Each decision leaf is represented by a sparse linear controller that operates on $\vec{x}$. Decisions are routed via decision nodes toward a leaf controller, which is then used to produce the continuous control output (e.g., acceleration or steering wheel angle). Our ICCT is similar to hierarchical models, which encompass a high-level controller that governs and coordinates multiple

low-level controllers. Prior work has shown this to be a successful paradigm in continuous control [226].

Each decision node, i, has an activation steepness weight, α, associated weights, $\vec{w}_i$, with cardinality, m, matching that of the input feature vector, $\vec{x}$, and a scalar bias term, b_i, similar to that of Equation 3.3. Each leaf node, l_d, where $d \in \{1, \ldots, N_l\}$, contains per-leaf weights, $\vec{\beta}_d \in \mathbb{R}^m$, per-leaf selector weights that learn the relative importance of candidate features, $\vec{\theta}_d \in \mathbb{R}^m$, per-leaf bias terms, $\vec{\phi}_d \in \mathbb{R}^m$, and per-leaf scalar standard deviations, γ_d. We note that if the action space is multi-dimensional, then *only* the leaf controllers (and associated weights) are expanded across $|A|$ dimensions, where $|A|$ is the cardinality of the action space. For each action dimension, the mean of the output action distribution is represented by the linear controller, l_d.

$$l_d \triangleq (\vec{u} \circ \vec{\beta}_d)^T (\vec{u} \circ \vec{x}) - \vec{u}^T \vec{\phi}_d \tag{7.2}$$

Before enforcing leaf controller sparsity (i.e., by forcing the controller to condition upon only a subset of features, Section subsubsection 7.3.2), $\vec{u} = [1, ..., N_l]^T$ is an all-ones vector, representing the set of active features within the leaf node, in which case Equation Equation 7.2 can be simplified as $l_d = \vec{\beta}_d^T \vec{x} - \vec{u}^T \vec{\phi}_d$. The output action can be determined via sampling ($a \sim \mathcal{N}(\vec{\beta}_d^T \vec{x} - \vec{u}^T \vec{\phi}_d, \gamma_d)$) during training and directly via the mean during runtime. We term decision nodes that are represented as Equation 3.3 as fuzzy decision nodes, displayed by the orange rectangles within the left-hand side of Figure 7.1. Similarly, we term the leaf node, l_d, when it is represented in the dense representation of $\vec{\beta}_d^T \vec{x} - \vec{u}^T \vec{\phi}_d$, as a fuzzy leaf node. Here, we parameterize the bias term as a vector, $\vec{\phi}_d$, instead of a scalar as in our decision nodes to provide a corresponding bias for each feature and facilitate feature-wise optimization across different dimensions of the bias term.

Utilizing a novel differentiable crispification procedure to convert fuzzy decision nodes into crisp decision nodes (i.e., 2-arity predicate with a variable and a

threshold) and fuzzy leaf nodes into sparse leaf nodes (i.e., linear controller conditioned upon a small subset of features), our model representation follows that of a decision tree with sparse linear controllers at the leafs (shown on the right-hand side of Figure 7.1). We further discuss our differentiable crispification procedure in Sections subsubsection 7.3.2-subsubsection 7.3.2 (i.e., the mechanism that translates orange blocks to blue within Figure 7.1) and leaf controller sparsification procedure in subsubsection 7.3.2.

While decision trees (DT) are generally considered interpretable [35], trees of arbitrarily large depths can be difficult to understand [227] and simulate [134]. A sufficiently sparse DT is desirable and considered interpretable [228]. Furthermore, the utilization of linear controllers at the leaves also allows us to maintain interpretability, as linear controllers are widely used and generally considered interpretable for humans [229]. However, for large feature spaces typically encountered in real-world problems, such a controller would not be interpretable. As such, in our work, we utilize *sparse* linear controllers at the leaves to balance the trade-off between sparsity/complexity in logic, model depth, and performance.

7.3.2 ICCT Key Elements

In this section, we discuss our ICCT's interpretable procedure for determining an action given an input feature. As our ICCT configuration maintains interpretability both during training via RL and deployment, the inference of an action must allow gradient flow. We present a novel approach that allows for direct optimization of sparse logical models via an online differentiable crispification procedure to determine feature importance (subsubsection 7.3.2) and allows for bifurcate decisions (subsubsection 7.3.2). In Algorithm Algorithm 5, we provide general pseudocode representing our ICCT's decision-making process.

At each timestep, the ICCT model, $I(\cdot)$, receives a state feature, $\vec{x}$. To determine

an action in an interpretable form, in Steps 1 and 2 of Algorithm Algorithm 5, we start by applying the differentiable crispification approaches of NODE_CRISP and OUTCOME_CRISP to decision nodes so that each decision node is only conditioned upon a single variable (subsubsection 7.3.2), and the evaluation of a decision node results in a Boolean (subsubsection 7.3.2). Once the operations are completed, in Step 3, we can utilize the input feature, $\vec{x}$, and logically evaluate each decision node until arrival at a linear leaf controller (INTERPRETABLE_NODE_ROUTING). The linear leaf controller is then modified, in Step 4, to only condition upon e features, where e is a sparsity parameter specified a priori (subsubsection 7.3.2). Finally, an action can be determined via sampling from a Gaussian distribution conditioned upon the mean generated via the input-parameterized sparse leaf controller, l_d^*, and scalar variance maintained within the leaf, γ_d, during training (Step 6) or directly through the outputted mean (Step 8) during runtime.

Algorithm 5 ICCT Action Determination

Input: ICCT $I(\cdot)$, state feature $\vec{x} \in S$, controller sparsity e, training flag $t \in \{\text{TRUE}, \text{FALSE}\}$
Output: action $a \in \mathbb{R}$

1: NODE_CRISP$(\sigma(\alpha(\vec{w}_i^T \vec{x} - b_i))) \rightarrow \sigma(\alpha(w_i^k x^k - b_i))$
2: OUTCOME_CRISP$(\sigma(\alpha(w_i^k x^k - b_i))) \rightarrow \mathbb{1}(\alpha(w_i^k x^k - b_i) > 0)$
3: $l_d \leftarrow$ INTERPRETABLE_NODE_ROUTING$(\vec{x})$
4: $l_d^* \leftarrow$ ENFORCE_CONTROLLER_SPARSITY(e, l_d)
5: **if** t **then**
6: $\quad a \sim \mathcal{N}(l_d^*(\vec{x}), \gamma_d)$
7: **else**
8: $\quad a \leftarrow l_d^*(\vec{x})$
9: **end if**

Decision Node Crispification

The NODE_CRISP procedure in Algorithm Algorithm 5 recasts each decision node to split upon a single dimension of $\vec{x}$. Instead of using a non-differentiable argument max function as in [131] to determine the most impactful feature dimension, we utilize a softmax function, also known as softargmax [230], described by Equa-

tion 7.3. In this equation, we denote the softmax function as $f(\cdot)$, which takes as input a set of class weights and produces class probabilities. Here, $\vec{w}_i$ represents a categorical distribution with class weights, individually denoted by w_i^j, and τ is the temperature, determining the steepness of $f(\cdot)$.

$$f(\vec{w}_i)_k = \frac{\exp\left(\frac{w_i^k}{\tau}\right)}{\sum_{j=1}^{m} \exp\left(\frac{w_i^j}{\tau}\right)} \tag{7.3}$$

While setting the temperature near-zero would satisfy our objective of producing a one-hot vector, where the outputted class probability of the index of the most impactful feature would be one, this operation can lead to a large variance within the gradients and unstable training. We therefore set the softmax temperature, τ equal to 1, which we find effective empirically, and utilize a differentiable one_hot function, $g(\cdot)$, to produce a one-hot vector with the element associated with the highest-weighted class set to one and all other elements set to zero. We display the procedure for determining the one-hot vector associated with the largest magnitude in Equation 7.4.

$$\vec{z_i} = g(f(|\vec{w}_i|)) \tag{7.4}$$

Here, $|\vec{w}_i|$ represents a vector with absolute elements within $\vec{w}_i$. *We maintain differentiability in the procedure described in Equation 7.4 by utilizing the straight-through trick [68].* This allows us to obtain the desired output, $\vec{z}_i$, a one-hot vector over weights required for the purpose of matching the decision node representation of a decision tree, while maintaining gradients for all weight parameters $\{w_i^j\}_{j=1}^m$ (by treating the gradient with respect to $\vec{z_i}$ as the gradient with respect to $f(|\vec{w}_i|)$). This procedure is further elaborated in subsubsection 7.3.2 and Algorithm Algorithm 8.

The one-hot encoding $\vec{z_i}$ can then be element-wise multiplied by the original

weights to produce a new set of weights with only one active weight, $\vec{z}_i \circ \vec{w}_i \rightarrow \vec{w}'_i$. Accordingly, the decision node representation is transferred from $\sigma(\alpha(\vec{w}_i^T \vec{x} - b_i)) \rightarrow \sigma(\alpha(\vec{w}_i'^T \vec{x} - b_i)) = \sigma(\alpha(w_i^k x^k - b_i))$, where k is the index of the most impactful feature. We provide an algorithm detailing the NODE_CRISP procedure in Algorithm Algorithm 6, where node crispification recasts each decision node to split upon a single dimension of the input.

Algorithm 6 Node Crispification: NODE_CRISP($\cdot$)

Input: The original fuzzy decision node $\sigma(\alpha(\vec{w}_i^T \vec{x} - b_i))$, where i is the decision node index, $\vec{w}_i = [w_i^1, w_i^2, ..., w_i^j, w_i^{j+1}, ..., w_i^m]^T$, and m is the number of input features

Output: The intermediate decision node representation $\sigma(\alpha(w_i^k x^k - b_i))$ (see the green box in Figure 7.2)

1: $\vec{z}_i$ = DIFF_ARGMAX($|\vec{w}_i|$) (DIFF_ARGMAX($\cdot$) displayed in Algorithm Algorithm 8)

2: $\vec{w}'_i = \vec{z}_i \circ \vec{w}_i$

3: $\sigma(\alpha(w_i^k x^k - b_i)) = \sigma(\alpha(\vec{w}_i'^T \vec{x} - b_i))$

Node crispification takes as input the original fuzzy decision node, $\sigma(\alpha(\vec{w}_i^T \vec{x} - b_i))$, where *all* input features are used in determining the output of decision node i. The output of this function is an intermediate decision node, $\sigma(\alpha(w_i^k x^k - b_i))$, where the output of decision node i is only determined by *a single* feature, x^k. To perform this transformation, in Line 1, we use the differentiable argument max function (in Algorithm Algorithm 8) to produce a one-hot vector, $\vec{z}_i$, with the element associated with the most impactful feature set to one and all other elements set to zero. In Line 2, we element-wise multiply the one-hot encoding, $\vec{z}_i$, by the original weights, $\vec{w}_i$, to produce a new set of weights with only one active weight, $\vec{w}'_i$. In Line 3, we show that by multiplying $\vec{x}$ by $\vec{w}'_i$, we can obtain the intermediate decision node $\sigma(\alpha(w_i^k x^k - b_i))$, where k is the index of the most impactful feature (i.e., $k = \arg\max_j(|w_i^j|)$). The transformation to each decision node performed by node crispification is further displayed by the green arrow in Figure 7.2.

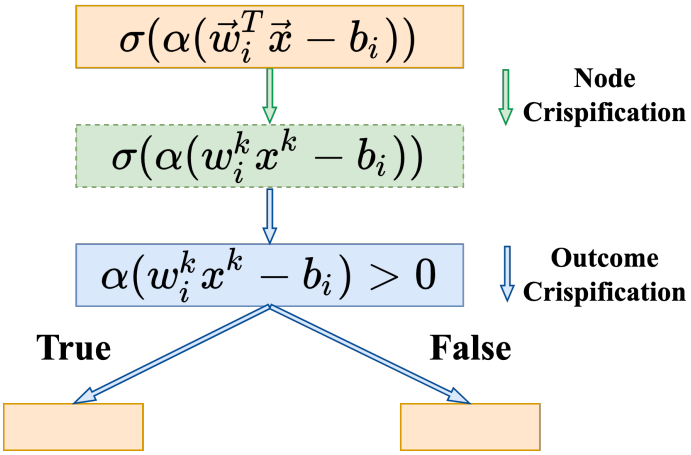


Figure 7.2: This figure displays the process of differentiable crispification, including node crispification (Algorithm Algorithm 6) and outcome crispification (Algorithm Algorithm 7). The node crispification sparsifies the weight vector, $\vec{w}_i$, and chooses the most impactful feature. The outcome crispification enforces a "hard" decision at the node rather than a "soft" decision, so the computation proceeds along one branch. Both operations are differentiable through the use of the straight-through trick.

Below, we conduct a short example detailing our procedure.

Example: Consider we have a two-leaf decision tree (one decision node) with an input feature, $\vec{x} = [2, 3]^T$ with a cardinality of 2 (i.e., $m = 2$), associated weights of $\vec{w_1} = [2, 1]^T$, and a bias term b_1 of 1. The sigmoid steepness, α, is also set equal to 1 for simplicity. It is easily seen that the most impactful weight within the decision node is $w_1^1 = 2$. Utilizing Equation 7.4, we can compute $\vec{z_1} = [1, 0]^T$. Multiplying $\vec{z_1}$ to the original weights, $\vec{w_1}$, and input feature, $\vec{x}$, subtracting b_1, and scaling by α, we have an crisp decision node $\sigma(2x^1 - 1)$ or $\sigma(3) = 0.95$. Here, 0.95 is the probability that the decision node evaluates to True. We display a depiction of this example in the left-hand side of Figure 7.3.

Decision Outcome Crispification

Here, we describe the second piece of our online differentiable crispification procedure, noted as Outcome_Crisp in Algorithm Algorithm 5. Outcome_Crisp trans-

lates the outcome of a decision node so that the outcome is a Boolean decision rather than a probability generated via a sigmoid function (i.e., $p = y_i$ for True/Left Branch and $q = 1 - y_i$ for False/Right Branch). We start by creating a soft vector, $\vec{v}_i = [\alpha(w_i^k x^k - b_i), 0]$, for the i^{th} decision node. Placing $\vec{v}_i$ through a softmax operation, we can produce the probability of tracing down the left branch or right. We can then apply the differentiable one-hot function, $g(\cdot)$, to produce a hard decision of whether to branch left or right, denoted by y_i and described by Equation 7.5.

$$[y_i, 1 - y_i] = g(f(\vec{v}_i)) \tag{7.5}$$

Essentially, the decision node will evaluate to True if $\alpha(w_i^k x^k - b_i) > 0$ and right otherwise. This process can be expressed as an indicator function $\mathbb{1}(\alpha(w_i^k x^k - b_i) > 0)$.

We note the procedure of $g(f(\vec{v}_i))$ is highly similar to that in Equation 7.4, both outputting a one-hot vector, with the former input being the decision node weights, $|\vec{w}_i|$, and the latter input being the soft vector representation of the decision node outcome, $\vec{v}_i$. We provide an algorithm detailing the Outcome_Crisp procedure in Algorithm Algorithm 7, where outcome crispification translates the outcome of a soft decision node to a hard decision node, resulting in a Boolean output from the decision node rather than a set of probabilities.

Algorithm 7 Outcome Crispfication: Outcome_Crisp(·)

Input: The intermediate decision node $\sigma(\alpha(w_i^k x^k - b_i))$, where i is the decision node index, $k = \arg\max_j(|w_i^j|)$, and w_i^j is the jth element in $\vec{w}_i$

Output: Crisp decision node $\mathbb{1}(\alpha(w_i^k x^k - b_i) > 0)$ (see the blue box in Figure 7.2)

1: $\vec{v}_i = [\alpha(w_i^k x^k - b_i), 0]$

2: $\vec{z}_i' =$ diff_argmax$(\vec{v}_i)$ (diff_argmax$(\cdot)$ displayed in Algorithm Algorithm 8)

3: $\mathbb{1}(\alpha(w_i^k x^k - b_i) > 0) = \vec{z}_i'[0]$

Outcome crispification takes in the intermediate decision node $\sigma(\alpha(w_i^k x^k - b_i))$,

which outputs the probability of branching left. The output of OUTCOME_CRISP is the crisp decision node, $\mathbb{1}(\alpha(w_i^k x^k - b_i) > 0)$, a Boolean decision to trace down to the left branch OR right. In Line 1, we construct a soft vector representation of the decision node i's output, $\vec{v}_i$, by concatenating $\alpha(w_i^k x^k - b_i)$ with a 0. In Line 2, we use the differentiable argument max function (in Algorithm Algorithm 8) to produce a one-hot vector, $\vec{z}_i$, where the first element represents the Boolean outcome of the decision node. In Line 3, we show that the output of the crisp decision node, $\mathbb{1}(\alpha(w_i^k x^k - b_i) > 0)$, can be obtained by choosing the first element of vector $\vec{z}_i$ (we use bracket indexing notation here, starting with zero). We further display the transformation performed by outcome crispification by the blue arrows in Figure 7.2.

Example: Continuing the example in subsubsection 7.3.2, we can take the outputted crisp decision node and generate a vector $\vec{v}_1 = [2x^1 - 1, 0]^T$, or by substituting the input feature, $\vec{v}_1 = [3, 0]^T$. Performing the operations specified in Equation 7.5, we receive the intermediate output from the softmax $[0.95, 0.05]^T$ (rounded to two decimal places), the one-hot vector $[1, 0]^T$ after performing the one_hot operation, and finally $y_1 = 1$, denoting that the decision-tree should follow the left branch. We display a depiction of this example on the right-hand side of Figure 7.3.

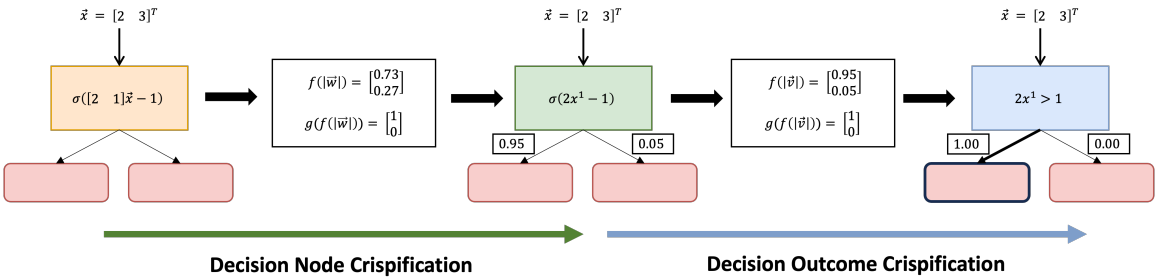


Figure 7.3: This figure displays the process of decision node crispification and decision outcome crispification across the Examples within subsubsection 7.3.2 and subsubsection 7.3.2.

Conversion to a Simple Form: The above crispification processes produce decision-

tracing equal to that of a DT. The node representation can thus be simplified to that of Figure 7.1 by algebraically reducing each crisp decision node to $x^k > \frac{b_i}{w_i^k}$ (given $\alpha w_i^k > 0$) or $x^k < \frac{b_i}{w_i^k}$ (given $\alpha w_i^k < 0$).

Sparse Linear Leaf Controllers

After applying the decision node and outcome crispification to all decision nodes and outcomes, the decision can be routed to leaf node (Step 3). This section describes the procedure to translate a linear leaf controller to condition upon e features (Enforce_Controller_Sparsity procedure in Algorithm Algorithm 5), enforcing sparsity within the leaf controller and thereby, enhancing ICCT interpretability. As noted in subsection 7.3.1, our ability to utilize sparse sub-controllers allows us to balance between interpretability and performance. The sparsity of the linear sub-controllers ranges from setting $e = 0$ and maintaining static leaf distributions, where each leaf node contains scalar value representing the mean (i.e., ICCT-static in section 7.7), to $e = m$, containing a linear controller parameterized by the entire feature space of $\vec{x}$ (i.e., ICCT-complete in section 7.7).

Equation 7.6 displays the procedure for determining a k-hot encoding, $\vec{u}_d$, that represents the k (or in our case, e) most impactful selection weights within a leaf's linear controller. The k-hot function, denoted by $h(\cdot)$, takes as input a vector of weights and returns an equal-dimensional vector with k elements set to one. The indexes associated with the elements set to one match the k highest-weighted elements within the input feature.

$$\vec{u}_d = h(f(|\vec{\theta}_d|)) \tag{7.6}$$

Here, $|\vec{\theta}_d|$ represents a vector with absolute elements within the per-leaf selector weights, $\vec{\theta}_d$. As before, we maintain differentiability and formulate a differentiable

top-k function in Equation 7.6 by utilizing the straight-through trick [68] and iteratively applying DIFF_ARGMAX$(\cdot)$ (Algorithm Algorithm 8) k times. In Equation 7.7, we transform a fuzzy leaf node, for leaf, l_d (represented in Equation 7.2), into a sparse linear sub-controller, l_d^*, with the sparse feature selection vector, $\vec{u_d}$, given by Equation Equation 7.6.

$$l_d^* \triangleq (\vec{u}_d \circ \vec{\beta}_d)^T (\vec{u}_d \circ \vec{x}) + \vec{u}_d^T \vec{\phi}_d \tag{7.7}$$

A depiction of the sparse sub-models can be seen at the bottom right-hand side of Figure 7.1, where the sparsity of the sub-controllers, e, is set to 1 and the dimension of the action space is 2.

Differentiable Argument Max Function for Differentiable Crispification

In this section, we provide a description of the differentiable argument max function.

Algorithm 8 Differentiable Argument Max Function for Crispification: DIFF_ARGMAX$(\cdot)$

Input: Logits $\vec{q}$

Output: One-Hot Vector $\vec{h}$

1: $\vec{h}_{soft} \leftarrow f(\vec{q})$

2: $\vec{h}_{hard} \leftarrow$ ONE_HOT(ARGMAX$(f(\vec{q}))$) {step 1 for function $g(\cdot)$}

3: $\vec{h} = \vec{h}_{hard} + \vec{h}_{soft} -$STOP_GRAD$(\vec{h}_{soft})$ {step 2 for function $g(\cdot)$}

Similar to [231], we present a function call (in Algorithm Algorithm 8) that can be utilized to maintain gradients over a non-differentiable argument max operation. The function takes in a set of logits, $\vec{q}$, and applies a softmax operation, denoted by $f(\cdot)$, to output $\vec{h}_{soft}$, as shown in Line 1. In Line 2, the logits are transformed using an argument max followed by a one-hot procedure, causing the removal of gradient information, producing $\vec{h}_{hard}$. In Line 3, we combine $\vec{h}_{soft}$, $\vec{h}_{hard}$, and STOP_GRAD$(\vec{h}_{soft})$ to output $\vec{h}$, where STOP_GRAD$(\cdot)$ keeps the values and removes the

gradient data of $\vec{h}_{soft}$. The outputted value of $\vec{h}$ is equal to that of $\vec{h}_{hard}$. However, the gradient maintained within $\vec{h}$ is associated with $\vec{h}_{soft}$. Automatic differentiation frameworks can then utilize the outputted term to perform backpropagation. Here, the operations in Line 2 and Line 3 compose function $g(\cdot)$ in Equation 7.4 and Equation 7.5.

Summary: In this section, we discuss our novel interpretable reinforcement learning architecture, ICCTs. We present a description of components of ICCTs, including decision nodes and linear leaf controllers, and provide a differentiable crispification procedure allowing for optimization similar to a differentiable decision tree (DDT) while maintaining a forward-propagation process identical to a sparse decision tree. To the best of our knowledge, we present the first truly interpretable tree-based framework for continuous control.

7.4 Universal Function Approximation

In this section, we provide a proof to show our ICCTs are universal function approximators, that is, can represent any decision surface given enough parameters. Our ICCT architecture consists of successive indicator functions, whose decision point lies among a single dimension of the feature space, followed by a linear controller to determine a continuous control output. For simplicity, we assume below that the leaf nodes contain static distributions. However, maintaining a linear controller at the leaves is more expressive and thus, the result below generalizes directly to ICCTs.

The decision-making of our ICCTs can be decomposed as a sum of products. In Equation 7.8, we display a computed output for a 4-leaf tree, where decision node outputs, y_i, are given by Equation 3.3. Here, the sigmoid steepness, α is set to infinity (transforming the sigmoid function into an indicator function) resulting in hard decision points ($y_i \in \{0, 1\}$). Equation 7.8 shows that the chosen action is

determined by computation of probability of reaching a leaf, y, multiplied by static tree weights maintained at the distribution, p.

$$ICCT(x) = p_1(y_1 * y_2) + p_2(y_1 * (1 - y_2)) \\ + p_3((1 - y_1) * y_3) + p_4 * ((1 - y_1) * (1 - y_3)) \tag{7.8}$$

Equation 7.8 can be directly simplified into the form of $G(x) = \sum_{j=1}^{N} p_j \sigma(w_j^T x + b_j)$, similar to Equation 1 in [232]. [232] demonstrates that finite combination of fixed, univariate functions can approximate any continuous function. The key difference between our architecture is that our univariate function is an indiator function rather than the commonly used sigmoid function. Below, we provide two lemmas to show that indicator functions fall within the space of univariate functions [232].

Lemma 7.4.1 *An indicator function is sigmoidal.*

Proof: This follows from the definition of sigmoidal: $\sigma(t) \to 1$ as $t \to \infty$ and $\sigma(t) \to 0$ as $t \to -\infty$.

Lemma 7.4.2 *An indicator function is discriminatory.*

Proof: As an indicator function is bounded and measureable, by Lemma 1 of [232], it is discriminatory.

Theorem 7.4.3 *Let σ be any continuous discriminatory function. ICCTs are universal function approximators, that is, dense in the space of $C(I_n)$. In other words, there is a representation of ICCTs, $I(x)$, for which $|I(x) - f(x)| < \epsilon$ for all $x \in I_n$, for any function, f ($f \in C(I_n)$), where $C(I_n)$ denotes the codomain of an n-dimensional unit cube, I_n.*

Proof: As the propositional conditions hold for Theorem 1 in [232], the result that ICCTs are dense in $C(I_n)$ directly follows. We note that as the indicator function jump-continuous, we refer readers to [233] whom extend UFA for $G(x) =$

$\sum_{j=1}^{N} p_j \sigma(w_j^T x + b_j)$ to the case when σ is jump-continuous. As ICCTs are dense in $C(I_n)$, ICCTs are universal function approximators. □

7.5 Model Robustness Verification

A desirable property for a controller is to be able to verify the policy, ensuring the controller outputs desirable values for a set of inputs. This often translates to answering the following question:

- *For a range of input features, what is the range of output values that can be expected?*

By answering this question, engineers and end-users can attain key insights into a policy's decision-making behavior and make guarantees about its behavior. Utilizing autonomous driving as an example, an engineer may want to verify that if a human is detected within 5 meters, the acceleration of the vehicle is never above $5m/s^{-2}$. Verification of policies is vital in creating models that are safe and can help ensure that models accurately perform the purpose they are designed for.

In this section, we analyze the time complexity of verifying an ICCT. Following [234], we formalize the problem of *robustness verification* as follows: Consider a regression model: $f : \mathbb{R}^d \rightarrow \mathbb{R}$, where d is the dimension of the input features and the output is a real-valued scalar. In [234], for input, x, the minimal adversarial perturbation is defined by Equation Equation 7.9, where $y = f(x)$ is the expected controller output value. The solution to this equation determines the minimum input perturbation required to have the controller output a value different from the expected value, y.

$$r^* = \min_{\lambda} ||\lambda||_\infty \text{ s.t. } f(x + \lambda) \neq y \tag{7.9}$$

As we are concerned with *continuous* control policies, where a slight perturbation on the input may cause a slight perturbation on the controller output, we instead generalize y to an expected output range, $y = \{a, b\}$, and search for an input pertur-

bation that causes y to fall out of the specified range, i.e., $f(x+\lambda) \notin \{a, b\}$. Following our autonomous driving example, finding the minimum adversarial perturbation allows an engineer to understand what input deviations may result in unsafe or undesirable controller outputs. Thus below, we present a discussion of the time complexity associated with solving Equation Equation 7.9 of neural network models and our ICCTs. *Positively, for our ICCTs, determining the minimal input perturbation can be done in linear time.*

Due to complex nonlinearities within neural network architectures, small input perturbations can lead to large changes in predicted values [235, 236], making it intractable to perform verification. Often, verification is only possible if the neural network architecture follows certain desiderata [237] or by utilizing convex relaxations of nonlinear activation functions [238]. Even so, such verification, in the best case, is only NP-Complete.

ICCTs, on the other hand, can be verified in linear time. Here, we first present an analysis showing ICCT-static (ICCT maintaining static Gaussian distributions at each leaf) can be verified in linear time. We then extend this proof to ICCT-complete, where linear controllers parameterized by the input features are maintained at each leaf. For simplicity, we assume that the output value of our controller is solely determined by the mean value within the leaf distribution of our ICCTs, similar to how ICCTs are deployed during runtime.

Assume we have a decision tree that has n decision nodes and $n + 1$ leaf nodes. For each decision node i, we define a variable, t_i, representing which feature is activated within the decision node, and a variable, η_i, representing the threshold maintained with the decision node (which is equal to $\frac{b_i}{w_i}$). Depending on the outcome of the decision node, the computation will further proceed to either the left or right child until arrival at a leaf node, where the leaf node contains a scalar value representing the Gaussian mean.

Following [234], the key of the proof is to split the d-dimensional input space to $n+1$ hyperspaces corresponding to leaf nodes, such that any input will result in falling into one and only one leaf node's hyperspace. This can be done by traversing the entire tree and computing bounding boxes via a depth-first search. By definition, all input variables will reach the root node, resulting in a root node box represented by the Cartesian product $B = [-\infty, \infty] \times \cdots [-\infty, \infty]$, of cardinality d. Each child's box can be obtained by splitting one interval from the parent box based on the split condition (the variable selected, t_i, and threshold, η_i). This process can be completed until the entire tree is traversed (via a depth-first search fashion), resulting in a time complexity of $O(nd)$, where n is the number of nodes and d is the cardinality of the input space.

The distance from an input x to a leaf node's region can be written as a vector, $\epsilon(x, B^i) \in \mathbb{R}^d$, defined by the Equation 7.10, where l_t^i, r_t^i, represent the upper and lower bound of a node's Cartesian product on dimension t for leaf i, respectively.

$$\epsilon(x, B^i)_t = \begin{cases} 0 & \text{if } i_t \in (l_t^i, r_t^i) \\ x_t - r_t^i & \text{if } x_t > r_t^i \\ l_t^i - x_t & \text{if } x_t \leq l_t^i \end{cases} \tag{7.10}$$

Thus, the minimal distortion required to result in an incorrect output value can be obtained by Equation 7.11, where $y = \{a, b\}$ is the output range desired, and v_i is the value for leaf node i. Intuitively, Equation Equation 7.11 finds the minimum perturbation on a feature, t, that leads to a leaf node with associated output outside of the desired output range.

$$r^* = \min_{i:v_i \notin y, t \in [d]} \epsilon(x, B^i)_t \tag{7.11}$$

The time complexity of the verification algorithm for ICCT robustness is $O(nd)$

due to the traversal of the tree, the combination of bounding boxes, and the minimal perturbation finding in Equation Equation 7.11 by iterating over all leaf node and all feature dimensions. As stated above, this results in addressing the *decision problem of robustness verification* in linear time.

When extending to ICCTs with linear controllers at the leaves, we can utilize the same formalism to obtain the bounding boxes represented by Cartesian products at each leaf. However, as each leaf controller depends on input variables, the range of outputs would require extra computation based on the bounding boxes and linear controller parameters. Because of the monotony of the linear controllers with respect to each input feature, the computation is still $O(d)$ for each leaf node, and therefore we can still achieve the overall $O(nd)$ time complexity for the *robustness verification*.

7.6 Environments

Here, we provide short descriptions across six domains used in our extensive evaluation. We start with two common continuous control problems, Inverted Pendulum, and Lunar Lander, provided by OpenAI Gym [69]. We then test across four autonomous driving scenarios: Lane-Keeping provided by [70] and Single-Lane Ring Network, Multi-Lane Ring Network, and Figure-8 Network all provided by the Flow deep reinforcement learning framework for mixed autonomy traffic scenarios [71].

Inverted Pendulum: In Inverted Pendulum [239], a control policy must apply throttle (ranging from +3 to move left to -3 to move right) to balance a pole. The observation includes the cart position, velocity, pole angle, and pole angular velocity.

- *Lunar Lander:* In Lunar Lander [240, 69], a policy must throttle the main engine and side engine thrusters for a lander to land on a specified landing pad. The observation is 8-dimensional, including the lander's current position, linear

Worst to Best:

Method	Common Continuous Control Problems		Autonomous Driving Problems			
	Inverted Pendulum	Lunar Lander	Lane Keeping	Single-Lane Ring	Multi-Lane Ring	Figure-8
DT	155.0 ± 0.9 256 leaves (766 params)	−285.5 ± 15.6 256 leaves (1022 params)	−359.0 ± 11.0 256 leaves (766 params)	**123.2 ± 0.03** 32 leaves (94 params)	503.2 ± 24.8 256 leaves (1022 params)	831.1 ± 1.1 256 leaves (766 params)
DT w\ DAgger	776.6 ± 54.2 32 leaves (94 params)	184.7 ± 17.3 32 leaves (126 params)	**395.2 ± 13.8** 16 leaves (46 params)	121.5 ± 0.01 16 leaves (46 params)	1249.4 ± 3.4 31 leaves (122 params)	**1113.8 ± 9.5** 16 leaves (46 params)
CDDT-Crisp	5.0 ± 0.0 2 leaves (5 params)	−451.6 ± 97.3 8 leaves (37 params)	−43526.0 ± 15905.0 16 leaves (61 params)	68.1 ± 18.7 16 leaves (61 params)	664.5 ± 192.6 16 leaves (77 params)	322.9 ± 47.1 16 leaves (61 params)
ICCT-static	**984.0 ± 10.4** 32 leaves (125 params)	**192.4 ± 10.7** 32 leaves (157 params)	374.2±55.8 16 leaves (61 params)	120.5 ± 0.5 16 leaves (61 params)	**1271.7 ± 4.1** 16 leaves (77 params)	1003.8±27.2 16 leaves (61 params)
ICCT-1-feature	1000.0 ± 0.0 8 leaves (45 params)	190.1 ± 13.7 8 leaves (69 params)	437.6 ± 7.0 16 leaves (93 params)	121.6 ± 0.5 16 leaves (93 params)	1269.6 ± 10.7 16 leaves (141 params)	1072.4 ± 37.1 16 leaves (93 params)
ICCT-2-feature	1000.0 ± 0.0 4 leaves (29 params)	258.4 ± 7.0 8 leaves (101 params)	458.5 ± 6.3 16 leaves (125 params)	**121.9 ± 0.5** 16 leaves (125 params)	1280.4±7.3 16 leaves (205 params)	**1088.6 ± 21.6** 16 leaves (125 params)
ICCT-3-feature	**1000.0 ± 0.0** 2 leaves (17 params)	275.8 ± 1.5 8 leaves (133 params)	448.8 ± 3.0 16 leaves (157 params)	120.8 ± 0.5 16 leaves (157 params)	**1280.8 ± 7.7** 16 leaves (269 params)	1048.7 ± 46.7 16 leaves (157 params)
ICCT-L1-sparse	1000.0 ± 0.0 4 leaves (29 params)	265.2 ± 4.3 8 leaves (165 params)	465.5 ± 4.3 16 leaves (253 params)	121.5 ± 0.3 16 leaves (765 params)	1275.3 ± 6.7 16 leaves (2189 params)	993.2 ± 14.6 16 leaves (509 params)
ICCT-complete	1000.0 ± 0.0 2 leaves (13 params)	**300.5 ± 1.2** 8 leaves (165 params)	**476.6 ± 3.1** 16 leaves (253 params)	120.7 ± 0.5 16 leaves (765 params)	1248.6 ± 3.6 16 leaves (2189 params)	994.1 ± 29.1 16 leaves (509 params)
CDDT-controllers Crisp	84.0 ± 10.4 2 leaves (13 params)	−126.6 ± 53.5 8 leaves (165 params)	−39826.4 ± 21230.0 16 leaves (253 params)	97.9 ± 12.0 16 leaves (765 params)	639.62 ± 160.4 16 leaves (2189 params)	245.5 ± 48.5 16 leaves (509 params)
MLP-Lower	1000.0 ± 0.0 79 params	231.6 ± 49.8 110 params	474.7 ± 5.8 127 params	**121.8 ± 0.6** 151 params	646.4 ± 151.2 221 params	868.4 ± 100.9 103 params
MLP-Upper	1000.0 ± 0.0 121 params	288.7 ± 2.8 222 params	467.9 ± 8.5 407 params	121.8 ± 0.3 709 params	1239.5 ± 4.2 3266 params	1077.7 ± 31.1 1021 params
MLP-Max	1000.0 ± 0.0 67329 params	**298.5 ± 0.7** 68610 params	**478.2 ± 6.7** 69377 params	121.7 ± 0.4 77569 params	1011.9 ± 141.3 83458 params	**1104.3 ± 9.4** 73473 params
CDDT	**1000.0 ± 0.0** 2 leaves (8 params)	226.4 ± 44.5 8 leaves (86 params)	464.7 ± 5.4 16 leaves (226 params)	120.9 ± 0.5 16 leaves (706 params)	**1248.0 ± 6.4** 16 leaves (1036 params)	1033.2 ± 24.1 16 leaves (466 params)
CDDT-controllers	1000.0 ± 0.0 2 leaves (16 params)	289.0 ± 2.4 8 leaves (214 params)	469.7 ± 11.1 16 leaves (418 params)	120.1 ± 0.3 16 leaves (1410 params)	1243.8 ± 3.6 16 leaves (2092 params)	1010.9 ± 25.7 16 leaves (914 params)

Table 7.1: In this table, we display the results of our evaluation. For each evaluation, we report the mean (± standard error) and the complexity of the model required to generate such a result. Our table is broken into three segments, the first containing equally interpretable approaches that utilize static distributions at their leaves. The second segment contains interpretable approaches that maintain linear controllers at their leaves. The ordering of methods denotes the relative interpretability. The third segments displays black-box approaches. We bold the highest-performing method in each segment, and break ties in performance by model complexity. We color table elements in association with the number of parameters and performance. Reddish colors relate to a larger number of policy parameters and lower average reward.

velocity, tilt, angular velocity, and information about ground contact. The continuous action space is two-dimensional, with the first dimension controlling the main engine thruster and the second controlling the side engine thrusters.

- *Lane-Keeping [70]:* A control policy must control a vehicle's steering angle to stay within a curving lane. The observation is 12-dimensional, which consists of lane information and the vehicle's lateral position, heading, lateral speed, yaw rate, and linear, lateral, and angular velocity. The action is the steering angle to control the vehicle.

- *Flow Single-Lane Ring Network [71]:* A control policy must apply acceleration commands to a vehicle agent to stabilize traffic flow consisting of 21 other human-driven (synthetic) vehicles. The observation includes the world position and velocity of all vehicles.

- *Flow Multi-Lane Ring Network [71]:* A control policy must apply acceleration and lane-changing commands to an ego vehicle to stabilize the flow of noisy traffic flow across multiple lanes. The observation includes the world position and velocity of all vehicles.

- *Flow Figure-8 Network [71]:* A control policy must apply acceleration to a vehicle to stabilize the flow in a Figure-8 network (contains a section where the vehicles must cross paths at the center of the 8), requiring the policy to adapt its control input to create a stable flow through this section. The observation is the world position and velocity of all vehicles.

7.7 Results

In this section, we present the set of baselines we test our model, ICCTs, against. Then, we report the results of our approach versus these baselines across the six

continuous control domains, as shown in Table Table 7.1. All presented results are across five random seeds, and all differentiable frameworks are trained via SAC [81]. Each tree-based framework is trained while maximizing performance and minimizing the complexity required to represent such a policy, thereby emphasizing interpretability. We release our codebase at https://github.com/CORE-Robotics-Lab/ICCT.

7.7.1 Baselines

We provide a list of baselines alongside abbreviations used for reference and brief definitions below. We compare against interpretable models, black-box models, and models that can be converted post-hoc into an interpretable form. For each method, we also include details regarding the number of active parameters utilized in each model (a surrogate measure for model complexity). We briefly list the following notations for an easier understanding of the computation of the number of parameters for each model discussed below. The number of leaf nodes is N_l (the number of decision nodes is $N_l - 1$), the dimension of the observation space is m, the number of active features within the leaf controllers is e, and the dimension of the action space is d_a. The calculated number of parameters is denoted as N_p for each model. Our approach, ICCT-e-feature, has a number of parameters of $N_p = 3(N_l - 1) + (2e + 1)d_a N_l = (2ed_a + d_a + 3)N_l - 3$.

- Continuous DDTs (CDDT): We translate the framework of [131] to function with continuous action-spaces by modifying the leaf nodes to represent static probability distributions. Here, $N_p = (m+2)(N_l-1)+d_a N_l = (d_a+m+2)N_l-m-2$. When converted into an interpretable form post-hoc, this approach is reported as CDDT-crisp which has a number of parameters, $N_p = 3(N_l - 1) + d_a N_l = (3 + d_a)N_l - 3$.

- Continuous DDTs with controllers (CDDT-controllers): We modify CDDT

leaf nodes to utilize linear controllers rather than static distributions. Here, $N_p = (m+2)(N_l-1) + (m+1)d_a N_l = (md_a + d_a + m + 2)N_l - m - 2$. When converted into an interpretable form post-hoc, this approach is reported as CDDT-controllers Crisp that has $N_p = 3(N_l-1)+(m+1)d_a N_l = (md_a+d_a+3)N_l-3$.

- ICCTs with static leaf distributions (ICCT-static): We modify the leaf architecture of our ICCTs to utilize static distributions for each leaf (i.e., set $e = 0$). Comparing ICCT and ICCT-static displays the effectiveness of the addition of sparse linear sub-controllers. Here, the number of parameters can be computed as $N_p = 3(N_l - 1) + d_a N_l = (3 + d_a)N_l - 3$.

- ICCT with complete linear sub-controllers (ICCT-complete): We allow the leaf controllers to maintain weights over all features (no sparsity enforced, i.e., $e = m$). Comparing ICCT-complete and CDDT-controllers displays the effectiveness of the proposed differentiable crispification procedure. Here, the number of parameters can be computed as $N_p = 3(N_l - 1) + (m + 1)d_a N_l = (md_a + d_a + 3)N_l - 3$.

- ICCT with L1-regularized controllers (ICCT-L1-sparse): We achieve sparsity via L1-regularization applied to ICCT-complete rather than enforcing sparsity directly via the Enforce_Controller_Sparsity procedure. While this baseline produces sparse sub-controllers, there are drawbacks limiting its interpretability. L1-regularization enforces weights to be near zero rather than exactly zero. These small weights must be represented within leaf nodes, and thus, the interpretability of the resulting model is limited. Here, the number of parameters can be computed as $N_p = 3(N_l-1)+(m+1)d_a N_l = (md_a+d_a+3)N_l-3$.

- Multi-layer Perceptron (MLP): We maintain three variants of an MLP. The first (MLP-Max) contains a very large number of parameters, typically utilized in continuous control domains. The second (MLP-Upper) maintains

approximately the same number of parameters of our ICCTs with sparse leaf controllers during training, including all inactive parameters. The last (MLP-Lower) maintains approximately the same number of *active* parameters as our ICCTs with sparse leaf controllers. The number of parameters of MLP depends on the size of the network, including all weights and bias parameters.

- Decision Tree (DT): We train a DT via CART [38] on state-action pairs generated from MLP-Max. This baseline represents policy distillation from a high-performance black-box policy to an interpretable model. Here, the number of parameters can be computed as $N_p = 2(N_l - 1) + d_a N_l = (2 + d_a)N_l - 2$.

- DT w\ DAgger: We utilize the DAgger imitation learning algorithm [207] to train a DT to mimic the MLP-Max policy. The number of parameters can be computed as in the DT baseline above.

7.7.2 Discussion

We present the results of our trained policies in Table Table 7.1. We provide the performance of each method alongside the associated complexity of each benchmark in Table Table 7.1 across three sections, with the top section representing interpretable approaches that maintain static distributions at their leaves, the middle section containing interpretable approaches that maintain linear controllers at their leaves, and the bottom section containing black-box methods.

Static Leaf Distributions (Top): The frameworks of DT, DT w\ DAgger, CDDT-Crisp, and ICCT-static maintain similar representations and are equal in terms of interpretability given that the approaches have the same depth. We see that across three of the six control domains, ICCT-static is able to widely outperform both the DT and CDDT-Crisp models. In the remaining three domains, ICCT-static

outperforms CDDT-Crisp by a large margin and achieves competitive performance compared to DTs, even without access to a superior expert policy.

Linear Controller Leaf (Middle): Here, we rank frameworks (top-down) by their relative interpretability. As the sparsity of the sub-controller decreases, the interpretability diminishes. We see that most approaches are able to achieve the maximum performance in the simple domain of Inverted Pendulum. However, CDDT-controllers-crisp encounters an inconsistency issue from the crispification procedure of [131, 9] and achieves very low performance. In regards to interpretability-performance tradeoff, in Inverted Pendulum, we see that as sparsity increases within the sub-controller, a lower-depth ICCT can be used to achieve a equally high-performing policy. We note that across all domains, we do not find such a linear relationship. We provide additional results within section 7.9 that provide deeper insight into the interpretability-performance tradeoff.

Black-Box Approaches (Bottom): MLP-based approaches and fuzzy DDTs are not interpretable. While the associated approaches perform well across many of the six domains, the lack of interpretability limits the utility of such frameworks in real-world applications such as autonomous driving. We see that in half the domains, highly-parameterized architectures with over 65,000 parameters are required to learn effective policies.

Comparison Across All Approaches: We see that across all continuous control domains, CDDT-Crisp and CDDT-controllers Crisp typically are the lowest-performing models. This displays the drawbacks of the crispification procedure of [131, 9] and the resultant performance inconsistency. Comparing our ICCTs to black-box models, we see that in all domains, we parity or outperform deep highly-parameterized models in performance while reducing the number of parameters required by orders of magnitude. In the difficult Multi-Lane Ring scenario, we see that we can outperform MLPs by 33% on average while achieving a 300x-600x

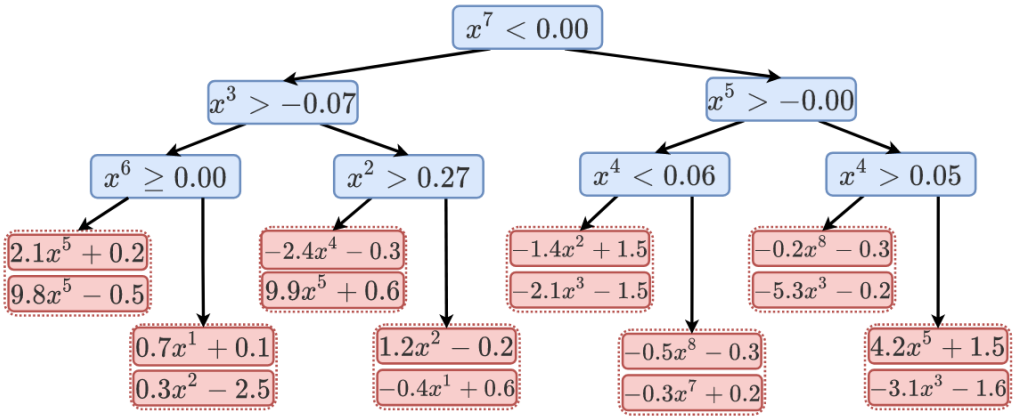


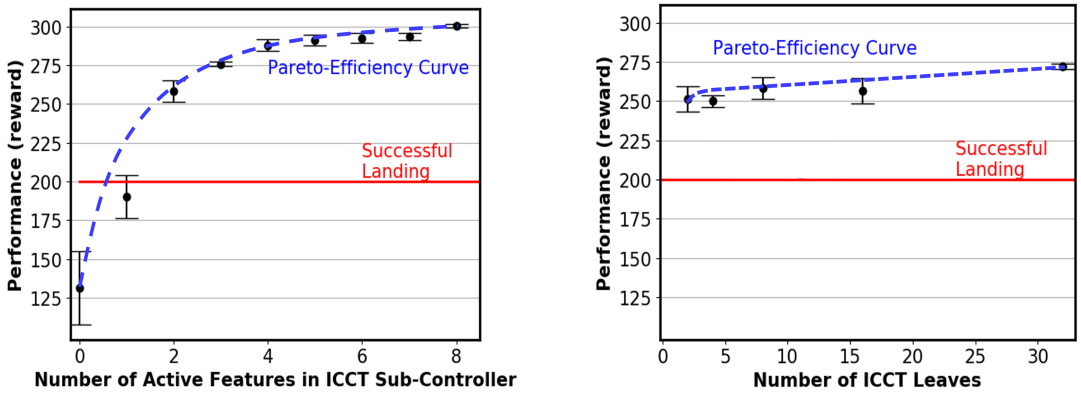
Figure 7.4: A Learned ICCT in Lunar Lander

reduction in the number of policy parameters required.

Overall, we find strong evidence for our Interpretable Continuous Control Trees, displaying and validating the ability to at least parity black-box approaches while maintaining high interpretability. Our novel architecture and training procedure provide a strong step towards providing solutions for two grand challenges in interpretableML: (1) Optimizing sparse logical models such as DTs and (10) Interpretable RL.

7.8 Qualitative Exposition of ICCT Interpretability

Here, we provide a display of the utility and interpretability of a learned ICCT model. In Figure 7.4, we present our learned ICCT model in Lunar Lander, rounding each element to two decimal places for brevity. The displayed figure is an ICCT-1-feature model (i.e., only one active feature within the sparse sub-controller). The 8-dimensional input in Lunar Lander is composed of position (x^1,x^2), velocity (x^3,x^4), angle (x^5), angular velocity (x^6), left (x^7) and right (x^8) lander leg-to-ground contact. The action space is two-dimensional: the first (dictated by the top of each pair of the red-colored leaves) controls the main engine thrust, and the second (bottom) controls the net thrust for the side-facing engines. The tree can be interpreted



(a) Performance vs. Number of Controller Features

(b) Performance vs. Number of ICCT Leaves

Figure 7.5: In this figure, we display the interpretability-performance tradeoff of our ICCTs with respect to the number of active features within our linear sub-controllers (Figure 7.5a) and the number of tree leaves (Figure 7.5b) in Lunar Lander. Within each figure, we display the approximate Pareto-Efficiency Curve and denote the reward required for a successful lunar landing as defined by [69].

as follows: taking the leftmost path as an example, if the left leg is not touching the ground, the horizontal velocity is greater than -0.07 m/s, and the angular velocity is greater than 0.00 rad/s, then the main engine action is $2.1 * (\text{the lander angle}) + 0.2$, and the side engine action is $9.8 * (\text{the lander angle}) - 0.5$. Such a tree has several use cases: 1) An engineer/developer may pick certain edge cases and verify the behavior of the lander. Performing robustness verification on our ICCTs can be done in linear time (see section 7.5), while DNN verification is NP-complete [237]. 2) An engineer can evaluate the decision-making in the tree and detect anomalies. Furthermore, there are hands-on use-cases of such a model, such as threshold editing (directly modifying nodes to increase affordances), etc. Finally, as seen throughout this example, the structure of our ICCTs lends itself to the classification of being a "neuro-symbolic system [241]", being able to discover high-performance decision-making policies by recognizing patterns between state-action pairs and high-reward behavior, and capturing these relationships through a hierarchical tree structure, mimicking a first-order logical formula.

7.9 Ablation: Interpretability-Performance Tradeoff

Here, we provide an ablation study over how ICCT performance changes with respect to the number of active features within our linear sub-controllers and depth of the learned policies. [228] states that decision trees are interpretable because of their simplicity and that there is a cognitive limit on how complex a model can be while still being understandable. Accordingly, for our ICCTs to maximize interpretability, we emphasize the sparsity of our sub-controllers and attempt to minimize the depth of our ICCTs. Here, we present a deeper analysis by displaying the performance of our ICCTs while varying the number of active features, e, from ICCT-static to ICCT-complete (Figure 7.5a), and varying the number of leaves maintained within the ICCT from $N_l = 2$ to $N_l = 32$. We conduct our ablation study within Lunar Lander.

In Figure 7.5a, we show how the performance of our ICCTs changes as a function of active features in the Sub-Controller. Here, we fix the number of ICCT leaves to 8. We see that as the number of active features increase, the performance also increases. However, there is a tradeoff in interpretability. As greater than 200 reward is considered successful in this domain, a domain expert may determine a point on the Pareto-Efficiency curve that maximizes the interpretability-performance tradeoff. In Figure 7.5b, we show how the performance of our ICCTs changes as a function of tree depth while fixing the number of active features in the ICCT sub-controller to two. We see a similar, albeit weaker, relationship between performance and interpretability. As model complexity increases, there is a slight gain in performance and a large decrease in interpretability. The Pareto-Efficiency curve provides insight into the interpretability-performance tradeoff for ICCT tree depth.

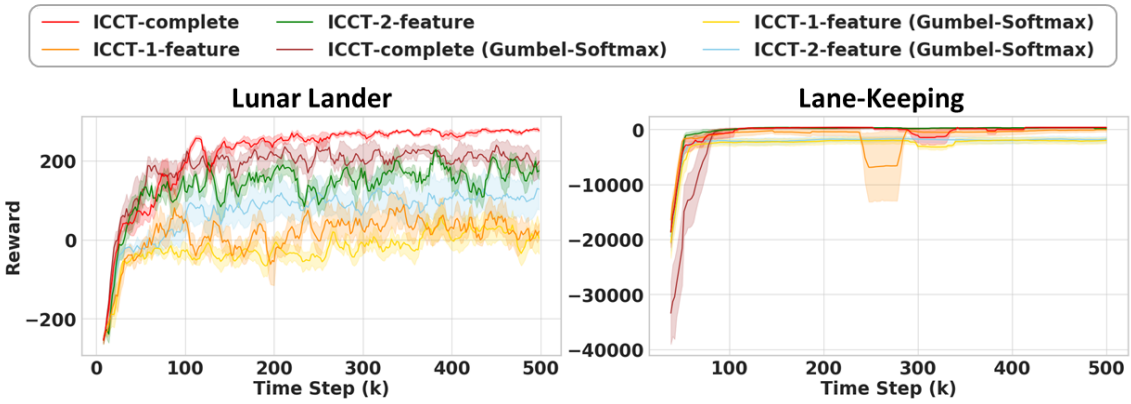


Figure 7.6: This figure displays the average running rollout rewards of six methods for the ablation study during training. The results are averaged over 5 seeds, and the shadow region represents the standard error.

7.10 Ablation: Differentiable Argument Max and Gumbel-Softmax

In this section, we provide an ablation study on the differentiable operator used in ICCTs to perform decision node crispification, perform decision outcome crispification, and enforce sub-controller sparsity. Here, we substitute the Softmax function with a Gumbel-Softmax [172] function, a widely-used differentiable approximate sampling mechanism for categorical variables. To allow ICCTs to utilize the Gumbel-Softmax function, as opposed to DIFF_ARGMAX$(\cdot)$, we modify the original Softmax function, f, introduced by Equation 7.3, to f' defined in Equation 7.12. Here, $\vec{w}_i$ is an m-dimensional vector, $[w_i^1, \cdots, w_i^m]^T$, and $\{g_i^j\}_{j=1}^m$ are i.i.d samples from a Gumbel$(0, 1)$ distribution [172].

$$f'(\vec{w}_i)_k = \frac{\exp\left(\frac{w_i^k + g_i^k}{\tau}\right)}{\sum_j^m \exp\left(\frac{w_i^j + g_i^j}{\tau}\right)} \tag{7.12}$$

In Table Table 7.2, we compare the performance of ICCT-complete, ICCT-1-feature, and ICCT-2-feature to their variants using Gumbel-Softmax in Lunar Lander and Lane-Keeping. All the methods and their corresponding variants are trained using the same hyperparameters. From the results shown in Figure 7.6 and

Method	Lunar Lander	Lane-Keeping
ICCT-complete	300.5 ± 1.2	476.6 ± 3.1
ICCT-complete (Gumbel-Softmax)	276.7 ± 7.0	412.6 ± 31.3
ICCT-complete (Gumbel-Softmax, Crisp)	239.0 ± 18.9	309.1 ± 94.6
ICCT-1-feature	190.1 ± 13.7	437.6 ± 7.0
ICCT-1-feature (Gumbel-Softmax)	113.2 ± 43.1	−853.4 ± 333.2
ICCT-1-feature (Gumbel-Softmax, Crisp)	−20.1 ± 50.0	−658.114 ± 345.3
ICCT-2-feature	258.4 ± 7.0	458.5 ± 6.3
ICCT-2-feature (Gumbel-Softmax)	161.7 ± 54.8	−560.6 ± 251.6
ICCT-2-feature (Gumbel-Softmax, Crisp)	62.3 ± 82.2	−945.0 ± 331.0

Table 7.2: This table shows a performance comparison between ICCTs utilizing our proposed differentiable argument max function (DIFF_ARGMAX(·) in Algorithm Algorithm 8), and a variant of ICCTs utilizing the Gumbel-Softmax function (fuzzy and crisp). Across each approach, we present our findings across Lunar Lander and Lane-Keeping and include ICCTs with fully parameterized sub-controllers (ICCT-complete) and sparse sub-controllers.

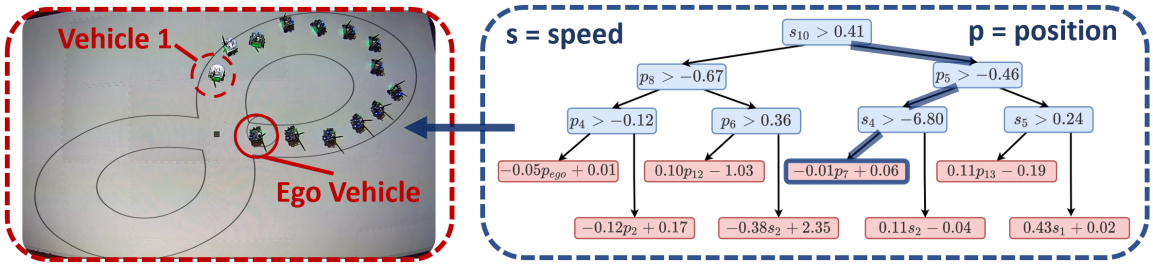


Figure 7.7: In this figure, we display our ICCTs controlling a vehicle in a 14-car physical robot demonstration within a Figure-8 traffic scenario. Active nodes and edges are highlighted by the right online visualization, where s_i represents the speed of vehicle i, and p_i represents the position of vehicle i. We include a full video, including an enlarged display of our ICCT at https://sites.google.com/view/icctree .

Table Table 7.2, we find that the addition of Gumbel noise reduces performance by a wide margin. Furthermore, comparing crisp ICCTs utilizing Gumbel-Softmax to ICCTs utilizing Gumbel-Softmax, we see that due to the sampling procedure within the Gumbel-Softmax, an inconsistency issue arises between fuzzy and crisp performance. Such results support our design choice of the differentiable argument max function over the Gumbel-Softmax.

7.11 Physical Robot Demonstration

Here, we demonstrate our algorithm with physical robots in a 14-car figure-8 driving scenario and provide an online, easy-to-inspect visualization of our ICCTs, which controls the ego vehicle. We utilize the Robotarium, a remotely accessible swarm robotic research platform [175], to demonstrate the learned ICCT policy. The demonstration displays the feasibility of ego vehicle behavior produced by our ICCT policy and provides an online visualization of our ICCTs. A frame taken from the demonstrated behavior is displayed in Figure 7.7. We provide a complete video of the demonstrated behavior and the online visualization of the control policy at https://sites.google.com/view/icctree.

7.12 Case Studies on Complex Driving Domain Grounded in Realistic Lane Geometries

We extend our analysis of the ICCT with experiments on two realistic driving domains modeled after 1) the I-94 highway in Michigan and 2) the I-280 highway in California, and verify ICCT's ability to drive safely on crowded highways.

7.12.1 The I-94 Domain

The I-94 domain is built with the SUMO traffic simulator [242] and modeled off the Interstate Highway 94 from Exit 190 to Exit 192. As illustrated in Figure 7.8, the I-94 domain consists of three ramp entries and three ramp exits. The task of the ego vehicle is to enter from the first ramp entry and exit from the last ramp exit and do so as quickly as possible while being safe.

The state space of I-94 is 19-dimensional and consists of the ego longitudinal location (i.e., progress of the driving), the ego speed, the ego latitudinal location (i.e., lane information), and the surrounding vehicles' status. We define "surrounding"

Figure 7.8: This figure illustrates the I-94 domain. The red arrows denote the traffic flow directions. There are four traffic inflows: one highway inflow (leftmost) and three ramp inflows. There are also four traffic outflows: highway outflow (rightmost) and three ramp outflows.

Metrics	ICCT					MLP		
	2 leaves 1-feature	2 leaves 2-feature	4 leaves 1-feature	4 leaves 2-feature	8 leaves 1-feature	Lower	Upper	Max
Environment Returns	878.72	899.65	858.62	**910.93**	871.80	907.11	901.29	897.13
# number of parameters	15	23	33	49	69	46	862	71426
# collisions	0	0	0	0	0	0	0	1
# hard-brakes	0	0	0	0	0	0	0	2
# unsafe lane changes	14	7	30	**0**	9	13	**0**	3

Table 7.3: This table shows our findings within the I-94 domain. Environment returns are the average of ten evaluation episodes after training has been completed. The remaining metrics are computed through a summation over occurrences of the respective phenomena across the ten evaluation episodes.

vehicles as the leading cars and following cars on each lane with respect to the ego car, and for each surrounding vehicle, we provide the distance to the ego car as well as its velocity. As there are four lanes on I-94, the surrounding vehicles' information is 16-dimensional. The action space of the RL agent is two-dimensional, which controls the ego vehicle's acceleration and steering angle.

As we would like to ensure both the performance and safety of the RL agent, we design a 6-component reward function, $R_{\text{I-94}} = \sum_{i=1}^{6} R_i$. Here, R_1 represents the positive reward for ego speed, R_2 represents the negative constant time penalty, R_3 denotes the negative penalty for too-small headways, R_4 provides the negative penalty for wrong routing (i.e., the ego vehicle does not exit properly via the last ramp), R_5 encodes the negative penalty for emergency braking behaviors, and R_6 is the negative penalty for unsafe lane changes. Intuitively, R_1 and R_2 motivate the ego car to move faster, R_4 helps the ego car to follow the desired route, and R_3, R_5, and R_6 encourage the RL vehicle to be safe by keeping headways and avoid

dangerous behaviors.

7.12.2 I-94 Results

We summarize the results of the I-94 domain in Table Table 7.3. We compare five sizes of ICCTs and three sizes of MLPs (as introduced in subsection 7.7.1) across four metrics. Firstly, we find that the ICCT and MLP can achieve similar performance on the environment returns metric, with ICCT (4 leaves, 2-feature) variant achieving the highest returns. The second and third metrics we consider are the number of collisions and the number of hard brakes in the evaluation of 10 episodes after training the policies. As our driving simulator, SUMO, always prevents an actual collision by applying a high deceleration, we regard a deceleration that is higher than $10m \cdot s^{-2}$ as a collision, and a deceleration that is between $5m \cdot s^{-2}$ and $10m \cdot s^{-2}$ as a hard brake. We observe that most ICCT and MLP models achieve zero collisions and hard brakes, with the exception of MLP-Max. We hypothesize that MLP-Max may overfit to the training data and generate undesired large-model artifacts, showing the brittleness of large MLPs in general. The last metric we consider is the number of unsafe lane changes, which counts the number of instances that the ego car tries to merge onto a lane with too small of a headway or tailway, or tries to change to a non-existing lane (e.g., attempts to change to the left lane on the left-most lane). We observe that the best-performing ICCT model with 4 leaves of 2 features achieves zero unsafe lane changes and higher rewards than the best MLP model. Comparing ICCT-2-feature with 4 leaves and MLP-Upper, although the two models have similar performance on all metrics, the ICCT model maintains 94% fewer parameters compared with the MLP-Upper model. Thus, we conclude that in the case study of the I-94 domain, ICCT models were able to learn to successfully accomplish the task, meeting both performance and safety objectives while being highly parameter-efficient.

7.12.3 The I-280 Domain

The I-280 domain is a complex environment modeled off the Interstate Highway 280 around the Palo Alto area in California[2]. An overview of the I-280 Domain is shown in Figure 7.9a. The ego vehicle is tasked to join the highway from the ramp and then exit the environment at the end of the highway. As shown in Figure 7.9b and Figure 7.9c, the I-280 domain is more challenging than the I-94 domain due to the multiple road geometries encountered across a trajectory, including road splits and the introduction and disappearance of lanes. The state space for I-280 is 19-dimensional and consists of the ego lane index, the ego speed, the speed limit within the current segment, the target lane (i.e., the lanes that lead the ego to the next road segment on its route), ego progress within the current segment, and the surrounding vehicles' information. In I-280, the surrounding vehicles include the leading cars and following cars on the lanes adjacent to the ego car (i.e., the left lane, the current lane of the ego car, and the right lane). For each surrounding vehicle, the domain provides its distance to the ego car and velocity information. The action space of the ego agent is two-dimensional, which controls the ego vehicle's acceleration and steering angle. *There are 119,347 passenger vehicles in total involved within this domain, creating an extremely large-scale simulation with complex, stochastic vehicle interactions.* However, as dense traffic makes the simulation significantly slow, we reduce the traffic density in the original I-280 Domain to involve vehicles that are near to the ego vehicle controlled by our model. The modification makes the environment renders faster and boosts the training process.

As I-280 introduces the notion of the speed limit and complex road routing, we design an 8-component reward function, $R_{\text{I-280}} = \sum_{i=1}^{8} R_i$. Here, R_1 represents the positive reward for ego speed, R_2 represents the negative penalty if the ego vehicle is not in the target lane, R_3 encodes the negative penalty for too-small headways,

[2]Our I-280 domain can be found here: https://github.com/songanz/flow_evaluation.

R_4 provides the negative penalty for too-large headways, R_5 encodes a negative penalty for emergency braking behaviors, R_6 represents a negative penalty for unsafe lane changes, R_7 is a progress reward once the ego vehicle finishes 40% and 60% of its route, and R_8 is a reward if the ego vehicle arrives at its destination (i.e., accomplishing the task). Intuitively, R_1 and R_4 motivate the ego vehicle to move faster and follow traffic flow, R_2 helps the ego car to follow the correct route, R_3, R_5, and R_6 encourage the RL vehicle to be safe by keeping headways and avoiding dangerous operation, and R_7 and R_8 encourage the RL vehicle to move further in its trajectory.

7.12.4 I-280 Results

We have summarized the results of the I-280 case study in Figure 7.10. We compare six sizes of the ICCT and three sizes of an MLP. The "environment returns" metric represents the average episode reward in 10 rollouts. The environment returns for the 2-feature ICCT with 2 leaves and 2-feature ICCT with 8 leaves are significantly higher than the MLP models, demonstrating the ICCT's capability on more complex and realistic domains. We also observe that ICCT-2-Feature performs better than ICCT-1-feature.

Summary: Across both realistic autonomous driving domains, we find that ICCTs can serve as safe continuous control models that follow traffic regulations and maintain high performance with respect to the domain objectives.

7.13 Interpretability User Study

In the previous sections, we have validated ICCT's efficacy in learning high-performance, safe policies. In this section, we seek to verify ICCT's interpretability with end-users through a human-subject experiment. To get a comprehensive understanding of ICCT's interpretability compared with neural networks, we design

a $3 \times 3 \times 2 \times 2$ experiment. We describe the four independent variables as follows:

Models (3-levels): 1) *Tree*, 2) *MLP*, and 3) *Paragraph*. We compare the interpretability across the three models. The condition, *Tree*, refers to an ICCT in its original tree form (e.g., Figure 7.4). The condition, *Paragraph*, refers to a text description encompassing a set of rules extracted from an ICCT. We obtain this form by first training an ICCT and then transforming it into a text description after training has been completed. During the transformation, each leaf node corresponds to a sub-paragraph, which is formed by two components, namely the conditions and the consequence. The conditions of a sub-paragraph describe the junction of all decision nodes on the path from the root node to the leaf, and the consequence is the leaf node itself. For example, a four-leaf tree corresponds to four sub-paragraphs. We hypothesize such a *Paragraph* form may positively contribute to the interpretability of the ICCT as it does not assume prior familiarity with tree-based models. The condition, *MLP*, displays a multi-layer perceptron.

Repeats of evaluation (3-level): 1st, 2nd, and 3rd. To test the interpretability of the three models, we task participants to calculate the output of the model given the model parameters and its inputs. For each model, we ask participants to make predictions with the same model three times, each with varying inputs. This allows us to attain a lower-variance estimation of both the participant's accuracy and time spent in model computation. This condition also allows us to compare the learning effect of interpreting each of the models (i.e., improvements from repeated evaluations).

Contexts (2-levels): 1) With Context and 2) Without Context. We investigate whether providing context (i.e., a visualization of the traffic scenario) contributes to the user's rating of model interpretability. This condition helps us understand whether each model's interpretability is its inherent property from its model structure or is related to specific example contexts.

Domains (2-levels): 1) Multi-Lane Ring and 2) I-94. We examine whether models' interpretability is impacted by the environment's complexity.

Out of the four factors, we design *Models* and *Repeats of evaluation* to be within-subject factors as we seek to test for the learning effects and choose *Contexts* and *Domains* to be between-subject factors. We test for the following six hypotheses from the user study.

- H1: *Tree* and *Paragraph* are easier to simulate than neural network, i.e., the simulation of *Tree* and *Paragraph* has higher accuracy than *MLP*.
- H2: *Tree* and *Paragraph* are quicker to validate than MLP.
- H3: *Tree* and *Paragraph* are more interpretable than *MLP*, measured by a 13-item Likert questionnaire introduced by [9].
- H4: Performance improvement by repeated evaluations of *Tree* is larger than of *MLP*.
- H5: Environment context of the decision-making increases interpretability.
- H6: The advantage of *Tree* and *Paragraph*'s subjective interpretability over *MLP* is domain-independent.

To test for the six hypotheses, we build an online survey with Qualtrics. In each question of the survey, we show a model (*Tree*, *MLP*, or *Paragraph*) and an input feature vector. The subject is asked to make predictions (i.e., compute the output of the model) given the respective input. We collect $N = 34$ responses. The average age of participants is 25.15 with a standard deviation of 4.28. Out of the 34 participants, 15 are male, and 19 are female.

7.13.1 User Study Results

For H1-H5, we illustrate results on the I-94 domain, and for H6, we show the results on both the Multi-lane Ring and I-94 domains to verify the conclusions hold for both environments. For all statistical tests, the assumptions for the ANOVA test are not satisfied. Thus, we instead perform a non-parametric Friedman test followed by a posthoc Nemenyi–Damico–Wolfe (Nemenyi) test [243].

H1-H3: We summarize the results for H1-H3 in Figure 7.11. As we test the user's simulation of outputs on each model three times and the action output is two-dimensional (acceleration and lane-changing), each user is validated across six action outputs for each model. We denote the number of accurate answers out of the six as the "score" in Figure 7.11 left. We observe that participants are able to simulate the outputs of the *ICCT* more accurately than an *MLP* ($p < .05$), supporting H1. Furthermore, the time spent on the *ICCT* and *Paragraph* to evaluate the output is significantly less than the time spent on MLP ($p < .001$), shown in Figure 7.11 (middle). *Tree* and *Paragraph* are also rated by users to have significantly higher interpretability ($p < .001$) (Figure 7.11 right). As such, the results from the user study support H1-H3, showing *Tree* and *Paragraph* are easier to simulate, quicker to validate, and more interpretable than neural networks. We find there is no significant difference across all three metrics between the tree form and the paragraph form of the ICCT.

H4: We show the results to test for H4 in Figure 7.12. We hypothesize that with practice, the ability of the users to evaluate the models may increase. However, Figure 7.12 shows that the accuracy on the first trial is the highest, and the time spent on the first trial is the lowest. Instead of the learning effect, the finding may be explained by a fatiguing effect, which causes the participants to have lower performance in later trials. Another hypothesis is that the first trial questions are relatively easy as they correspond to the early stages of the execution, where the

environment has fewer cars. The changes in accuracy score and time across three trials do not have a significant interaction effect with models, and therefore H4 is not supported. However, across repeated iterations, we observe that the score and the prediction time of the ICCT tree form and paragraph form are close while being much easier to simulate than an MLP.

H5: We illustrate the result for H5 in Figure 7.13. We observe that generally, for all three models, the condition with context results in slightly lower accuracy scores and interpretability scores. For *Tree* and *Paragraph*, the time spent with context is slightly higher than the condition without context. However, the time spent on MLP without context is higher than with context. One possible reason could be that tree depictions and paragraph descriptions are interpretable and quick to evaluate, and therefore context does not provide more benefit but introduces some workload overhead. For MLP, the context helps the user to understand the situation and therefore makes the evaluation faster. Overall, H5 is not supported as context does not provide a significant boost to subjective interpretability.

H6: The comparison of results between the two domains, Multi-Lane Ring and I-94, can be viewed in Figure 7.14. We observe that the results for H1-H3 are similar for both domains and therefore, H6 is supported by displaying the advantage of different representations of the ICCT's (both the *Tree* and *Paragraph* form) interpretability over an *MLP* is regardless of the two domains ($p < .05$ on accuracy score and $p < .001$ on both time and subjective interpretability).

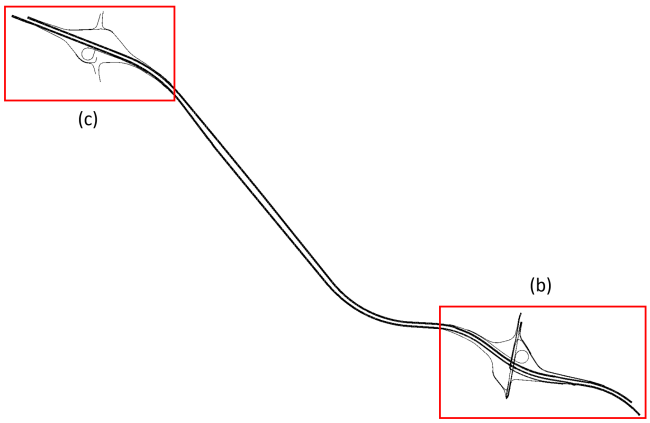
7.14 Conclusion

In this work, we present a novel tree-based model for continuous control, applicable to a wide variety of domains including robotic manipulation, autonomous driving, etc. Our Interpretable Continuous Control Trees (ICCTs) have competitive performance to that of deep neural networks across several continuous control domains,

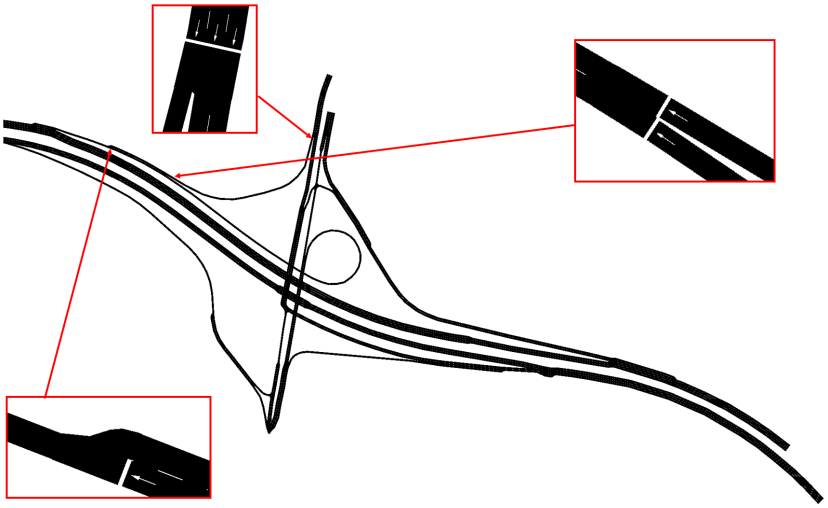
including six difficult autonomous driving scenarios and two driving domains grounded in realistic lane geometries, while maintaining high interpretability. The maintenance of high performance within an interpretable and verifiable reinforcement learning architecture provides a paradigm that would be beneficial for the safe real-world deployment of autonomous systems.

7.15 Limitations and Future Work:

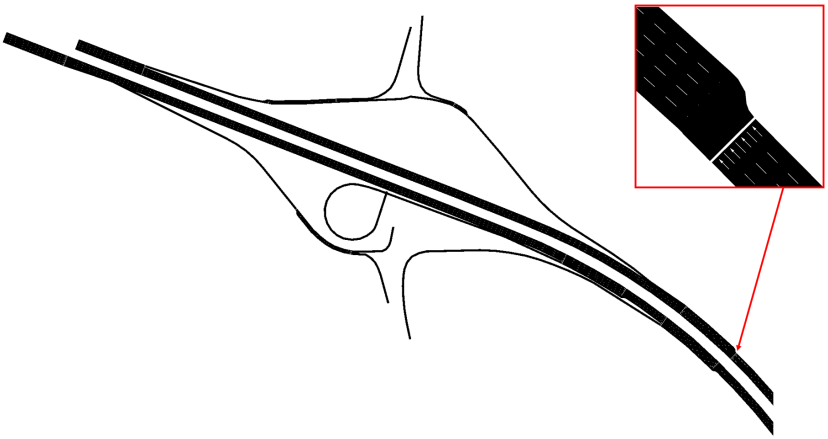
Our framework has several limitations. Continuous control outputs (e.g., predicting a steering angle) may not be interpretable to end-users and may require post-processing to enhance a user's understanding. Also, the relationship between controller sparsity, tree depth, and interpretability is not clear, making controller sparsity and tree depth difficult-to-define hyperparameters. We also note that in more challenging environments, larger ICCTs may be required to increase their representative power. In these instances, although the size of Interpretable Continuous Control Trees (ICCTs) may pose challenges for end-users in terms of interpretation, it is important to note that they can still be verified by experts and interpreted within specific tree sub-spaces.In future work, we will extend ICCTs to incorporate constraints from end-users, provide safety guarantees on our ICCTs, and reason about how the complexity of an ICCT may change as we move to higher-abstraction state spaces.



(a) I280-overview



(b) I280-begin



(c) I280-end

Figure 7.9: Figure 7.9a presents the overview of the I-280 domain. The ego vehicle is tasked to join the highway from the ramp and then exit the environment at the end of the highway. The ego vehicle's entrance ramp and exit is zoomed in and presented in Figure 7.9b and Figure 7.9c, respectively.

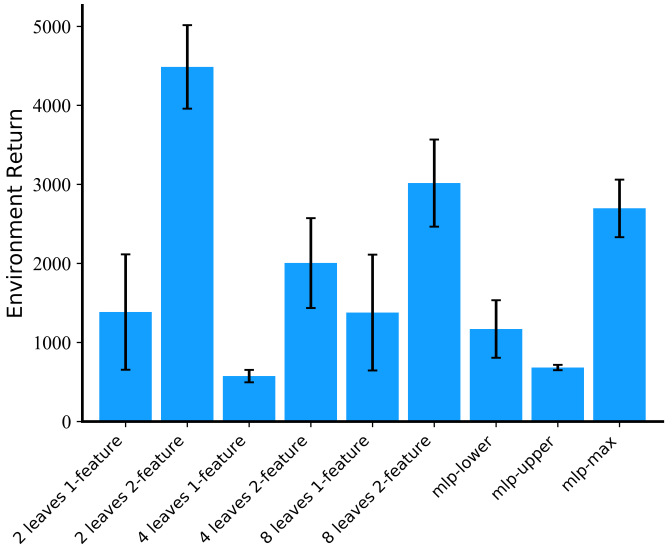


Figure 7.10: This figure compares the performance of ICCT agents and MLP agents in the I-280 domain. The environment returns for each model are displayed through a mean and standard deviation across ten evaluation episodes.

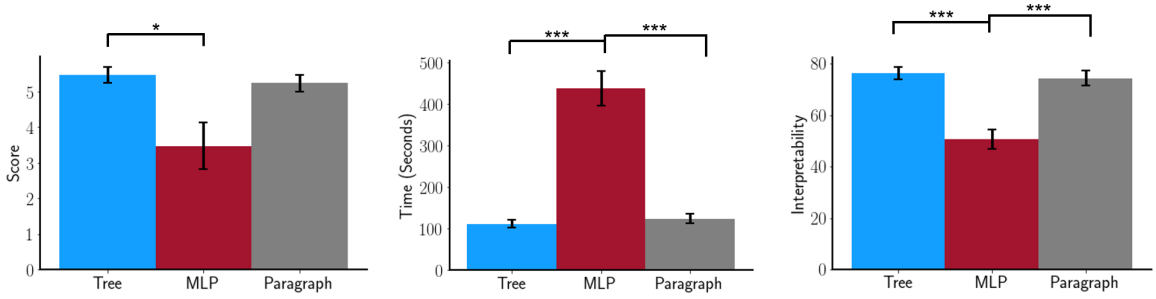


Figure 7.11: This figure shows the comparisons of accuracy score (left), time spent (middle), and subjective interpretability rated (right) across the three models in the I-94 user study. * denotes a significant difference of $p < .05$. *** denotes a significant difference of $p < .001$.

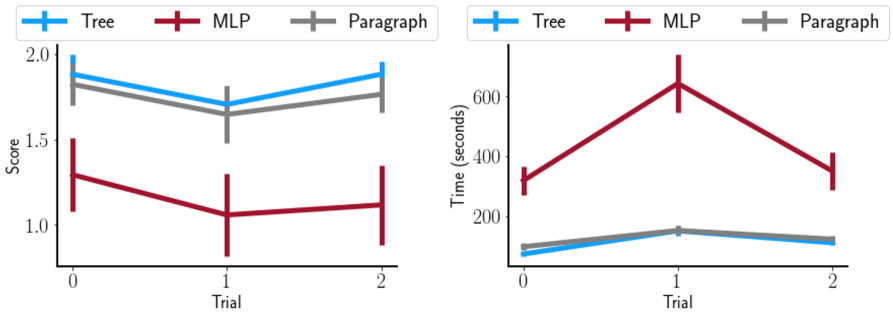


Figure 7.12: This figure shows the accuracy score (left) and time spent (right) changes in three repeats trials across the three models in the user study.

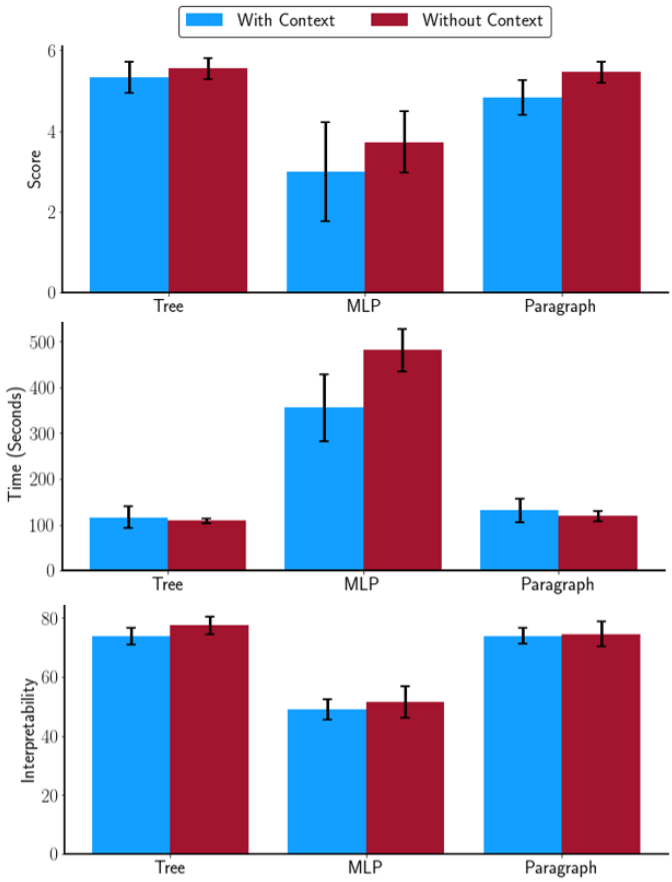


Figure 7.13: This figure shows the comparisons of accuracy score (left), time spent (middle), and interpretability rated (right) with or without context across the three models in the user study.

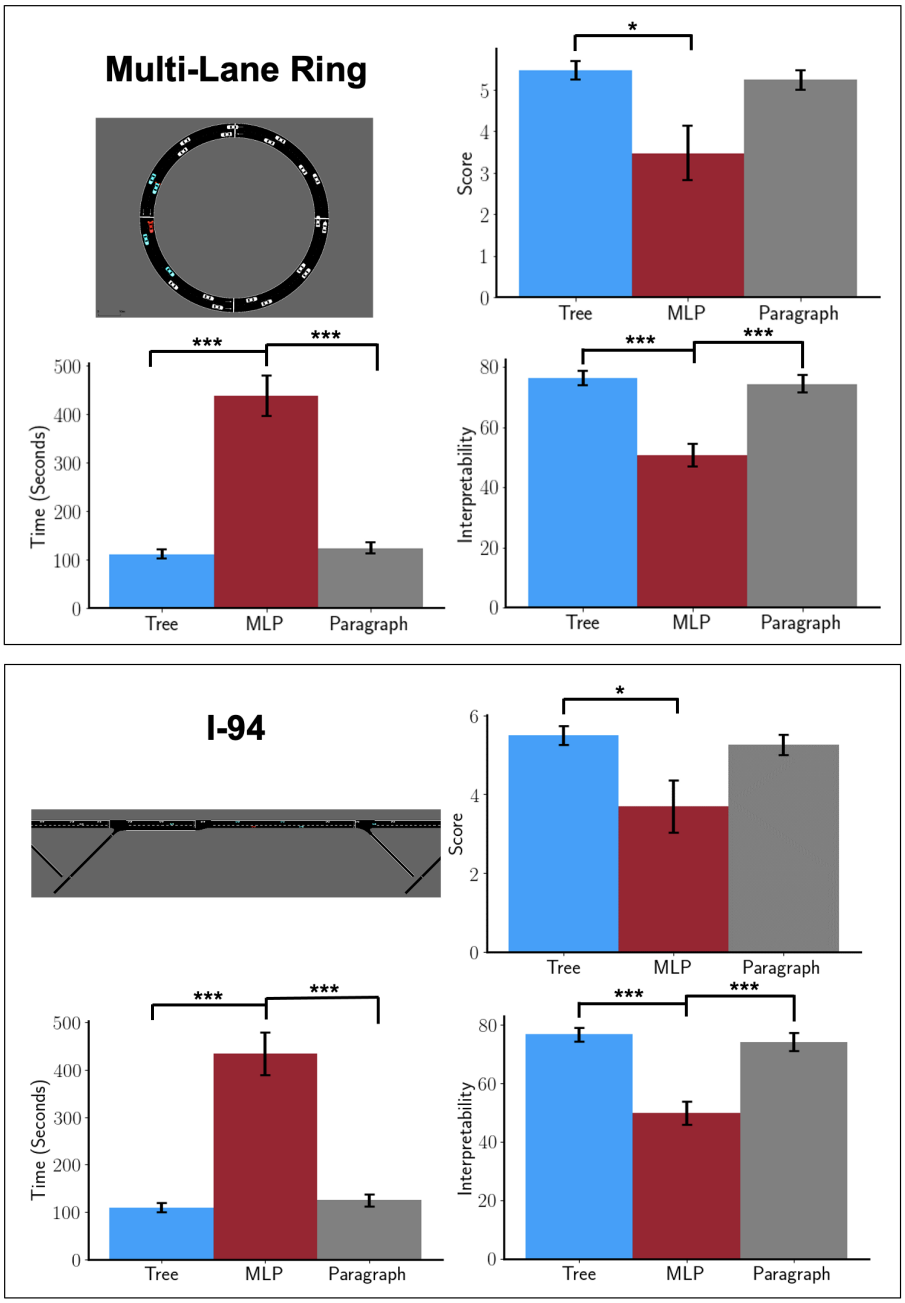


Figure 7.14: This figure shows the comparison of results between the Multi-Lane Ring domain and the I-94 domain across the three models in the user study. * denotes a significant difference of $p < .05$. *** denotes a significant difference of $p < .001$.

CHAPTER 8
THE UTILITY OF EXPLAINABLE AI IN AD HOC HUMAN-MACHINE TEAMING

In this chapter, we characterize the utility of Explainable AI techniques in Human-Machine Teaming. Importantly, I assess the ability for human teammates to gain improved situational awareness (SA) through the augmentation of xAI techniques and quantified the subjective and objective impact of xAI-supported SA on human-machine team fluency.

8.1 Introduction

Collaborative robots (i.e., "cobots") and machine learning-based virtual agents are increasingly entering the human workspace with the aim of increasing productivity, enhancing safety, and improving the quality of our lives [2, 1]. In the envisage of ubiquitous cobots, these agents will dynamically interact with a wide variety of people in dynamic and novel contexts. Ad hoc teaming characterizes this type of scenario, where multiple unacquainted agents (in this case, humans and cobots) with varying capabilities must collectively collaborate to accomplish a shared goal [244, 245]. Ad hoc teaming presents a significant challenge in that agents are unaware of the capabilities and behaviors of other agents, and lack the opportunity to develop a team identity, shared mental models, and trust [246, 149, 73]. For example, the human may not be aware of the cobot's possible actions and proclivities towards specific activities, limiting her ability to coordinate and plan effectively [247, 248]. Furthermore, this lack of understanding negatively impacts the user's ability to perform situational analysis, impeding a key component of the Observe-Orient-Decide-Act (OODA) loop [249]. For a human teammate to main-

tain situational awareness (SA) and effectively make decisions in human-machine teaming, the human must maintain an internal model of the cobot's behavior. However, to develop such an understanding of the cobot, the human may have to constantly observe or monitor the cobot's behavior, a costly and tedious process.

Effective collaboration in teaming arises from the ability for team members to coordinate their actions by *understanding* both the capabilities and decision-making criterion of their teammates [7, 6]. For human-human teams, [250] states that successful joint coordination among agents depends on the abilities to share representations, to predict other agents' actions, and to integrate the effects of these action predictions. Similarly, we believe these findings should correlate in human-machine teaming. Explainable AI (xAI) techniques, utilizing abstractions or explanations that provide the user insight into the AI's rationale, strengths and weaknesses, and expected behavior [251], can supply the human teammate a representation of the cobot's behavior policy and may assist in the human teammate's ability to predict and develop a collaboration plan. Furthermore, effective human-machine teaming requires the ability for team members to develop a shared mental model (i.e., team members share common expectations about the team coordination strategy, the outcomes of individual strategies, and the individual roles in achieving the team's objective [252]) [248]. xAI techniques offer the promise of enhancing team situational awareness, shared mental model development, and human-machine teaming performance.

Recent work in the machine learning community on xAI has emphasized the importance of interpretability, and post-hoc explainability in enhancing the prevalence of machine learning-based approaches [253]. Other work has even explored simulatability [9, 131], assessing a human's ability to observe a model (e.g., a decision tree, which lends itself to interpretability [34]) and be able to produce the correct output given an input feature. Augmenting machine learning-based sys-

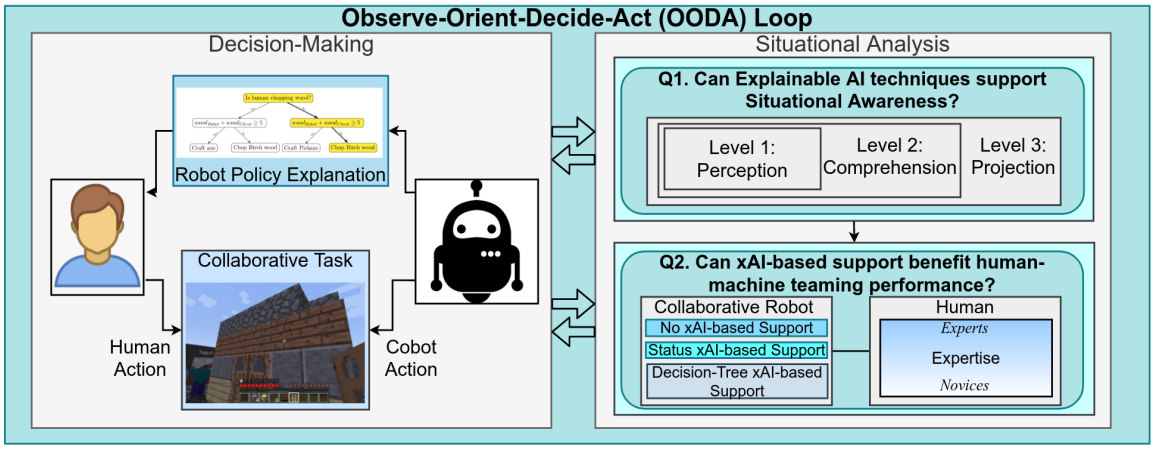


Figure 8.1: This figure displays an overview of our experimentation in relation to the Observe-Orient-Decide-Act (OODA) loop. On the left, we display the human-machine teaming interaction with both agents taking actions and the cobot outputting a policy explanation to the human teammate. On the right-hand side, we display the two questions assessed by our human-subjects experiments.

tems with some form of intepretability or simulatability can enable these systems to gain human trust [209, 210, 211], an essential quality in high-performance teaming [221]. While prior work has provided approaches for explaining machine behavior through natural language [148], interpretable decision trees [9], and attention-based focusing[254], the utility of collaborative agents augmented with explainable AI techniques in *human-machine teaming* has not been explored.

In this work, we present two novel human-subjects studies to quantify the utility of xAI in human-machine teaming. We assess the ability for human teammates to gain improved SA through the augmentation of xAI techniques and quantify the subjective and objective impact of xAI-supported SA on human-machine team fluency. We first assess if xAI can support the different levels of SA [255] by assessing how different abstractions of the AI's policy support SA within a human-machine teaming scenario. Here, users are presented visualizations of human-machine teaming gameplay and are tested on their understanding of the human and cobot's decision-making behaviors through an adapted Situation Awareness Global Assessment Technique (SAGAT) [256] questionnaire. Second, we study the

effect of augmenting cobots with online xAI and assess how different abstractions of the AI's policy affect *ad hoc human-machine teaming performance*. In Figure 8.1, we present an overview of our experimentation in relation to the Observe-Orient-Decide-Act (OODA) loop. We provide the following contributions:

1. We design and conduct a study relating different abstractions of the cobot's policy to their induced situational awareness levels, measuring how different explanations can help a human perceive the current environment (Level 1), comprehend the AI's decision-making model (Level 2), and project into the future to develop a collaboration plan (Level 3). Our results show that xAI techniques can support situational awareness ($p < 0.05$).

2. We design and conduct an ad hoc human-machine teaming study assessing how online xAI-based support, generated via cobot abstractions, and the human's ability to process higher levels of information affect teaming performance. We find novices benefit from xAI-based support ($p < 0.05$) but are susceptible to information overload from more involved xAI abstractions ($p < 0.05$). Expert performance, on the other hand, degrades with the addition of xAI-based support ($p < 0.05$), indicating that the cost of paying attention to the explanation outweighs the benefits obtained from generating an accurate mental model of the cobot's behavior.

8.2 Human-Machine Teaming Domain

For our experiments investigating the deployment of xAI techniques in human-machine teaming, we utilize the Microsoft Malmo Minecraft AI Project [72]. Minecraft is an open-world environment where players can build structures, craft tools, and play alongside other individuals. Crafting within Minecraft can be classified as a hierarchical task that requires obtaining base materials before generating a more

complex object/tool and may require traveling to a crafting table. Building structures is also a hierarchical task as lower layers must be (partially) constructed before the upper layers of a structure can be built. Collaboration among other individuals is highly complex as high-performance teaming requires intent recognition, task coordination, resource/tool sharing, among other personal factors such as game proficiency and trust among teammates.

We generate a 61×61 grid utilizing the Python API provided by [72]. Within this grid is two agents, the human and the cobot. The human plays with standard Minecraft controls, utilizing both a keyboard and mouse to continuously move and change her field of view. This teaming scenario is notably different from prior human-machine teaming experiments [73] that restrict human motion to discrete cardinal movements. The cobot is also programmed in continuous space with several action primitives further described below.

Planning within this domain is challenging as agents take macro-actions that can take an arbitrary number of time-steps to execute which induces asynchronicity, closely resembling the complexities associated with that of solving a Decentralized Partially Observable Semi-Markov Decision Process (Dec-POSMDP) [257].

Within this domain, the human is unable to explicitly communicate with the cobot (e.g., through chat), and the cobot is able to communicate with the human through a policy abstraction depending on the experiment factor (cf. section 8.4). The human also has access to a first-person view of the cobot's screen, providing partial knowledge of the cobot's current location, cobot's current resources, cobot's current action. The cobot has complete access to the human's global state, including current resources, location, and approximate action. The cobot and human are able to share resources through a chest located at a static position, with both agents able to store and take resources depending on need. We provide additional environment dynamics within the supplementary material.

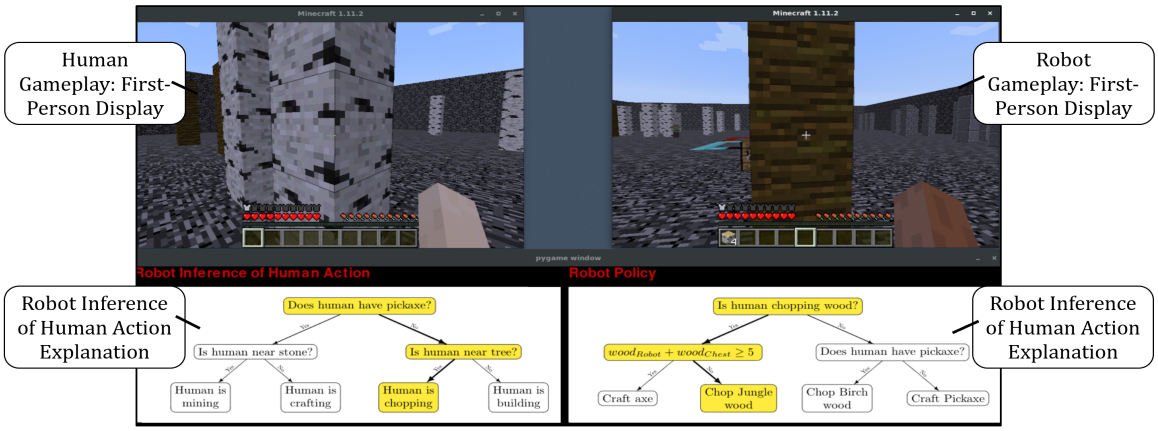


Figure 8.2: This figure displays a sample gameplay image where the cobot is augmented with the decision-tree explanation. Note this shows IV1:SA1-2-3 condition and IV2:Display Cobot Inference of Human Policy and Cobot Policy in section 8.4.

As shown in Figure 8.2, the human will see up to four elements on her screen. On the top-left, the participant will play Minecraft. On the top-right, the participant has a display showing the first-person view of the cobot. On the bottom of the screen, we utilize the Pygame interface to display information regarding the cobot's policy. On the bottom-left, the participant has a display that may show information regarding the cobot's inference of her action (i.e., the cobot's inference of the human's behavior). On the bottom-right, the participant has a display that may show information regarding the robot's policy (i.e., how the cobot chooses an action).

8.2.1 Human-Machine Collaborative Task

The human and cobot are assigned to build a house with certain specifications. The 4-level house contains eight unique objects including two types of planks, two types of stone, doors, stairs, a fence, and fence gates totaling 89 objects. The latter four items are crafted materials that require combining base materials. Each agent has unique capabilities. The cobot has the ability to collect resources and craft certain items. The human is made aware of all possible behaviors of the cobot.

However, the cobot cannot help the human build. The human player is able to collect resources *and* is in charge of building the house. The human is told a priori that her goal is to "Collaborate with the AI as best as possible to complete the house as quickly as possible." However, the human is not strictly specified to build or collect resources in any order.

Cobot Policy – The cobot maintains a hierarchical policy. A low-level policy determines the cobot's action, and a high-level policy determines the cobot's inference of the human's action (which the low-level policy is conditioned upon). Both components of the hierarchical policy are decision tree-based policies of depth two and with four leaf nodes. The cobot's high-level policy stays fixed throughout gameplay and is displayed in the bottom-left of Figure 8.2. The cobot's low-level policy changes throughout gameplay across five phases of play, correlating with different periods of the task (i.e., building different portions of the house). A depiction of one low-level policy in displayed on the bottom-right in Figure 8.2. Across all policies, the cobot has the ability to 1) Chop Birch Wood, 2) Chop Jungle Wood, 3) Mine Andesite Stone, 4) Mine Cobblestone, 5) Put resources in chest, 6) Craft pickaxe, 7) Craft axe, 8) Craft sticks, and 9) Craft planks.

The cobot's behavior loop is to observe the human behavior and perform inference of the human's action. Following the inference of the human's behavior, the cobot determines its action based on the current tree-based low-level policy. Once the action has been completed, the cobot will deposit its excess resources into the chest and begin the loop again.

The cobot deposits any materials that have a quantity above two into the chest for the human to use for building/crafting and any tools with a quantity above one into the chest. The cobot's movement policy (contained within each macro-action) is to move in a straight-line path towards the object. If the agent meets an obstacle and/or takes longer than 9 seconds to reach its goal, the cobot will teleport to the

object. For "Chop" or "Mine" actions, the cobot will perform the chopping/mining action until each block of the target has disappeared (detected using Line of Sight variables).

We conducted an initial IRB-approved pilot study with 30 participants to fine-tune the thresholds within the cobot's policy to enforce that the cobot is highly collaborative. We note that while the cobot policy models are hand-designed, the decision-tree based policies serve as a surrogate for the interpretable models produced by interpretable machine learning-based approaches [258, 9].

8.3 Study 1: Relationship Between Explanations and Situational Awareness

Here, we present the details of our first human-subjects study relating different xAI techniques to the three situational awareness levels specified by [255]. We explore the question:

- **Q1: Can xAI techniques be used to support situational awareness (SA)?** If so, which levels of SA are enhanced by xAI?

We review the different levels of situational awareness, study conditions, design, and procedures, and describe the measures employed.

8.3.1 Situational Awareness

Situation awareness (SA) represents a user's internal model of the world around her at any point in time [255], representing the user's ability to perceive the elements of the environment, comprehend their meaning, and project their status in the near future. Situational awareness plays a significant role in mental model alignment among teams and can largely impact performance [259]. Following [255, 260], the three levels of situational awareness induced through xAI techniques are as follows:

- **Level 1: Perception** - Explanation of the current state of the world.
- **Level 2: Comprehension** - Explanation of the agent's current decision-making.
- **Level 3: Projection** - Explanation enabling user to predict future behavior.

The degree to which SA is maintained is typically assessed through the Situation Awareness Global Assessment Technique (SAGAT). The SAGAT prompts a user at a random point within a simulation with a series of fact-based questions to determine her knowledge while blanking out the simulation display.

Situational Awareness Assessment – We conduct a SAGAT questionnaire following the recommendations of [260, 261] that provide considerations for adapting the SAGAT towards evaluating situational awareness in human-machine teaming. Questions for Level 1,2 and 3 consisted of those that determine the current state and action of the human and the cobot, determine the human's understanding of the cobot's current capabilities and features considered in the cobot's decision-making policy, and assess the user's ability to predict the cobot's next action, cobot action given modified inputs, and input requirements to produce a favorable action. Similar to [255], we maintain a sample of questions for each level of situational awareness. We provide the user three questions per situational awareness level, restricting each question to multiple choice with 2-4 answer choices. We provide the complete list of questions for each SA level within the supplementary material.

While the SAGAT can provide insight into a user's SA, this test is highly intrusive as it requires task interruption. These interruptions can be distracting, interfering with performance [262], increasing stress [263], and likelihood of failure [264]. Accordingly, we choose to assess user SA and the effect of SA separately to reduce intrusiveness in the team fluency study and contribute a novel, targeted SA study across xAI techniques.

8.3.2 Experiment Conditions and Procedures

In this study, we seek to determine how different abstractions of the AI's policy induce different levels of situational awareness. As such, we utilize a 1×3 within-subjects design varying across three abstractions: 1) No explanation of the robot's hierarchical policy, 2) A status explanation of the cobot's hierarchical policy, and 3) A decision-tree explanation of cobot's hierarchical policy.

- **No Explanation –** In this condition, the human is given no information about the cobot's policy. We note that even within the no explanation condition, the human still has access to the first-person displays of both the robot and human.

- **Status Explanation –** In the status explanation, the human is given information about the cobot's policy relating the cobot's hierarchical policy outputs. The information is given in the form of a short phrase representing the information within the leaf node of the decision-tree explanation.

- **Decision-Tree Explanation –** In the decision-tree explanation, the human is given a decision tree representing the cobot's policy with active edges and decision nodes highlighted. The decision trees displays complete information about the state features that the cobot uses to make decisions and the choice of action. We display the sample game display of this condition in Figure 8.2.

The experiment was conducted through an online platform where the first page provided the participants with introductory information regarding the human-machine teaming task. Participants are informed that the survey will take approximately one hour, the experiment is completely voluntary, and that the participants will be compensated $20 for the study. Users will then conduct three episodes in which each episode corresponds to a different factor. Each episode consists of

≈10 minutes of gameplay that users must view interrupted by the SAGAT at two-minute intervals (five trials within each episode). The factor/episode ordering was randomized to mitigate confounds.

8.3.3 Results

We recruited 48 participants in an IRB-approved experiment, whose ages range from 18 to 58 (Mean age: 21.69; Standard deviation: 5.61; 18.75% Female). We analyze our data using a repeated-measures ANOVA for the omnibus significance and a Tukey HSD for pairwise comparisons, which includes Bonferroni corrections. Assumptions of normality and homoscedasticity are tested with Shapiro-Wilke and Levene's test respectively. We found that only our Level 2 data failed to meet these parametric assumptions, and thus a Friedman's test with Nemenyi post-hoc (with corrections) was employed instead. We display our findings in Figure 8.3a. We provide additional analysis details within the supplementary material.

Q1: Overall, we find that xAI techniques support higher levels of situational awareness. No significant difference was found in the ability for users to perceive their environment (Level 1) with and without xAI-based support. This indicates that access to a first-person view of teammate gameplay and one's own gameplay is sufficient for perception of the environment. For Level 2 SA, a Friedman's test found significance ($\chi^2(2) = 34.2; p < 0.001$) with the pairwise results showing both status xAI-based support ($p < 0.05$) and decision-tree xAI-based support ($p < 0.001$) significantly increases the ability for the user to comprehend her environment compared to cobots without xAI-based support. Lastly, we also observe an omnibus significance in the Level 3 ($F(2, 94) = 4.01; p < 0.05$), and find that the decision-tree xAI-based support enhances the user's ability to project cobot's behavior into the near future ($p < 0.05$). These results support the hypothesis that xAI techniques enhance SA in human-machine teaming.

8.4 Study 2: Situational Awareness in Ad Hoc Human-Machine Teaming

In this section, we present the details of our second human-subjects experiment exploring:

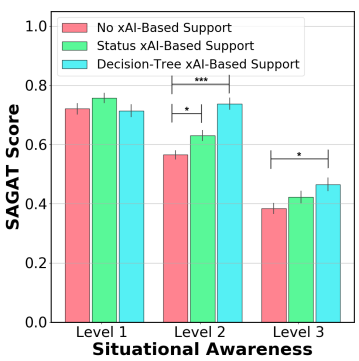
- **Q2: How does xAI-based support affect ad hoc human-machine teaming performance?**
- **Q3: How does the amount of information regarding the cobot's policy affect ad hoc human-machine teaming performance?**

In Study 2, a participant and a cobot are asked to complete the human-machine collaborative task defined in subsection 8.2.1. We set an exclusion criteria to allow only those with a minimum of 20 hours of lifetime experience in Minecraft to participate. The exclusion criteria removes participants who are unfamiliar with the gameplay controls and ensures that participants are not actively learning to play Minecraft while attempting to collaborate with a cobot they are unfamiliar with. We note that the collaboration between the human and cobot is highly unstructured, and the human's behavior can range from choosing to ignore the cobot to actively perturbing the cobot to understand its behavior more clearly. Below, we review the experimental conditions, design, procedure, and describe the measures employed.

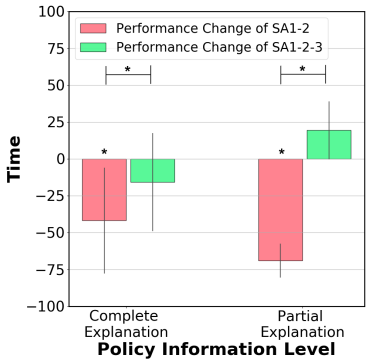
8.4.1 Experiment Conditions

We conduct a 2×3 mixed between-/within-subject design with two factors: **IV1:** Induced SA (subsection 8.4.1) and **IV2:** Policy Information (Figure 8.4.1).

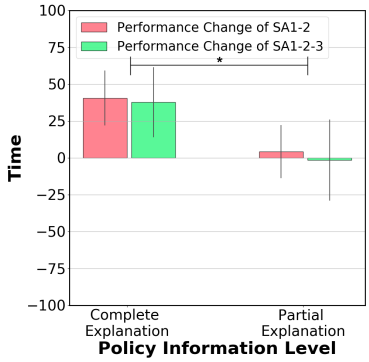
IV1: Induced Situational Awareness Levels – The situational awareness levels are aligned via our first study detailed in section 8.3. We vary across 1) **SA1: No Explanation**, in which the human is able to perceive the environment, 2) **SA1-2: Status Explanation**, in which the human is able to perceive and comprehend the



(a) Study 1 Findings



(b) Study 2 Novice Findings



(c) Study 2 Expert Findings

Figure 8.3: This figure represents the findings of Study 1 (a) and Study 2 (b-c). Figure 8.3a displays the SAGAT scores across SA levels and xAI abstractions. Figure 8.3b and Figure 8.3c display the performance residuals (inverse scale: lower is better) with xAI-based support across policy information levels with respect to the no-explanation condition for novices (Figure 8.3b) and experts (Figure 8.3c).

environment, and 3) **SA1-2-3: Decision Tree Explanation**, in which the human is able to perceive, comprehend, and project the environment.

IV2: Policy Information Level – Here, we describe the policy information levels that vary between subjects. Specifically, complete information provides users assistance with Theory of Mind (ToM) [265], reasoning to strategically reason about one's actions in the context of the cobot's decision-making. However, the additional information provided for supporting ToM reasoning may come at the cost of increased complexity and risk of information overload.

- **Condition 1: Display Cobot Inference of Human Policy and Cobot Policy**
  - The cobot displays both its low-level and high-level policy, providing the

user with increased information regarding its decision-making strategy.

- **Condition 2: Display Cobot Policy –** The cobot displays only its low-level cobot policy. This condition provides partial information of how the cobot makes decisions, leaving out the cobot's inference of the human policy.

8.4.2 Procedure

Participants are first informed that the experiment will take approximately 1.75 hours, the experiment is completely voluntary, and that the participants will be compensated $10 per hour of the study. The participant is first briefed on the objective of the experiment, to investigate the impact of online explanations in human-machine teaming. The participant is told that she will be playing with three *different* autonomous agents (use of deception). The participant starts with a pre-experiment survey collecting demographic information, gaming background (experience with video games and Minecraft), and the Big Five Personality Questionnaire [266]. Afterwards, the participant is handed a instructional document regarding specifications for the house and some notes regarding the cobot's behavior. Next, the participant conducts a brief hand-crafted tutorial in Minecraft. Once completed, the participant is tasked with individually building the entire house. The individual house build helps the human gain familiarity with the house specifications, which benefits in the user's task understanding. Once completed, the primary experimentation begins. Users will conduct three episodes in which she will play with three cobots, each of which are programmed the same but with varying xAI-based support and a randomly selected level of policy information. The ordering across xAI-based support levels is randomized to mitigate potential experimental confounds, such as learning effect, fatigue, etc. In each episode, the participant will first be given a sample ≈30-second video that describes the upcoming visualization (Figure 8.2). Following the video, the participant is given a

written tutorial describing the xAI-based support. Once completed, the participant will build the specified house in Minecraft with the cobot, a timed task. To conclude the episode, we administer several post-study scales to support our quantitative findings including the Human-Robot Collaborative Fluency Assessment [267], Inclusion of Other in the Self scale [268], Godspeed Questionnaire [269], NASA-TLX Workload Survey [270]. Each scale has been verified for validity in prior work and is used to assess the quality of the human-machine teaming interaction. The Human-Robot Collaborative Fluency Assessment [267] measures team fluency, cobot contribution, trust, positive teammate traits, and perceived improvement through several sub-scales. The Inclusion of Other in the Self [268] scale measures the perceived closeness between teammates. The Godspeed Questionnaire [269] is used to measure perceived likability and perceived intelligence. The NASA-TLX [270] measures the task workload. We provide additional details regarding our procedure in the supplementary material.

8.4.3 Results

We recruited 30 participants under an IRB-approved protocol, whose ages range 18 to 23 (Mean age: 19.97; Standard deviation: 1.30; 16.67% female), with participants randomly assigned to each of the factor levels of **IV2** (the between-subjects variable) with 15 subjects per level. We evaluate participant performance by using the time taken for the human-machine team to finish building the house. Our data was modeled as a mixed-effects ANOVA to capture any possible relation between independent variables across **IV1** and **IV2**. We test for normality and homoschedascity in our data, and employed a corresponding non-parametric test if the data failed to meet these assumptions. We provide details regarding our analysis within the supplementary. We display our objective findings in Figure 8.3b and Figure 8.3c.

Q2: We conduct a preliminary meta-analysis on our participant data and find

two distinctive clusters in participants' individual build times (the separate, pre-test calibration task), implying two distinct categories of participants that vary in gameplay speed. We dichotomize our subject data with an additional categorical variable within our analysis representing each cluster, the first referring to "experts" (i.e., those with higher proficiency in the game of Minecraft) and "novices" (i.e., those with lower proficiency in the game of Minecraft). We find that there are 17 experts and 13 novices within this dataset. While 13 are characterized as novice, it should be noted that our exclusion criteria serves to ensure that all participants are familiar with the gameplay controls of Minecraft and should thus be able to accomplish all key components of the teaming task. With respect to our separate calibration task, we find a significant effect between a participant's teaming ability with the cobot and the participant's individual build speed ($F(1, 26) = 23.5; p < 0.001$).

Novice performance significantly differs across levels of xAI-based support ($F(2, 22) = 4.67; p < 0.05$). We find that novices working with cobots augmented with status xAI-based support (*SA1-2*) are able to complete the human-machine teaming task significantly quicker than users that did not have xAI-based support (i.e., the baseline condition) ($p < 0.05$). While we see that xAI-based support in the form of a short phrase is beneficial, more cognitively intense xAI-based support (decision-tree based support *SA1-2-3*) does not provide novices with benefits in performance. Status xAI-based support *SA1-2* led to significantly improved performance compared to decision-tree xAI-based support (*SA1-2-3*) for novices ($p < 0.05$). Conversely, we find experts given xAI-based support have significantly increased time to build compared to those without xAI-based support, indicating a performance decrease ($t(33) = 2.09; p < 0.05$). We provide further discussion in section 8.5.

Q3: Expert performance significantly differs across policy information levels

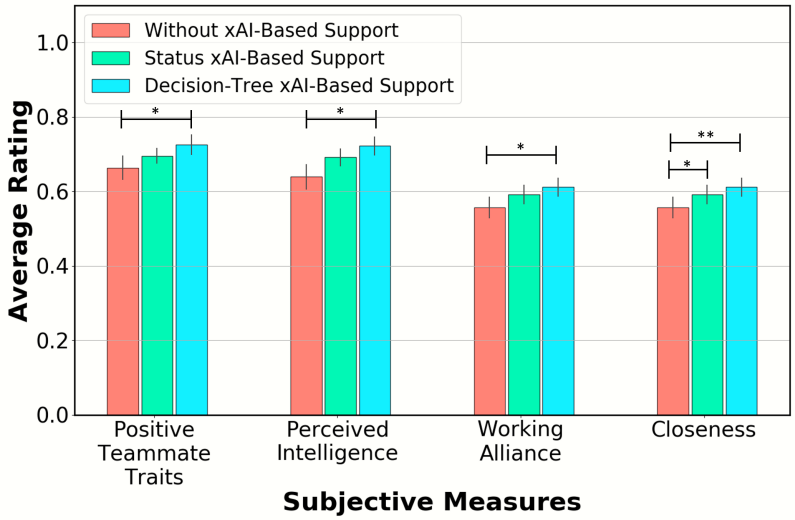


Figure 8.4: This figure represents the normalized subjective findings of Study 2. We see that all users find cobots with decision-tree xAI-based support to maintain more positive teammate traits, maintain a better working alliance, and are perceived as more intelligent than cobots without xAI-based support. Users also perceive both cobots with status xAI-based support and those with decision-tree xAI-based support as more close than cobots without xAI-based support.

($F(1, 15) = 5.27; p < 0.05$). We find that experts working with cobots augmented with partial information (only the cobot's low-level policy) were able to complete the human-machine teaming task significantly quicker than users with complete information of the cobot's hierarchical policy ($p < 0.05$). We find no significant difference in novice performance across information levels (**IV2**).

Subjective Findings: While experts and novices interacted differently with cobots, the subjective findings were similar and are presented as an aggregate. Here, we report on subjective measures that yielded statistical significance. We provide a full analysis within the supplementary material.

Q2: We find that users assess positive teammate traits, team working alliance, closeness, and perceived intelligence significantly differently across xAI-based support ($F(2, 56) = [3.16, 4.08, 7.29, 5.31]; [p < 0.05, p < 0.05, p < 0.01, p < 0.01]$). Users rate a cobot with decision-tree xAI-based support as significantly more positive than a cobot with no explanation ($p < 0.05$) and rate human-machine teams with

decision-tree xAI-based support with higher working alliance scores than human-machine teams without xAI-based support ($p < 0.05$). We also find that users perceive cobots with decision-tree xAI-based support as significantly more close than cobots without xAI-based support ($p < 0.01$) and cobots with status xAI-based support as significantly more close than cobots with no xAI-based support ($p < 0.05$). In assessing the perceived intelligence of the cobot, we conduct a Friedman's test and find that users perceive cobots with decision-tree xAI-based support as significantly more intelligent than cobots with no explanation ($p < 0.01$). We provide a depiction of our results for **Q2** in Figure 8.4.

Q3: When displaying partial information **IV2: Condition 2**, we find that users trust cobots and rate cobots with higher improvement scores across xAI-based support ($F(2, 28) = [3.55, 5.14]; [p < 0.05, p < 0.05]$). We find users trust cobots augmented with decision-tree xAI-based support in comparison to a cobot augmented with no xAI-based support ($p < 0.05$) and human-machine teams consisting of a cobot with decision-tree xAI-based support are rated with higher improvement scores than human-machine teams consisting of cobots with no xAI-based support ($p < 0.05$).

8.5 Discussion

In Study 1 (section 8.3), we start by exploring if xAI techniques can support situational awareness in human-machine teaming. We find objectively that xAI techniques can support SA. Specifically, we see that providing users with status explanations supports the ability to comprehend AI behavior (Level 2, $p < 0.05$). More so, decision-tree based support provides users with the ability to both comprehend AI behavior (Level 2, $p < 0.001$) and project the cobot's behavior into the near future (Level 3, $p < 0.05$). These results provide promising evidence supporting the deployment of xAI-based support in human-machine teaming.

In Study 2 (section 8.4), we assess how cobots augmented with xAI-based support subjectively and objectively affect ad hoc human-machine teaming performance. Novices benefit from a cobot augmented with status xAI-based support ($p < 0.05$) but do not benefit similarly from a cobot augmented with decision-tree xAI-based support. The benefit achieved through status explanations (*SA1-2*) indicates that novices are able to use the support to develop a shared mental model that benefits them. However, the more cognitively intense decision-tree xAI-based support does not provide any benefit, noting that such an xAI technique is not suitable for novices. Expert performance, on the other hand, degrades with the addition of xAI-based support, suggesting that the support serves as a distraction. As experts are more experienced with the game of Minecraft, we hypothesize that the cost of paying attention to the xAI-based support outweighs the benefits obtained from generating an accurate shared mental model. As the cobot is programmed to be collaborative, the cobot behavior may already fall reasonably within the expert's shared mental model, and the support, while providing an accurate description of the cobot's policy, may reduce the user performance. This hypothesis is supported by the further reduction in performance when experts are presented with complete xAI-based support as opposed to partial (**IV2**, $p < 0.05$). Thus, xAI-based support may not be universally beneficial and depends on the composition of the team.

While the performance benefits vary across experts and non-experts, we see that all users find cobots with decision-tree xAI-based support to be more trustworthy, capable of learning, maintain more positive teammate traits, maintain a better working alliance, and be perceived as more intelligent than cobots without xAI-based support. Given the results of our study, we wish to provide some key takeaways for future research in xAI and Human-Machine Teaming (HMT): 1) The addition of xAI techniques can induce SA, an important element of the OODA loop, 2) It is important to account for heterogeneity within both the cobot design [199]

and the xAI-based support, and 3) Information overload can exist as an impedance in human-machine teaming and support strategies should consider the cognitive bandwidth of humans.

Limitations – In our teaming scenario, the cobot has specific capabilities and does not have equal capability to the human. Our findings are limited to teaming scenarios with similar heterogeneous agents. Our experiment is limited to two broad classes of interpretable models, representing xAI approaches that can generate decision trees and those that can generate a short phrase representing policy information. These abstractions can be generated via a number of methods, including [9, 258]. We also did not explicitly vary the explanation size (e.g., depth of the decision tree, specificity of cobot status, etc.) in our experiment. Increasing the size/complexity of xAI-based support and accordingly modifying the training time is an important future direction to explore, as there is a tradeoff between the utility of xAI-based support and its complexity.

8.6 Conclusion

In this paper, we present findings relating xAI and situational awareness in human-machine teaming. Within our first study, we find that cobots with xAI-based support can provide human teammates a higher level of SA, benefiting a human teammate's ability to perform situational analysis and understand the human-machine teaming scenario. Within our second study, we find that novices working with cobots augmented with status xAI-based support gain significant performance benefits ($p < 0.05$) within the ad hoc human-machine teaming scenario compared to cobots without xAI-based support. Expert performance, on the other hand, degrades with the addition of xAI-based support ($p < 0.05$), indicating that the cost of paying attention to the xAI-based support outweighs the benefits obtained from generating an accurate shared mental model. These results display the benefits and drawbacks of deploying xAI in ad hoc human-machine teaming and provide

interesting future directions for xAI-based support in human-machine teaming.

Broader Impact – Our work has the potential to benefit future research in deploying xAI in human-machine teaming and advance the envisage of democratizing cobots, focusing on developing higher levels of understanding between humans and cobots. We note that it is important to account for the differences across humans, such as expertise. In other scenarios, it may be important to take into account other demographic information.

CHAPTER 9

TEAM DEVELOPMENT IN HUMAN-MACHINE TEAMING

9.1 Introduction

Successful human-machine teaming (HMT) has long been sought after for its wide variety of potential benefits, ranging from applications such as robotic healthcare aides that can provide doctors with a helping hand [21], to collaborative robots assisting humans with object assembly (i.e., co-assembly) [271]. While promising, achieving fluent HMT is challenging because interactions with humans can be incredibly complex due to the diversity across users [272], human teammates require explainable systems to support the development of mental models [11], and the lack of bidirectional communication (i.e., unclear how humans can "tell" a machine online to perform a desired collaborative behavior) [273]. In this work, we provide a potential solution to these challenges by allowing humans to modify a collaborative AI's interpretable policy representation across repeated iterations of teaming, enabling end-users to create personalized AI teammates to support them.

Recently, data-driven techniques (e.g., reinforcement learning) have become popular in HMT, allowing for the generation of collaborative agent behavior without cumbersome manual programming [74, 73]. In one approach, training with diverse simulated partners resulted in robust agents capable of cooperating with humans without prior human data [74]. However, a drawback of this prior work is the use of opaque (black-box) models, limiting human's ability to develop a shared mental model online and maintain situational awareness [176, 274], crucial for high-performance teaming [24, 260].

Furthermore, collaborative interactions with machines have often been rigid,

lacking the ability to effectively learn with and adapt to human teammates in real-time [43]. In ad hoc human-human teams (i.e., those without prior training), effective teaming is often developed through an iterative process, going through several stages before the team arrives at a fluent collaboration [44]. Bi-directional communication is often a key component of this process, enabling the development of successful coordination strategies [28]. In our work, we build towards adaptive, effective HMT by creating a pathway of bi-directional communication, utilizing interpretable policy representations as a mechanism to allow users to understand their machine teammates and allowing for explicit teammate policy modification through an interface (users can modify the machine's policy via a Graphical User Interface).

In this work, we first characterize prior work in HMT [73, 74], finding that machine behavior is unable to adapt to human-preferred strategies, and that high performance is typically driven by independent machine actions rather than collaboration, which can ultimately result in a higher team score. Next, we create a novel InterpretableML architecture to support the creation of interpretable cooperative agent policies via reinforcement learning (RL) and a GUI to allow users to modify the AI's behavior to their specifications. This capability is promising, enabling end-users to "go under-the-hood" of machine learning models and tune affordances or interactively and iteratively reprogram behavior. Finally, we conduct a 50-participant between-subjects user study assessing the effects of interpretability and policy modification across repeated interactions with an AI. Objectively, we find that white-box models underperformed black-box models from prior work in teaming with humans. This is confounded by black-box models being easier to train, and thus producing higher-performance agents. Furthermore, we find that 1) users found low-performance black-box models incredibly frustrating, 2) white-box teaming supported with interactive modification outperforms white-box

approaches alone, and 3) users that were extroverted, had familiarity with decision trees, and were experienced in gaming were better able to create high-performing personalized AI teammates. Lastly, as teaming can develop across different stages of Tuckman's Model [44], we relate trends in HMT performance variation across repeated gameplay to this model. Given these findings, in the future, to absolve the gap in performance between individualized coordination and successful HMT collaboration, researchers must focus on developing better interpretableML approaches to support the generation of high-performance white-box teammates, the modality of communication between agents and humans (tree policies may be difficult to understand for some users), and effective mixed-initiative interfaces that allow users, who may vary in ability and experience, to interact and improve team behavior.

9.2 Preliminaries

In this section, we introduce relevant work in HMT and Explainable AI (xAI), our HMT testbed, Overcooked-AI, Tuckman's Model, and the modeling framework we use to create collaborative AI agents, Markov Games.

Human-Machine Teaming – The field of HMT is concerned with understanding, designing, and evaluating machines for use by or with humans [140]. This growing field has recently attracted much attention from researchers, aiming to facilitate better collaborative performance between humans and machines. A common technique that has been used to produce collaborative AI agents is Deep Reinforcement Learning [275], where researchers have concentrated efforts on reducing the dissimilarity between synthetic human training partners and testing with human end-users. Approaches that have achieved success in the past include utilizing human gameplay data to finetune simulated training partners to behave more human-like [73], which can be expensive, and training with a diverse-skilled

population of synthetic partners to create an agent that can better generalize to non-expert end-users [74]. However, as we find in later sections, this training scheme may be biased towards training AI teammates to exhibit individualized strategies. As approximately a third of the diverse-skilled population of agents used in training are completely random agents, the teammate agent must compensate and exhibit individualized behavior.

In our work, we explore a paradigm where a user can directly modify and visualize a tree-based AI teammate she is interacting with. While prior work [276, 277, 278, 279] has explored similar paradigms, creating a system that allows end-users to modify robot policy trees, none have attempted to utilize tree-based models in an HMT setting and deployed these models within a repeated teaming scenario.

Overcooked-AI – Overcooked-AI [73] has become a common testbed to evaluate human-AI interaction. Here, two agents are tasked with creating and delivering as many soups as possible within a given time. Achieving a high score within this collaborative domain requires agents to navigate a kitchen and repeatedly complete a set of sequential high-level actions, including collecting ingredients, placing ingredients in pots, cooking the ingredients, collecting a dish, getting the soup, and delivering it. Both players receive the same score increase upon delivering the soup. *We modify the original Overcooked-AI game to be a simultaneous-move game as opposed to the original formulation of allowing agents to perform actions asynchronously as done in previous work.* This modification prevents the collaborative score metric from being dominated by super-human AI speed, causing the overall score to be more reliant upon effective collaboration and strategy. We provide further details about the domain in the supplementary material.

State-Space: Our interpretable agent policies reason over a semantically meaningful feature space as opposed to pixel space. Our feature space details the objects each

agent is holding, pot statuses, and counter objects.

Action-Space: Instead of using the lower-level actions utilized in Overcooked-AI, we allow the AI agent to utilize macro-actions that can accomplish high-level objectives such as ingredient collection, ingredient placement, and soup serving. Macro-actions are planned at the low-level using an A* planner, and we perform dynamic replanning at each timestep. Prior work has shown macro-actions enhance interpretability [280].

Tuckman's Model – [44] describes the different stages that a team goes through before reaching high performance, including "Forming", "Storming", "Norming" and "Performing." The Norming stage is associated with a drop in performance as team members are unfamiliar with each other and still understanding how they should collaborate. In the Storming stage, team members continue to understand each other and begin to establish roles and strategies. In the Norming stage, the team performance begins to improve as agents learn to collaborate harmoniously. Finally, in the Performing stage, the team is achieving its full potential, exhibiting the highest level of cooperation and score. We provide a depiction of these stages as part of Figure 9.2. We note that all teams may not follow these stages linearly, as literature has suggested stages can be skipped or experience a back-and-forth.

Markov Game – We formulate our two-player simultaneous-move environment as a Markov Game [157]. A Markov game for 2 agents is defined by a set of global states, $S_1, S_2 \in S$, a set of actions for each agent, $A_1, A_2 \in A$, and the transition function, $T : S \times A_1 \times A_2 \mapsto S$. In each time step, agent i chooses action, $a_i \in A_i$, obtains reward as a function of state, S, and its action $r_i : S \times A_i \mapsto \mathbb{R}$. Agent i aims to maximize its discounted reward $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$, where $\gamma \in [0, 1]$ is a discounted factor. For training teammates, we utilize agent-agent collaborative training, which trains two separate agents jointly via single-agent Proximal Policy Optimization

(PPO) [119] [1].

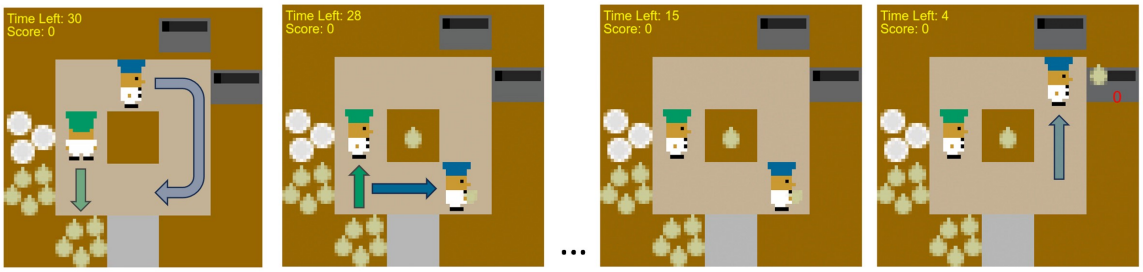
9.3 Teaming with Real Humans

In this section, we present two examples to display a gap in the quality of AIs in HMT. Specifically, we look at two recent approaches to produce collaborative AI agents [74, 73]. We argue and display that the AIs trained via these approaches are rigid and exhibit individualized behaviors, missing out on collaborative teaming strategies that can ultimately result in higher team scores.

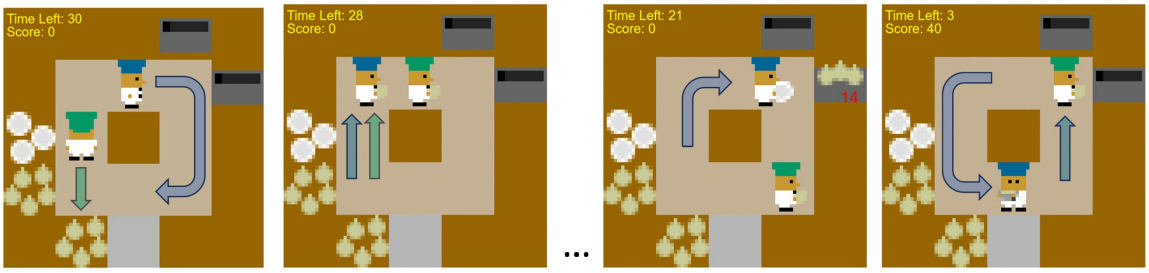
We require collaborative AI agents that can effectively reach a consensus with humans on a teaming strategy that ultimately results in high performance. In cases where the human has a preferred strategy, the AI teammate should be able to support said strategy. Furthermore, across repeated gameplay, team performance within an HMT should improve until the HMT reaches maximum performance associated with a specific human partner's capability. As our domains are relatively low-dimensional, we expect that with a collaborative AI maintaining these attributes, HMT performance should reach maximal values.

Within many Overcooked-AI scenarios, it is possible to design heuristic high-performance collaboration strategies. For our first example, in Figure 9.1, we display the *Coordination Ring* scenario. A simple collaboration strategy in this domain is to utilize the counter to continuously pass objects, minimizing agent movement through efficient handoffs (displayed in Figure 9.1a). To test this collaboration strategy, we utilize agents publicly available from [73] In Figure 9.1, we display a frame-by-frame of the human-preferred coordination strategy (top column) and AI-preferred coordination strategy (bottom column), which was a strategy where agents individually collect ingredients and place them in pots, moving clockwise. This behavior was inferred through trial and error. With the human-preferred

[1] We utilize PantheonRL [281], a library built upon StableBaselines3 for training our agents,



(a) Here, we display the **human-preferred collaboration behavior** that focuses on minimizing agent movement and efficient handoffs using the middle counter. We see that the human (green agent) picks up an ingredient and places it on the counter at the start of the game. The AI agent (blue) is unsure how to handle this strategy and freezes for approximately 80% of the remaining episode before finally placing an onion in the pot. This is a display of unsuccessful collaboration that receives a total score of 0.



(b) Here, we display a human adapting to an **AI-preferred suboptimal teaming strategy**, where agents act individually. We see that agents are able to successfully retrieve ingredients, and create and serve soups. This individualized coordination results in some success, achieving a score of 40.

Figure 9.1: Case Study in Human-Machine Teaming with Gameplay Images with Different Teaming Strategies. It is clear that the models produced are not robust to multiple strategies of play and can result in agents performing nonsensical behavior (stuck in place).

strategy, the AI agent freezes for the majority of the game, creating an extremely frustrating and low-performing AI teammate. With the AI-preferred strategy, the human is able to successfully team with the AI, but the strategy is not optimal or what the human prefers. We provide videos and further details in the supplementary.

In a second example, we utilize the *Optional Collaboration* domain, displayed in Figure 9.4b, which is also utilized in our human-subjects experiment. This domain was designed to incentivize collaboration, where mixing ingredients will result in a

incorporating our novel architecture into the codebase.

higher score per dish. Here, we program two intelligent heuristics: In the first, each agent acts completely individually, cooking single-ingredient dishes and serving. In the second, agents share ingredients (which costs additional timesteps) but are able to successfully cook mixed ingredient dishes. We find that the collaboration strategy achieves a 408 team score, approximately 30% more score compared to the individual strategy of 306. Later in the paper, we find that trained policies under [74] exhibit similar team score to that of the individual agent and through observation, find that these agents exhibit mostly independent behaviors. Further, real human end-users collaborating with these agents are unable to far surpass the individual strategy score.

Thus, in the rest of the paper, we look to explore xAI techniques as a mechanism for closing this gap and allowing agents within a human-machine team to facilitate collaborative strategies that outperform the individualized behaviors agents assume.

9.4 Methodology

In this section, we first present our architecture for training interpretable AI teammates in Overcooked-AI. We then present a contextual pruning algorithm, noting parallels to the broader "lottery ticket hypothesis" [282] for neural network models. We display a depiction of our training procedure as part of Figure 9.2.

9.4.1 Interpretable Discrete Control Trees

We create an interpretable machine learning architecture, Interpretable Discrete Control Trees (IDCTs), that can be used directly with RL to produce interpretable teammate policies. Below, we briefly detail our architecture, as well as advancements to enhance ease-of-training and interpretability.

Architecture

Our IDCTs are based off of differentiable decision trees (DDTs) [67, 131] – a neural network architecture that takes the topology of a decision tree (DT). DDTs contain decision nodes and leaf nodes; however, each decision node within the DDT utilizes a sigmoid activation function (i.e., a "soft" decision) instead of a Boolean decision (i.e., a "hard" decision). Each decision node, i, is represented by a sigmoid function, displayed as $y_i = \frac{1}{1+\exp(-\alpha(\vec{w}_i^T \vec{x} - b_i))}$. As this representation is difficult to interpret, [10] presented differentiable crispification, consisting of two components: 1) Decision node crispification, which recasts each decision node to split upon a single dimension of our input feature, and 2) Decision outcome crispification, which translates the outcome of a decision node so that the outcome is a Boolean decision rather than a set of probabilities. Both operations utilize the straight-through trick [68] to maintain gradients, allowing for both an interpretable forward propagation through the model that traces down a single branch of a tree as well as gradient flow to update parameters of the neural tree model.

We initialize our IDCTs to be symmetric complete decision trees with N_l decision leaves and $N_l - 1$ decision nodes. Each decision leaf is represented by a sparse categorical probability distribution over actions. At each timestep, a state variable is propagated through each decision node, split on a single decision rule, with the output being a Boolean causing the decision to proceed via the left or right branch until arrival at a leaf node. At each leaf node, we sample from the respective probability distribution to produce a macro-action (e.g., in Overcooked-AI, "get an onion" or "place held ingredient on counter"). As we are training interpretable stochastic tree models that reason over macro-actions, we improve model predictability by applying an L1 norm loss over leaf node distributions to ensure sparsity. This penalizes high entropy action distributions at a leaf, increasing predictability and interpretability. While utilizing deterministic AI policies (IDCT with a single action

at each leaf node) may be easier to understand for users, we found these models could not converge to similar performance as our stochastic interpretable policies. We provide an ablation displaying this finding in the supplementary.

Contextual Pruning

As we focus on creating agents that cooperate with humans, we must limit the size of our interpretable tree-based models to a certain depth to promote user understanding. This follows prior work, finding trees of arbitrarily large depths can be difficult to understand [227] and simulate [134], and that a sufficiently sparse DT is desirable and considered interpretable [228]. However, this can make training difficult, as a small tree may not have the representational power to learn a high-performing policy.

[282] present the idea of a "lottery ticket hypothesis" in neural network training. This hypothesis suggests that initializing a large neural network with numerous sub-networks (referred to as lottery tickets) and conducting training can unveil a sub-network (a winning ticket) that, when isolated, retains the same performance as the original larger network. This finding supports the practicality of employing large models. Accordingly, a small tree with a limited number of sub-trees may fail to learn a good policy. Thus, the ability to effectively train IDCTs is at odds with maintaining user readability and simulatability.

There is much literature that effectively prunes neural networks [283], detecting sub-networks and increasing computational efficiency. Following this thread of literature, we design a *contextual pruning* algorithm that allows us to simplify large IDCT models post-hoc by accounting for the following:

1. **Boundaries of a variable's state distribution**: We utilize the minimum and maximum of each variable's range to parse impossible subspaces of a tree.

2. **Node hierarchy**: Ancestor nodes for a specific decision node may have al-

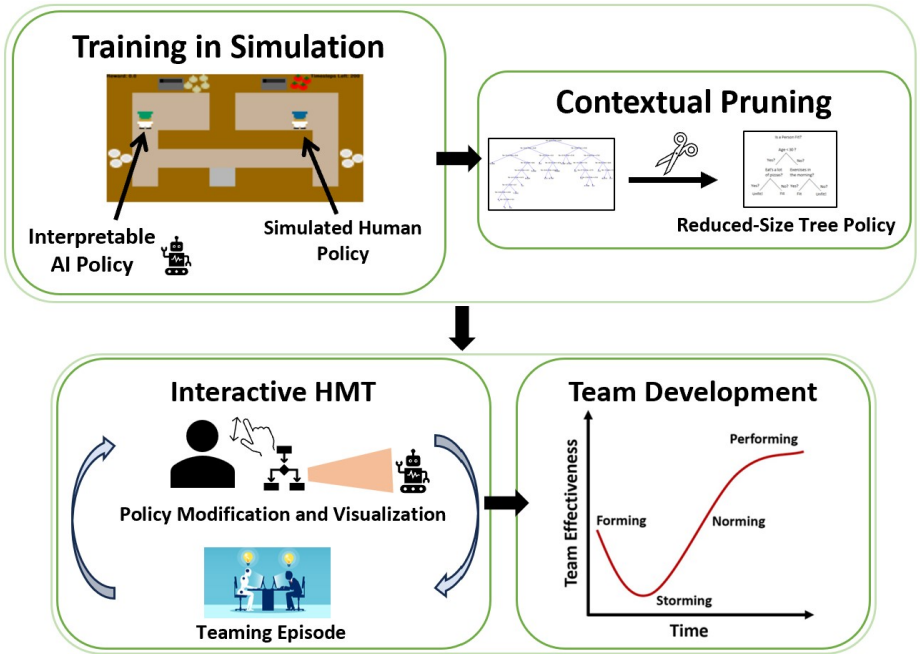


Figure 9.2: In this figure, we provide a high-level overview of the steps to produce a collaborative AI teammate with an interpretable policy representation and the proposed policy modification scheme evaluated in our user study.

ready captured a specific splitting criterion and, thus, may lead to redundancy.

By detecting redundancies, we can prune subspaces of the tree.

Formally, both operations can be accomplished by computing a corresponding hyperspace for each decision node, representing the input variable space leading to that specific node. Initially, all input variables reach the root node, defining a root node box, denoted by the Cartesian product $B = [-\infty, \infty] \times \cdots [-\infty, \infty]$, of cardinality d (where d is the dimensionality of the state space). However, since each state variable has a predefined range, B can be simplified by considering the upper and lower bounds of each variable's range. Additionally, by evaluating the splitting rule within a child node, we can further reduce B to represent an additional constraint. Applying this process to the entire tree through a complete tree traversal allows us to compute bounding boxes for each node. In cases where certain nodes do not yield a reduction in B, we can apply pruning to specific subtrees. We provide an algorithm for contextual pruning in the supplementary

material. [2] *This, in turn, allows us the benefit of training large tree-based models, greatly improving ease-of-training, while still being able to simplify the resultant model to an equivalent representation.* Empirically, we find that we can train tree sizes with over 200 leaves and often reduce the size 8-16x in tree depth.

9.4.2 Teammate Policy Modification

While the above architecture can be used alongside RL to produce a collaborative AI policy, the result may not actually be helpful or what the human wants. *Humans, when teaming with machines, should be able to intuitively update what the robot has learned or change it based upon preferences that may evolve over time.* Such is critical in the development of coordination strategies [44], and is associated with the calibration of trust, assignment of roles, and development of a shared mental model. As such, we propose a policy modification scheme that allows the user to repeatedly team with an AI and modify its AI's behavior, allowing for team development over repeated teaming episodes.

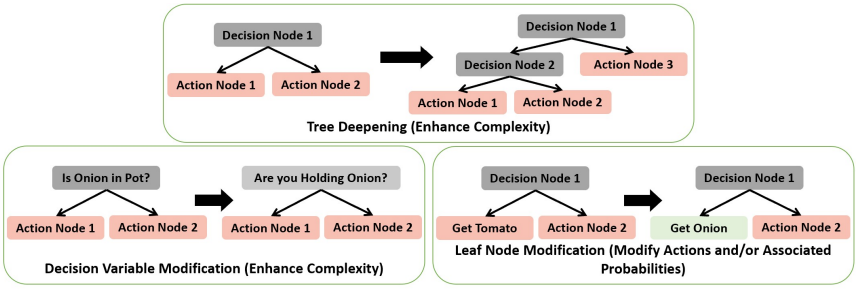


Figure 9.3: General Overview of the Human-Led Policy Modification GUI

We term our modification scheme *human-led policy modification*. Here, we provide humans with an explicit pathway to "communicate" with an AI after each teaming interaction through a GUI, displayed in Figure 9.3. Within this interface, users start with the pre-trained collaborative AI IDCT policy and can modify the

[2] In future, it would be interesting to instill such constraints directly into the learning process instead of applying post-hoc pruning.

AI's behavior by creating a new tree structure that may vary in what state features appear in the decision nodes, actions taken in leaf nodes, and the respective probabilities of actions within the leaf node. While cognitively challenging, such an interface provides users with the ability to modify teammate policies to their specifications.

9.5 Human-Subjects Study

In this section, we discuss our novel between-subjects user study that seeks to understand how users interact with an AI across repeated play under different factors. Below, we introduce our research questions, provide a description of the independent variables and procedure, include brief descriptions of the behaviors learned by collaborative AI, and finally discuss our findings.

Research Questions:

1. *RQ1:How does team coordination performance vary across different factors?*
2. *RQ2:How does team development vary across different factors?*

Independent Variables

We have two independent variables, **IV1**: the teaming method, and **IV2:** the domain. For **IV1**, we consider the following conditions (abbreviated by **IV1-C**), as follows:

1. **IV1-C1: Human-Led Policy Modification:** User teams with an AI maintaining an IDCT policy. After interacting with the agent (one teaming episode), the user can modify the policy via the modification GUI, allowing the user to update decision nodes and action nodes in the tree as well as tune affordances. Upon completion of user modification, the user can visualize the updated AI policy in its interpretable tree form prior to the next interaction.

2. **IV1-C2: AI-Led Policy Modification:** User teams with an AI maintaining an IDCT policy. After interacting with the agent, the AI utilizes recent gameplay to fine-tune a human gameplay model via Behavioral Cloning and performs RL for five minutes to optimize its own policy to better support the human teammate. Upon completion of policy optimization, the user can visualize the updated AI policy in its interpretable tree form prior to the next interaction. This is similar to Human-Aware PPO [73], adapted to an online setting.

3. **IV1-C3: Static Policy - Interpretability:** User teams with an AI maintaining an IDCT policy. After interacting with the agent, the user can visualize the AI's policy in its interpretable tree form prior to the next interaction. *Throughout this condition, the AI's policy is static.*

4. **IV1-C4: Static Policy - Black-Box:** User teams with an AI maintaining an IDCT policy. However, after interacting with the agent, the user does *not* see the AI's policy. *Here, the AI policy is the same as **IV1-C3**, but the human has lost access to direct insight into the model.*

5. **IV1-C5: Static Policy - Fictitious Co-Play:** [74]: User teams with an AI maintaining a static neural network (NN) policy trained across a diverse partner set. As this is a baseline, we utilize an NN rather than the IDCT policy model used in **IV1-C1-4**.

We display a brief table displaying the different characteristics across conditions in Table 9.1. Across the varying factors in **IV1** (in reverse order), there is an additional capability added to each condition. As we compare against two prior works [73, 74], it is important to note that when interpreting the results, readers should keep in mind that Human-Led Policy Modification has an additional capability beyond model training (i.e., users have the ability to change the policy). As such,

readers should consider 1) training performance, 2) HMT performance, and 3) HMT development in evaluating the pros and cons across conditions.

Table 9.1: An overview of the characteristics across different **IV1** factors.

Approaches	Explicit Interaction	Policy Changes Across Iterations	White-Box	Base Policy
IV1-C1	✓	✓	✓	IDCT
IV1-C2	✗	✓	✓	IDCT
IV1-C3	✗	✗	✓	IDCT
IV1-C4	✗	✗	✗	IDCT
IV1-C5	✗	✗	✗	NN

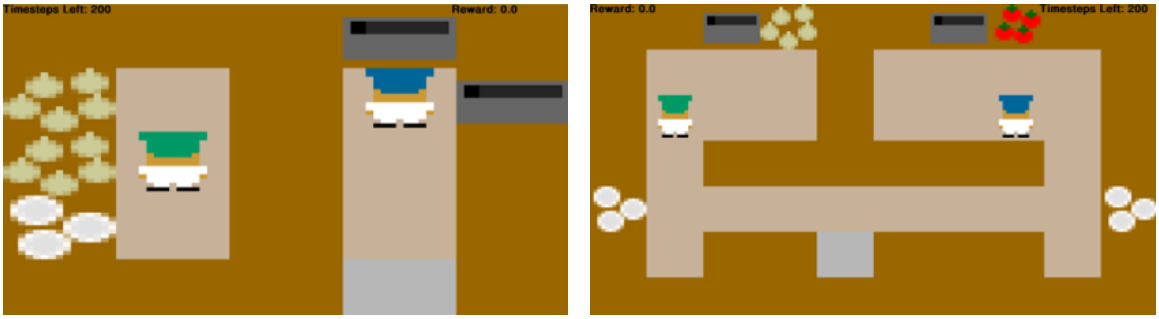
For **IV2**, we consider the following domains (abbreviated by **D**), displayed in Figure 9.4:

1. **IV2-D1: Forced Coordination:** Users team with an AI that is separated by a barrier and must pass over onions and plates in a timely manner. In this domain, agents are forced to collaborate.

2. **IV2-D2: Two-Rooms Narrow:** In this domain, the team can operate individually or collaboratively, and members are not forced to coordinate. This domain has increased complexity, both with respect to the size of the domain and the types of soups that can be cooked. Furthermore, in this domain, collaboration is incentivized through a higher reward for mixed-ingredient dishes (combining onions and tomatoes) over single-ingredient dishes.

Each domain was chosen so that collaborating with the teammate would result in a higher score than working individually. For each domain, we train an IDCT policy via agent-agent collaborative training and a NN policy following the training scheme in [74]. As the total score is a combination of the behavior across both agents within the human-machine team, it is important to note the collaborative policies produced via training. In **IV2-D1**, the IDCT policy (utilized in **C1-4**) converged to

a policy with an average reward of 315.22 ± 14.59, and the neural network policy converged to an average reward of 403.16 ± 16.08 over 50 teaming simulations with the synthetic human teammate the policy was trained with. In **IV2-D2**, the IDCT policy converged to a policy with an average reward of 171.46 ± 18.89, and the neural network policy converged to an average reward of 295.02 ± 1.86. Thus, a consequent confound due to the current difference in performance capabilities between interpretable vs. black-box models is that the NN policy outperforms the IDCT policy in both domains. This displays a gap in We can also compare to the heuristic policies presented previously, observing that both IDCT and NN policies in **IV2-D2** underperform high-performance collaboration between the two agents. We provide depictions of the interpretable policies in the supplementary material.



(a) Forced Collaboration (b) Optional Collaboration

Figure 9.4: This figure depicts each domain that we will be using in our experiment.

Procedure

A participant is first randomly placed into one of the five conditions in **IV1**. The participant starts with a pre-experiment survey collecting demographic information, experience with video games and decision trees, and the Big Five Personality Questionnaire [266]. Afterward, a participant conducts a brief tutorial in Overcooked with a random AI agent, improving the user's understanding of game controls and the assigned task. Once completed, the primary experimentation begins. Users

will team with an AI four times in each domain (randomly ordered), and are told that their goal is to maximize their score in the last teaming interaction, the "performance round". After each teaming interaction, in the first three factors, the user will modify the AI's policy (**IV1-C1**), the AI will optimize its own policy (**IV1-C2**), or the user will view the policy (**IV1-C3**). In **IV1-C4** and **IV1-C5**, as the AI is black-box, transitionary pages are shown to the participant, providing them a pause before they team with the agent again. Upon completion of the condition-specific (or lack of) actions, users complete a NASA-TLX Workload Survey [270]. After users have completed a domain, providing us with four episodes of teaming data and workload assessments, we administer several post-study scales to support our quantitative findings, including the Human-Robot Collaborative Fluency Assessment [267], Inclusion of Other in the Self scale [268], and Godspeed Questionnaire [269] as well as essay-like questions asking users to describe their experience with the AI. Each respective scale has been verified for validity in prior work and is used to assess the quality of the HMT interaction. Upon completion of the two domains, the experiment concludes. We provide additional details regarding our procedure in the supplementary material. By comparing different conditions, we can gain valuable insights into the advantages of AI interpretability and interaction.

9.5.1 Results

We recruited 50 participants under an IRB-approved protocol, whose ages range from 18 to 32 (Mean age: 24.14; Standard deviation: 4.10; 46% Female, 2% Non-Binary), with participants randomly assigned to each of the factor levels, with ten total subjects per level. Our data was modeled as a full-factorial, between-subjects ANOVA. We test for normality and homoschedascity in our data and employed a corresponding non-parametric test if the data failed to meet these assumptions. We provide complete details regarding our analysis within the supplementary. We

display our objective findings in Figure 9.5.

RQ1: Team Coordination Performance

In analyzing team reward, we find interesting trends with respect to the maximum reward participants obtained within a domain across all teaming iterations (Figure 9.5b) and reward participants obtained within a domain within the performance round (Figure 9.5d). Utilizing a Friedman's test, we find that there is a significant difference across domains ($\chi^2(1) = 46.08, p < 0.001$). Accordingly, we analyze the two domains separately.[3]

In **IV2-D1**, a Kruskal-Wallis Test was conducted to analyze differences in maximum performance obtained across teaming paradigms, finding a significant effect ($\chi^2(4) = 20.146, p < 0.001$). We conduct post-hoc pairwise comparisons, utilizing Dunn's test with the Bejamini-Hochberg adjustment, and find that **IV1-C5** is significantly better than **IV1-C1** ($p < 0.001$), **IV1-C2** ($p < 0.01$), **IV1-C3** ($p < 0.01$), and **IV1-C4** ($p < 0.05$). While **IV1-C5** should outperform the tree-based models as it converged to a higher-performance teaming policy, it is interesting that **IV1-C1** has several participants that outperform the maximum performance of **IV1-C5**.

In **IV2-D2**, trends while observing maximum reward and rewards during the performance round are similar. Thus, we present the abbreviated results here for the maximum reward attained by participants across iterations. A Kruskal-Wallis Test was conducted to analyze differences in participant teaming performance across conditions, finding a significant effect ($\chi^2(4) = 29.922, p < 0.001$). We conduct post-hoc pairwise comparisons, utilizing Dunn's test with the Bejamini-Hochberg adjustment, and find that **IV1-C5** is significantly better than **IV1-C2** ($p < 0.001$), **IV1-C3** ($p < 0.001$), and **IV1-C4** ($p < 0.001$), and **IV1-C1** is significantly better than **IV1-C2** ($p < 0.05$), **IV1-C3** ($p < 0.05$), and **IV1-C4** ($p < 0.05$).

[3]We provide the complete test statistics in the supplementary material.

These findings display that black-box models can outperform white-box approaches in HMT. This can be attributed to black-box models being easier to train, and decision trees being difficult to interpret for some users. Furthermore, we find that white-box approaches supported with policy modification can outperform white-box approaches alone in the second domain. Third, by comparing **IV1-C3** to **IV1-C4**, we see that interpretability did not provide any direct benefits in HMT performance. Lastly, in **IV2-D2**, we see that users do not attain team scores near that of a collaborative heuristic strategy, displaying a gap in HMT performance.

RQ2: Team Development

In analyzing team development, we look at the change in reward across iterations and relate our findings to Tuckman's model [44]. Specifically, we analyze the trends across iterations (did agents improve from iteration one to four) and identify characteristics of users that performed well in team development. Utilizing a Friedman's test, we find that there is a significant difference across domains ($\chi^2(1)=20.48$, $p < 0.001$). Accordingly, we analyze the two domains separately.

We first conduct an analysis to see which conditions facilitate team development. In **IV2-D1**, we see only **IV1-C5** ($p < 0.05$) improves significantly over repeated iterations. In **IV2-D2**, we see the only **IV1-C1** ($p < 0.01$) and **IV1-C2** ($p < 0.01$) significantly improve over repeated teaming interactions. The improving interactions can be related to the Norming stage in team development, where teams begin to establish a coordination strategy and shared mental models. In **IV2-D2**, we see that both conditions that facilitate the Norming stage of teaming had the attribute of policy adaptation (see Table 9.1).

Next, we analyze whether different factors allow HMT to improve more quickly than others. In **IV2-D1**, we find that there is no significant difference across conditions in how much participants improve from the first teaming iteration. However,

we do find that conscientiousness is trending in its correlation with faster improvement ($0.05 < p < 0.1$). In **IV2-D2**, similarly, we find that there is no significant difference across conditions in how much participants improve from the first teaming iteration. However, we do find that there is a significant effect in how much participants improve and their familiarity with Decision Trees ($F(1) = 7.448, p < 0.01$).

Finally, as we detect an interesting trend in **IV2-D1** under the **IV1-C1** condition, we analyze this data further. We see a drop in performance between the first teaming iteration and later iterations, followed by a rising trend. We believe this relates to the Forming and Storming stages in team development, where team members are still developing effective strategies to coordinate. As the last iteration begins to see an improvement in performance, we hypothesize that the team development process was beginning to shift into the Norming stage. In future work, it would be interesting to evaluate a larger number of iterations to see if the behavior in this domain would continue to trend upward.

Subjective Findings

In **IV2-D1**, we find that users did not find any subjective differences toward the teaming interaction across conditions. In **IV2-D2**, we find that users find collaboration with AIs under condition **IV1-C2** and **IV1-C4**, on average as less fluent than **IV1-C1** ($p < 0.01, p < 0.01$), and **IV1-C3** as less fluent than **IV1-C5** ($p < 0.05$). Users trusted the AI and perceived the AI contributed more in **IV1-C5** than in **IV1-C2** ($p < 0.05, p < 0.05$) and **IV1-C4** ($p < 0.05, p < 0.05$). Furthermore, the users viewed the AI more positively in **IV1-C1** and **IV1-C5** than in both **IV1-C2** ($p < 0.05, p < 0.05$) and **IV1-C4** ($p < 0.05, p < 0.01$). Also, the users viewed the AI more positively in **IV1-C3** than **IV1-C4** ($p < 0.05$). From the last finding, we can see that given the same policy, users view an AI with an interpretable policy more positively. Furthermore, we see approaches that perform better (i.e., **IV1-C1** and

IV1-C5) are often rated with higher subjective ratings.

Qualitative Findings

End-users provided qualitative feedback by describing 1) the AI with characteristics and 2) their teaming strategy with the AI. While users were not asked to provide commentary during the experiment, experimenters noted down general frustration and quotes. One specific quote said by a participant was "I can't get it [the AI] to do what I want it to do, so I'll do what it wants me to do," accurately describing the rigidity of HMT. In Table 9.2, we display a sentiment analysis conducted for words used to describe AI teammates within each domain. We see a positive correlation between sentiment and HMT performance. A wide variety of characteristics were used to describe the AI. Notably, under white-box conditions, the most common words to describe the AI were "helpful" and "predictable" in **IV-D1**, and "unintelligent" in **IV-D2**. Under black-box conditions, descriptors widely differed with "slow" and "unhelpful" being the most common word for **IV1-C3** and "intelligent" and "individual" for **IV1-C5**.

In looking at descriptions of teaming strategies, in **IV1-C1**, many users noted how they attempted to modify the AI, what they learned across repeated interactions, and whether they were successful in improving the AI collaborator. With other conditions, participants noted how they could act preemptively to best support the AI's strategy. Furthermore, in black-box conditions, several participants mentioned a trial-and-error process in understanding the AI teammate.

9.6 Call-to-Action

Achieving successful collaboration between humans and machines is incredibly challenging. Toward this goal, we specify two directions that must be addressed to achieve fluent HMT.

		IV1-C1	IV1-C2	IV1-C3	IV1-C4	IV1-C5
Forced Coordination	Pos	0.25	0.49	0.56	0.22	0.49
	Neu	0.34	0.42	0.39	0.61	0.36
	Neg	0.41	0.09	0.06	0.17	0.15
Optional Coordination	Pos	0.11	0.07	0.24	0.06	0.52
	Neu	0.39	0.24	0.36	0.46	0.29
	Neg	0.49	0.69	0.39	0.48	0.19

Table 9.2: Sentiment Analysis over User-Specified AI Characteristics, presenting positive, neutral, and negative sentiment. We see a positive correlation between sentiment and performance.

1. Researchers must focus on the development of white-box approaches that can achieve competitive initial performance to that of black-box models and learning techniques to support the generation of collaborative AI behaviors rather than individual coordination.

2. The creation of mixed-initiative interfaces that facilitate users, who may vary widely in ability and experience, to improve team collaboration across and within interactions.

Furthermore, developing AI teammates that must function over longer-term interactions may require pivoting from episodic measures of teaming, such as minimizing workload or maximizing performance, to longer-term measures that may provide overarching benefits, such as team development.

9.7 Conclusion

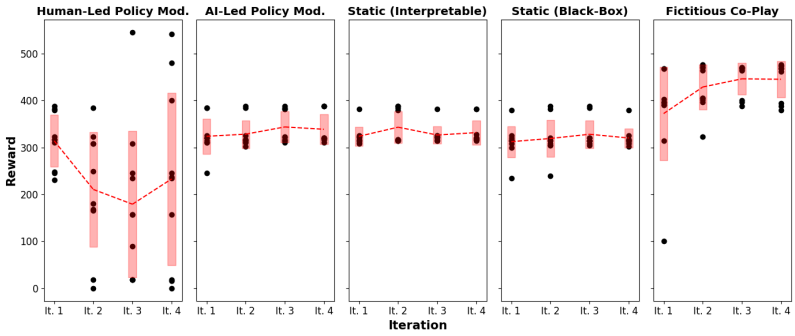
This work delves into the realm of repeated interactions with machine learning models within a sequential decision-making HMT paradigm. Importantly, we present a key gap in HMT, displaying that current methods do not facilitate human-machine collaboration to the fullest. We deploy a possible solution, human-led policy modification, providing the human with the ability to go under-the-hood" of his/her AI teammate and iteratively reprogram behavior. We find that human-led

policy modification allows for a human-machine team to achieve higher performance than white-box models without this capability. However, as interpretable models are more difficult to generate, Fictitious Co-Play is able to better support high-performance HMT. Given these mixed findings, in future, researchers must focus on developing better interpretableML approaches to support the generation of white-box teammates, study the modality of communication between agents and humans, and explore mechanisms to allow HMT to scale beyond individual coordination and toward effective collaboration.

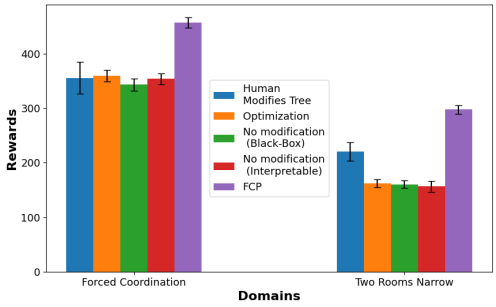
Future Work: In future, it would be interesting to conduct the experiment to a higher number of iterations, or until the team converges to a set of coordination strategies (the "performing" stage) and distribution of roles. Further, the possibility of adding in feedback from the AI regarding human-led policy modification (checking for logic inconsistencies, etc.) may be used to facilitate speedier team development. It would also be interesting to utilize different paradigms in communicating with the human as language may be an easier medium than a decision tree interface. An interesting additional baseline that would be beneficial in further assessing the quality of HMT (both quantitatively and qualitatively) is comparing human-machine collaboration to human-human collaboration. Lastly, future work should be done to optimize the accessibility of GUIs for policy modification via xAI techniques.

Limitations: This study was conducted at a university campus during a summer semester. While the population was diverse in age, gender, and university major, most students were based in engineering, presenting a population bias. This study is also cross-sectional and thus, it is difficult to determine the cause of different stages of team development. Finally, we evaluate the paradigm of human-led policy modification in a collaborative game, where good collaboration outperforms individualized coordination, but poor collaboration underperforms individualized

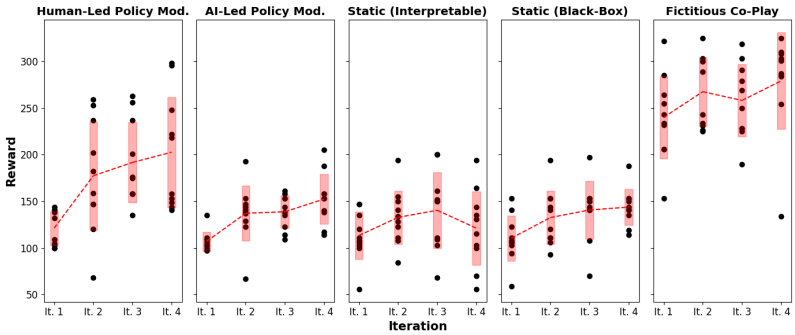
coordination. The findings may not generalize to other types of games or different reward schemas. In the future, further testing over different layouts and reward schemes alongside a deeper analysis of agent roles would provide insights into the benefits and effects of interactive programming for neural tree models.



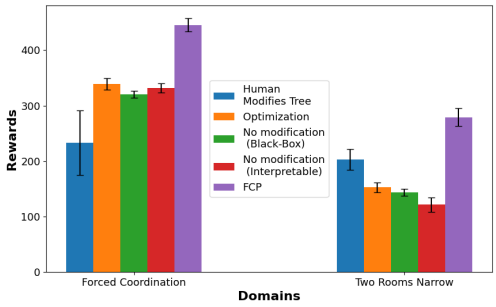
(a) Performance Data Across Iterations in **IV2-D1**: Forced Coordination



(b) Aggregate **Maximum Rewards** Across Each Condition



(c) Performance Data Across Iterations in **IV2-D2**: Optional Collaboration



(d) Aggregate **Performance Round Rewards** Across Each Condition

Figure 9.5: This figure displays gameplay scores from participants over different iterations (Left) and aggregate findings (Right).

CHAPTER 10
LIMITATIONS AND FUTURE WORK

While much of my work has been successful in creating architectures and algorithms to support human-robot collaboration, this field is relatively new, and there is much that can be explored as well as limitations that can be addressed. Below, I discuss limitations to my work as well as some possible avenues for future work.

10.1 Limitations

Each of the different works presented throughout this thesis has limitations. Below, we present several limitations, grouped by high-level descriptions or techniques.

10.1.1 Multi-Agent Reinforcement Learning

In chapter 4 and chapter 5, we utilize MARL to infer coordination and communication policies for decentralized agents under partial observability. These techniques, even for small-scale domains that have been discretized to 20x20 grids, require large amounts of compute and can take several days to run on high-performance computing machines. While there has been success in the past with training MARL with simulated agents and substituting a human user online [284, 74], we have not tested our coordination-communication protocols with humans. The key limitation that stops these frameworks from readily being deployed with humans is the use of black-box agent-to-agent communication. For a human to participate as a member of the team and communicate with other agents, black-box communication messages must be first related to semantic information prior to communication. Further, if the human would like to communicate back to agents, a similar mapping would have to be created, translating semantic information to black-box messages

that an agent can then receive.

10.1.2 Interpretability of Tree-Based Models

Interpretability is a subjective concept that is person-specific, and the utility of interpretability can be functionally grounded within a certain domain or for a certain task. In our work, we consider interpretable policies that can represent a robot's decision-making policy or a robot's representation of a human's behavior. For a robot's decision-making policy, it is unclear whether continuous control outputs (e.g., predicting a steering angle) are useful for end-users, as this is an extremely short-term prediction presented at a low-level. For engineers, such a tree may provide more benefit, allowing them to verify models for safety and perform debugging. However, this has not been tested and is left to future work. Even in higher-level spaces (such as reasoning over macro-actions), online, we see that trees may not be the correct modality for communicating during human-robot collaboration. These complex models require time to understand, presenting a large cost that can reduce collaboration performance. Further, as robots may maintain a second model that represents a teammate's decision-making (and other components), we found that presenting multiple trees results in information overload and a decrease in performance. Here, it may be important to consider "when to communicate," reducing communication when a robot behaves as expected, or the human is busy. Finally, large trees may be required in complex domains, limiting the simulatability of such a model. In these cases, we still may be able to find utility with white-box methods by interpreting the tree in sub-spaces or creating programs to post-process the tree into a closed-form natural language description.

10.1.3 Evaluating the Utility of Our Systems

In many of our tasks, we evaluate collaboration in simulated domains with participants that mostly consist of university students. Simulated agents have been shown to be perceived differently than embodied agents [285]. As such, in future studies, we would like to deploy our models on actual robot systems with users. Further, as our human-subjects studies have mainly consisted of university students, our studies maintain a population bias. In the future, it would be beneficial to deploy our studies with a broader set of end-users, more closely associated with our target population. Further, as we utilize explanations in sequential decision-making settings or across repeated games, it is important to characterize the short-term and long-term benefits of xAI-based support. Evaluating across a larger number of repeated iterations may help to ensure that we effectively capture long-term benefits of xAI (i.e., by allowing for sufficient time for the development of a shared mental model and the creation of a collaboration plan). Evaluating at too short a scale may result in incorrectly finding that certain xAI is not beneficial.

10.2 Future Work

Below, I present several directions of future work that would provide valuable advancements to the field of human-robot collaboration.

10.2.1 Communicating with Humans

Throughout my thesis, I have explored interpretable tree-based models as the primary form of communication with humans. While trees can be understood by users, they can prove cognitively intense, especially at larger scales. Exploring methodologies to allow users to digest tree information more easily may provide a path for users to more quickly develop a shared mental model. Furthermore, as

tree-based models have properties that allow for fast computation and verification, users may also be able to query the tree to better understand a model. Questions such as "what behaviors should I expect under the following states?" can be quickly answered by such models and may facilitate a speedier development of a shared mental model. It is important to continue to develop techniques that provide agents with mechanisms to communicate objectives and important information to humans, and develop human-machine interfaces, evaluated via user studies, to allow humans to communicate with robots at scale. Finally, it would be beneficial to utilize context-based communication, similar to that used in chapter 4 and chapter 5. Here, the modality of communication would vary based upon state variables. An example of a case where context-based communication would be beneficial is modulating the modality and frequency of communication while considering the human teammate's current workload, allowing for reduced and/or abstracted communication during stress and increased and/or transparent communication during idling.

10.2.2 Interacting with Interpretable Models

We have seen some instances where users can improve interpretable teammate policies via an interface. This capability is promising, enabling end-users to "go under-the-hood" of machine learning models and tune affordances or interactively and iteratively reprogram behavior. However, humans may incorrectly modify the model, leading to unsafe or unwanted behavior. It would be beneficial to combine the AI's abilities in facilitating a sense of supervision over the tree policy while maintaining the user's preferences. Further, in future, it would be beneficial to explore other use-cases for such an interface, such as allowing engineers to quickly detect and remove unwanted model characteristics. For example, such a technique may allow for removing unsafe behavior or unintended model biases, ensuring

that only positive-impact models are being deployed in our world.

CHAPTER 11
CONCLUSION

I envision a future with seamless collaboration between humans and robots. Human-robot collaboration has the potential to increase productivity, enhance safety, and improve the quality of our lives. My aim is to develop new computational methods to support robots dynamically interacting with a wide variety of people in dynamic and novel contexts, ultimately moving towards facilitating human-robot collaboration in applications spanning from healthcare and manufacturing to household assistance.

Today, most industries utilizing robots keep them in caged setups or human-free environments to avoid accidents. Current interactions between humans and robots are incredibly limited, only allowing the robot to perform rigid, predefined behaviors. In this thesis, I have contributed several techniques and findings to push the frontier of real-world robotics systems toward those that understand human behavior, maintain interpretability, develop over time, and coordinate with high performance. I provide a summary of each contribution as follows:

11.1 The Importance of Communication in Multi-Agent Systems

In this work, we create a multi-agent coordination framework augmented with communication to address coordinating multiple decentralized agents under partial observability. We emulate human-human teams, where personnel judiciously choose when to communicate and whom to communicate with, communicating only when beneficial, and propose a novel multi-agent reinforcement learning algorithm, Multi-Agent Graph-attentIon Communication (MAGIC), with a graph-attention communication protocol in which we learn 1) a Scheduler to help with

the problems of when to communicate and whom to address messages to, and 2) a Message Processor using Graph Attention Networks (GATs) with dynamic graphs to deal with communication signals. We evaluate our method and baselines in several environments. In our most challenging domain, an adversarial soccer domain, we achieve a near-perfect success rate in scoring, while most baselines struggle to reach 70%. Not only does MAGIC produce state-of-the-art results, MAGIC is able to converge 52% faster than the next-quickest baseline, and communicates 27.4% more efficiently than the average baseline. Further, we find that targeted communication results in better multi-agent coordination performance than communicating all the time.

11.2 Modeling Heterogeneity in Multi-Agent Systems

In this work, we attempt to emulate the communication strategy of high-performing human-human teams within a multi-agent coordination framework, where members implicitly understand the different roles of other heterogeneous team members and adapt their communication protocols accordingly. As such, we formulate a MARL framework that supports heterogeneous teams with different state, action, and observation spaces and allows for stylized communication between these different classes of agents. Specifically, we propose Heterogeneous Policy Networks (HetNet) to learn efficient and diverse communication models for coordinating cooperative heterogeneous teams. Here, we cast the cooperative MARL problem into a heterogeneous graph structure, and propose a novel heterogeneous graph-attention network capable of learning diverse communication strategies based on agent classes. Our empirical evaluation shows that HetNet sets a new state of the art in learning coordination and communication strategies for heterogeneous multi-agent teams by achieving a 434.7% performance improvement over the next-best baseline in a heterogeneous firefighting domain that requires sensing agents

(e.g., UAVs) to coordinate with action agents with poor sensors (e.g., fast-moving fixed-wing aircraft). Importantly, in comparing HetNet to homogeneous MARL baselines, we see modeling heterogeneity explicitly within a multi-agent system can lead to performance benefits.

11.3 Inferring Personalized Behavioral Policies of Heterogeneous Human Decision-Makers

The ubiquity of robotics will depend upon robots being able to team with and understand a diverse set of users. Accounting for personalization is essential in human-machine teaming as a robot inferring incorrect teammate behavior (i.e., by assuming all users fit to the mean or mode) could lead to poor teaming performance. In this section, I create a data-efficient technique to create personalized models of user behavior directly from a dataset of unlabeled heterogeneous users, enabling robots to gain a personalized, implicit understanding of their human teammate's decision-making behavior via an inferred, *person-specific embedding*, non-parametric in the number of demonstrator types. We find our approach, Personalized Neural Trees (PNTs), achieve state-of-the-art performance in predicting user behavior across two synthetic domains and one real-world dataset of users providing routing preferences. Furthermore, in a user study, we find PNT is more interpretable, easier to follow, and quicker to validate than neural network models. With this new architecture, machines can better learn from heterogeneous users and detect person-specific behaviors, allowing for greater personalization in robotic counterparts.

11.4 Generating Interpretable Robot Policies

Interpretable policies are extremely important in the creation of systems that are ready for real-world deployment and interaction with humans. In this section, I

propose Interpretable Continuous Control Trees (ICCTs), a tree-based framework for continuous control that allows for direct synthesization of robot behavior via gradient-based techniques within an interpretable representation. We evaluate our model in its ability to learn high-performance control behaviors across several continuous control problems, including four autonomous driving (AD) scenarios. We find extremely positive support for our ICCTs, displaying the ability to learn interpretable policy representations that parity or outperform baselines by up to 33% in autonomous driving scenarios while achieving a 300x-600x reduction in the number of policy parameters against deep learning baselines. Importantly, this work provides a strong step toward solutions for two grand challenges in interpretableML announced in 2021: (1) Optimizing sparse logical models such as decision trees and (2) Interpretable Reinforcement Learning.

11.5 The Utility of Explainable AI in Human-Robot Collaboration

Recent advances in machine learning have led to growing interest in xAI to enable humans to gain insight into the decision-making of machine learning models. In this work, we deploy xAI techniques with humans while they are teaming with an AI, presenting a novel analysis of xAI under sequential decision-making settings for human-machine teams. Significantly, I utilized Minecraft as a human-machine teaming platform, creating a complex collaboration task where humans and machines must make decisions in real-time and reason about the world in a continuous space. The resultant interaction is much closer to the envisage of a real-world human-robot interaction scenario compared to prior research that has focused on point interactions and does not assess the preoccupation cost of online explanations. Importantly, I found that 1) using interpretable models that can support information sharing with humans can lead to increased situational awareness and 2) xAI-based support is not always beneficial, as there is a cost of paying attention

to the xAI and this may outweigh the benefits obtained from generating an accurate shared mental model. These findings emphasize the importance of developing the "right" xAI models for human-machine teaming and the optimization methods to support learning these xAI models.

11.6 Team Development in Human-Robot Collaboration

The current state of human-robot collaboration requires human teammates to adapt to machines as the programmed machine behaviors are rigid. In this work, we build towards adaptive, effective teaming by creating a pathway of bi-directional communication, utilizing interpretable policy representations as a mechanism to allow users to understand their machine teammates and allowing for explicit teammate policy modification through an interface (users can modify the machine's policy via a Graphical User Interface). In a large-scale user study provides several key findings. First, we find black-box architectures are easier to train and result in high HMT performance compared to white-box architectures. Second, we find that white-box approaches supported by interactive modification can lead to significant team development, outperforming white-box approaches alone. Together, these findings present a call-to-action to 1) improve approaches for generating collaborative agents with interpretable models and 2) generate effective mixed-initiative interfaces that allow users, who may vary in ability and experience, to interact and improve team behavior.

REFERENCES

[1] A. Gupta, A. Murali, D. P. Gandhi, and L. Pinto, "Robot learning in homes: Improving generalization and reducing dataset bias," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31, Curran Associates, Inc., 2018.

[2] D. Ferguson *et al.*, "An autonomous robotic system for mapping abandoned mines," in *Advances in Neural Information Processing Systems*, S. Thrun, L. Saul, and B. Schölkopf, Eds., vol. 16, MIT Press, 2004.

[3] L. Riek, "Healthcare robotics," *Communications of the ACM*, vol. 60, pp. 68–78, 2017.

[4] S. Nikolaidis, P. A. Lasota, G. F. Rossano, C. Martínez, T. Fuhlbrigge, and J. Shah, "Human-robot collaboration in manufacturing: Quantitative evaluation of predictable, convergent joint action," *IEEE ISR 2013*, pp. 1–6, 2013.

[5] I. Nourbakhsh, K. Sycara, M. Koes, M. Yong, M. Lewis, and S. Burion, "Human-robot teaming for search and rescue," *IEEE Pervasive Computing*, vol. 4, pp. 72–79, 2005.

[6] Y. Niu, R. Paleja, and M. Gombolay, "Multi-agent graph-attention communication and teaming," in *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '21, Virtual Event, United Kingdom: International Foundation for Autonomous Agents and Multiagent Systems, 2021, pp. 964–973, ISBN: 9781450383073.

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

[8] R. Paleja and M. Gombolay, "Heterogeneous learning from demonstration," in *Proceedings of the 14th ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI '19, Daegu, Republic of Korea: IEEE Press, 2019, pp. 730–732, ISBN: 9781538685556.

[9] R. Paleja, A. Silva, L. Chen, and M. Gombolay, "Interpretable and personalized apprenticeship scheduling: Learning interpretable scheduling policies from heterogeneous user demonstrations," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 6417–6428.

[10] R. R. Paleja, Y. Niu, A. Silva, C. Ritchie, S. Choi, and M. C. Gombolay, "Learning interpretable, high-performing policies for autonomous driving," *Robotics: Science and Systems XVIII*, 2022.

[11] R. Paleja, M. Ghuy, N. Ranawaka Arachchige, R. Jensen, and M. Gombolay, "The utility of explainable ai in ad hoc human-machine teaming," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34, Curran Associates, Inc., 2021, pp. 610–623.

[12] A. Staff, "10 years of amazon robotics: How robots help sort packages, move product, and improve safety," Jun. 2022.

[13] G. D. Cubber *et al.*, *Search and Rescue Robotics - From Theory to Practice*. London, GBR: InTechOpen, 2017, ISBN: 9535133756.

[14] L. M. Ma, T. Fong, M. J. Micire, Y. Kim, and K. M. Feigh, "Human-robot teaming: Concepts and components for design," in *Field and Service Robotics, Results of the 11th International Conference, FSR 2017, Zurich, Switzerland, 12-15 September 2017*, M. Hutter and R. Siegwart, Eds., ser. Springer Proceedings in Advanced Robotics, vol. 5, Springer, 2017, pp. 649–663.

[15] K. M. Lee *et al.*, "The effect of robot skill level and communication in rapid, proximate human-robot collaboration," in *Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction, HRI 2023, Stockholm, Sweden, March 13-16, 2023*, G. Castellano, L. D. Riek, M. Cakmak, and I. Leite, Eds., ACM, 2023, pp. 261–270.

[16] Q. Dai, D. Shen, J. Wang, S. Huang, and D. P. Filev, "Calibration of human driving behavior and preference using naturalistic traffic data," *CoRR*, vol. abs/2105.01820, 2021. arXiv: 2105.01820.

[17] L. Sanneman, C. K. Fourie, and J. A. Shah, "The state of industrial robotics: Emerging technologies, challenges, and key research directions," *Found. Trends Robotics*, vol. 8, no. 3, pp. 225–306, 2021.

[18] J. Krüger, T. K. Lien, and A. Verl, "Cooperation of human and machines in assembly lines," *CIRP annals*, vol. 58, no. 2, pp. 628–646, 2009.

[19] W. Knight, "Smart robots can now work right next to auto workers," *MIT Technology Review*, vol. 17, 2013.

[20] C. Liu and M. Tomizuka, "Algorithmic safety measures for intelligent industrial co-robots," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 3095–3102.

[21] *Diligent robotics collects $3m seed funding, launches autonomous robot assistants for hospitals*, Oct. 2019.

[22] O. Iroju, O. A. Ojerinde, and R. Ikono, "State of the art: A study of human-robot interaction in healthcare," 2017.

[23] R. E. Giachetti, V. Marcelli, J. Cifuentes, and J. A. Rojas, "An agent-based simulation model of human-robot team performance in military environments," *Systems Engineering*, vol. 16, no. 1, pp. 15–28, 2013.

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

[25] J. MacMillan, E. E. Entin, and D. Serfaty, "Communication overhead: The hidden cost of team cognition.," 2004.

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

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

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

[29] H. Taylor, "The effects of interpersonal communication style on task performance and well being," Ph.D. dissertation, University of Buckingham, 2007.

[30] A. Thomaz, G. Hoffman, and M. Çakmak, "Computational human-robot interaction," *Found. Trends Robotics*, vol. 4, no. 2-3, pp. 105–223, 2016.

[31] F. Gervits, T. W. Fong, and M. Scheutz, "Shared mental models to support distributed human-robot teaming in space," in *2018 aiaa space and astronautics forum and exposition*, 2018, p. 5340.

[32] S. Chernova and M. Veloso, "Confidence-based policy learning from demonstration using gaussian mixture models," in *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, ACM, 2007, p. 233.

[33] R. D. Dias *et al.*, "Using machine learning to predict perfusionists' critical decision-making during cardiac surgery," *Comput. methods Biomech. Biomed. Eng. Imaging Vis.*, vol. 10, no. 3, pp. 308–312, 2022.

[34] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017.

[35] B. Letham, C. Rudin, T. H. McCormick, D. Madigan, *et al.*, "Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model," *The Annals of Applied Statistics*, vol. 9, no. 3, pp. 1350–1371, 2015.

[36] U. Bhatt *et al.*, *Explainable machine learning in deployment*, 2019. arXiv: 1909.06342 [cs.LG].

[37] P. Voigt and A. Von dem Bussche, "The eu general data protection regulation (gdpr)," *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 2017.

[38] W.-Y. Loh, "Classification and regression trees," *Wiley interdisciplinary reviews: data mining and knowledge discovery*, vol. 1, no. 1, pp. 14–23, 2011.

[39] P. Barbiero, G. Ciravegna, D. Georgiev, and F. Giannini, "Pytorch, explain! a python library for logic explained networks," *arXiv preprint arXiv:2105.11697*, 2021.

[40] J. de Greeff *et al.*, "Workshop on longitudinal human-robot teaming," *Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, 2018.

[41] M. A. Goodrich, "Using narrative to enable longitudinal human-robot interactions," 2018.

[42] A. Logacjov, M. Kerzel, and S. Wermter, "Learning then, learning now, and every second in between: Lifelong learning with a simulated humanoid robot," *Frontiers in Neurorobotics*, vol. 15, 2021.

[43] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, "Building machines that learn and think like people," *CoRR*, vol. abs/1604.00289, 2016. arXiv: 1604.00289.

[44] B. W. Tuckman, "Developmental sequence in small groups.," *Psychological bulletin*, vol. 63, pp. 384–99, 1965.

[45] D. Adjodah *et al.*, "Leveraging communication topologies between learning agents in deep reinforcement learning," A. E. F. Seghrouchni, G. Sukthankar, B. An, and N. Yorke-Smith, Eds., pp. 1738–1740, 2020.

[46] M. Gombolay, R. Jensen, J. Stigile, S.-H. Son, and J. Shah, "Learning to tutor from expert demonstration via apprenticeship scheduling," in *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) Workshop on Human-Machine Collaborative Learning (HMCL)*, San Francisco, California, Feb. 2017.

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

[48] M. Gombolay *et al.*, "Human-machine collaborative optimization via apprenticeship scheduling," *J. Artif. Int. Res.*, vol. 63, no. 1, pp. 1–49, Sep. 2018.

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

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

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

[52] S. Sukhbaatar, A. Szlam, and R. Fergus, "Learning multiagent communication with backpropagation," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, Eds., 2016, pp. 2244–2252.

[53] K. Kurach *et al.*, "Google research football: A novel reinforcement learning environment," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, AAAI Press, 2020, pp. 4501–4510.

[54] Y. Liu, W. Wang, Y. Hu, J. Hao, X. Chen, and Y. Gao, "Multi-agent game abstraction via graph attention neural network," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative*

Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, AAAI Press, 2020, pp. 7211–7218.

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

[56] H. A. Simon *et al.*, *Models of a man: Essays in memory of Herbert A. Simon*. MIT Press, 2004.

[57] G. A. Klein, "A recognition-primed decision (rpd) model of rapid decision making," *Decision making in action: Models and methods*, vol. 5, no. 4, pp. 138–147, 1993.

[58] G. A. Korsah, "Exploring bounded optimal coordination for heterogeneous teams with cross-schedule dependencies," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, Jan. 2011.

[59] T. G. Dietterich, "Hierarchical reinforcement learning with the maxq value function decomposition," *J. Artif. Int. Res.*, vol. 13, no. 1, pp. 227–303, Nov. 2000.

[60] C. Sammut, S. Hurst, D. Kedzier, and D. Michie, "Learning to fly," in *Imitation in Animals and Artifacts*. Cambridge, MA, USA: MIT Press, 2002, pp. 171–189.

[61] S. Nikolaidis, R. Ramakrishnan, K. Gu, and J. Shah, "Efficient model learning from joint-action demonstrations for human-robot collaborative tasks," in *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI '15, Portland, Oregon, USA: ACM, 2015, pp. 189–196.

[62] Y. Li, J. Song, and S. Ermon, "Infogail: Interpretable imitation learning from visual demonstrations," in *Advances in Neural Information Processing Systems 30*, I. Guyon *et al.*, Eds., Curran Associates, Inc., 2017, pp. 3812–3822.

[63] A. Tamar *et al.*, "Imitation learning from visual data with multiple intentions," in *International Conference on Learning Representations*, 2018.

[64] F.-I. Hsiao, J.-H. Kuo, and M. Sun, "Learning a multi-modal policy via imitating demonstrations with mixed behaviors," *ArXiv*, vol. abs/1903.10304, 2019.

[65] M. Gombolay, R. Jensen, J. Stigile, S.-H. Son, and J. Shah, "Apprenticeship scheduling: Learning to schedule from human experts," in *IJCAI*, 2016.

[66] C. Rudin, C. Chen, Z. Chen, H. Huang, L. Semenova, and C. Zhong, "Interpretable machine learning: Fundamental principles and 10 grand challenges," *ArXiv*, vol. abs/2103.11251, 2021.

[67] A. Suárez and J. F. Lutsko, "Globally optimal fuzzy decision trees for classification and regression," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 12, pp. 1297–1311, 1999.

[68] Y. Bengio, N. Léonard, and A. C. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *ArXiv*, vol. abs/1308.3432, 2013.

[69] G. Brockman *et al.*, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[70] E. Leurent, *An environment for autonomous driving decision-making*, https://github.com/eleure
env, 2018.

[71] C. Wu, A. Kreidieh, K. Parvate, E. Vinitsky, and A. M. Bayen, "Flow: Architecture and benchmarking for reinforcement learning in traffic control," *ArXiv*, vol. abs/1710.05465, 2017.

[72] M. Johnson, K. Hofmann, T. Hutton, and D. Bignell, "The malmo platform for artificial intelligence experimentation," in *IJCAI*, 2016.

[73] M. Carroll *et al.*, "On the utility of learning about humans for human-ai coordination," in *NeurIPS*, 2019.

[74] D. Strouse, K. R. McKee, M. M. Botvinick, E. Hughes, and R. Everett, "Collaborating with humans without human data," M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, Eds., pp. 14 502–14 515, 2021.

[75] M. Gombolay, A. Bair, C. Huang, and J. Shah, "Computational design of mixed-initiative human–robot teaming that considers human factors: Situational awareness, workload, and workflow preferences," *The International journal of robotics research*, vol. 36, no. 5-7, pp. 597–617, 2017.

[76] S. Konan, E. Seraj, and M. Gombolay, *Iterated reasoning with mutual information in cooperative and byzantine decentralized teaming*, 2022. arXiv: 2201.08484 [cs.MA].

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

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

[79] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, *Counterfactual multi-agent policy gradients*, S. A. McIlraith and K. Q. Weinberger, Eds., 2018.

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

[81] T. Haarnoja *et al.*, "Soft actor-critic algorithms and applications," *CoRR*, vol. abs/1812.05905, 2018. arXiv: 1812.05905.

[82] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *Handbook of Reinforcement Learning and Control*, pp. 321–384, 2021.

[83] C. Zhang and V. R. Lesser, "Coordinating multi-agent reinforcement learning with limited communication.," in *International Conference on Autonomous Agents and Multiagent Systems*, 2013, pp. 1101–1108.

[84] T. Chu, S. Chinchali, and S. Katti, "Multi-agent reinforcement learning for networked system control," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, OpenReview.net, 2020.

[85] E. Pesce and G. Montana, "Improving coordination in small-scale multi-agent deep reinforcement learning through memory-driven communication," *Machine Learning*, pp. 1–21, 2020.

[86] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, D. D.

Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, Eds., 2016, pp. 2137–2145.

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

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

[89] Z. Wang, C. Liu, and M. C. Gombolay, "Heterogeneous graph attention networks for scalable multi-robot scheduling with temporospatial constraints," 1, vol. 46, 2022, pp. 249–268.

[90] Z. Wang and M. C. Gombolay, "Learning scheduling policies for multi-robot coordination with graph attention networks," *IEEE Robotics Autom. Lett.*, vol. 5, no. 3, pp. 4509–4516, 2020.

[91] J. Sheng *et al.*, "Learning structured communication for multi-agent reinforcement learning," N. Agmon, B. An, A. Ricci, and W. Yeoh, Eds., pp. 436–438, 2023.

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

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

[94] S. Li, J. K. Gupta, P. Morales, R. E. Allen, and M. J. Kochenderfer, "Deep implicit coordination graphs for multi-agent reinforcement learning," F. Dignum, A. Lomuscio, U. Endriss, and A. Nowé, Eds., pp. 764–772, 2021.

[95] J. Jiang, C. Dun, T. Huang, and Z. Lu, "Graph convolutional reinforcement learning," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, OpenReview.net, 2020.

[96] M. Bravo, J. A. Reyes-Ortiz, J. Rodríguez, and B. Silva-López, "Multi-agent communication heterogeneity," in *2015 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2015, pp. 583–588.

[97] S. Xiong, Q. Wu, and Y. Wang, "Distributed coordination of heterogeneous multi-agent systems with output feedback control," in *2019 IEEE International Conference on Unmanned Systems and Artificial Intelligence (ICUSAI)*, IEEE, 2019, pp. 106–111.

[98] D. D. R. Meneghetti and R. A. d. C. Bianchi, "Towards heterogeneous multi-agent reinforcement learning with graph neural networks," *arXiv preprint arXiv:2009.13161*, 2020.

[99] T. Mischel, "Psychology and explanations of human behavior," *Philosophy and Phenomenological Research*, vol. 23, no. 4, pp. 578–594, 1963.

[100] A. Tabrez, M. B. Luebbers, and B. Hayes, "A survey of mental modeling techniques in human–robot teaming," *Current Robotics Reports*, vol. 1, pp. 259–267, 2020.

[101] S. Maghsudi and M. Davy, "Computational models of human decision-making with application to the internet of everything," *IEEE Wirel. Commun.*, vol. 28, no. 1, pp. 152–159, 2021.

[102] S. Amershi, M. Cakmak, W. B. Knox, and T. Kulesza, "Power to the people: The role of humans in interactive machine learning," *AI Magazine*, vol. 35, pp. 105–120, 2014.

[103] C.-M. Huang and B. Mutlu, "Learning-based modeling of multimodal behaviors for humanlike robots," in *Proceedings of the 2014 ACM/IEEE International Conference on Human-robot Interaction*, ser. HRI '14, Bielefeld, Germany: ACM, 2014, pp. 57–64, ISBN: 978-1-4503-2658-2.

[104] A. Coates, P. Abbeel, and A. Y. Ng, "Apprenticeship learning for helicopter control," *Communications of the ACM*, vol. 52, no. 7, pp. 97–105, 2009.

[105] L. Chen, R. Paleja, and M. Gombolay, "Learning from suboptimal demonstration via self-supervised reward regression," in *CoRL*, 2020.

[106] L. Chen, S. Jayanthi, R. R. Paleja, D. Martin, V. Zakharov, and M. Gombolay, "Fast lifelong adaptive inverse reinforcement learning from demonstrations," in *Conference on Robot Learning*, PMLR, 2023, pp. 2083–2094.

[107] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annu. Rev. Control. Robotics Auton. Syst.*, vol. 3, pp. 297–330, 2020.

[108] M. Chen, S. Nikolaidis, H. Soh, D. Hsu, and S. Srinivasa, "Planning with trust for human-robot collaboration," in *Proceedings of the 2018 ACM/IEEE international conference on human-robot interaction*, 2018, pp. 307–315.

[109] R. Gervasi, K. Aliev, L. Mastrogiacomo, and F. Franceschini, "User experience and physiological response in human-robot collaboration: A preliminary investigation," *Journal of Intelligent & Robotic Systems*, vol. 106, no. 2, p. 36, 2022.

[110] A. Ramachandran, S. S. Sebo, and B. Scassellati, "Personalized robot tutoring using the assistive tutor pomdp (at-pomdp)," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 8050–8057.

[111] J. P. Vasconez, D. Carvajal, and F. A. Cheein, "On the design of a human–robot interaction strategy for commercial vehicle driving based on human cognitive parameters," *Advances in Mechanical Engineering*, vol. 11, no. 7, p. 1 687 814 019 862 715, 2019.

[112] M. Gombolay, C. Huang, and J. Shah, "Coordination of human-robot teaming with human task preferences," in *AAAI Fall Symposium Series on Artificial Intelligence for Human-Robot Interaction*, 2015.

[113] M. C. Gombolay, R. Jensen, and S.-H. Son, "Machine learning techniques for analyzing training behavior in serious gaming," *IEEE Transactions on Computational Intelligence and AI in Games*, 2017.

[114] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, pp. 206–215, 2018.

[115] P. M. Berry, M. Gervasio, B. Peintner, and N. Yorke-Smith, "Ptime: Personalized assistance for calendaring," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 4, 40:1–40:22, Jul. 2011.

[116] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020.

[117] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, "Explainable ai: A review of machine learning interpretability methods," *Entropy*, vol. 23, 2021.

[118] J. Kim and J. F. Canny, "Interpretable learning for self-driving cars by visualizing causal attention," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2961–2969, 2017.

[119] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *ArXiv*, vol. abs/1707.06347, 2017.

[120] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," *CoRR*, vol. abs/1509.02971, 2016.

[121] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," Proceedings of Machine Learning Research, vol. 80, J. G. Dy and A. Krause, Eds., pp. 1582–1591, 2018.

[122] J. Basak, "Online adaptive decision trees," *Neural computation*, vol. 16, no. 9, pp. 1959–1981, 2004.

[123] C. Olaru and L. Wehenkel, "A complete fuzzy decision tree technique," *Fuzzy sets and systems*, vol. 138, no. 2, pp. 221–254, 2003.

[124] E. Angelino, N. Larus-Stone, D. Alabi, M. Seltzer, and C. Rudin, "Learning certifiably optimal rule lists for categorical data," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 8753–8830, 2017.

[125] S. M. Weiss and N. Indurkhya, "Rule-based machine learning methods for functional prediction," *Journal of Artificial Intelligence Research*, vol. 3, pp. 383–403, 1995.

[126] C. Chen and C. Rudin, "An optimization approach to learning falling rule lists," Proceedings of Machine Learning Research, vol. 84, A. J. Storkey and F. Pérez-Cruz, Eds., pp. 604–612, 2018.

[127] D. Laptev and J. M. Buhmann, "Convolutional decision trees for feature learning and segmentation," in *German Conference on Pattern Recognition*, Springer, 2014, pp. 95–106.

[128] P. Kontschieder, M. Fiterau, A. Criminisi, and S. Rota Bulo, "Deep neural decision forests," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1467–1475.

[129] R. Tanno, K. Arulkumaran, D. C. Alexander, A. Criminisi, and A. V. Nori, "Adaptive neural trees," Proceedings of Machine Learning Research, vol. 97, K. Chaudhuri and R. Salakhutdinov, Eds., pp. 6166–6175, 2019.

[130] M. T. Correia Ribeiro, "Model-agnostic explanations and evaluation of machine learning," Ph.D. dissertation, University of Washington, 2018.

[131] A. Silva and M. C. Gombolay, "Encoding human domain knowledge to warm start reinforcement learning," in *AAAI*, 2021.

[132] M. Wu *et al.*, "Regional tree regularization for interpretability in deep neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 6413–6421.

[133] O. Bastani, Y. Pu, and A. Solar-Lezama, "Verifiable reinforcement learning via policy extraction," *Advances in neural information processing systems*, vol. 31, 2018.

[134] Z. C. Lipton, "The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery.," *Queue*, vol. 16, no. 3, pp. 31–57, 2018.

[135] A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (xai)," *IEEE access*, vol. 6, pp. 52 138–52 160, 2018.

[136] A. N. Sheth, K. Roy, and M. Gaur, "Neurosymbolic ai - why, what, and how," *ArXiv*, vol. abs/2305.00813, 2023.

[137] P. Hitzler, A. Eberhart, M. Ebrahimi, M. K. Sarker, and L. Zhou, "Neuro-symbolic approaches in artificial intelligence," *National Science Review*, vol. 9, 2022.

[138] A. G. Puranic, J. V. Deshmukh, and S. Nikolaidis, "Learning from demonstrations using signal temporal logic," in *Conference on Robot Learning*, 2021.

[139] G. D. Konidaris, L. P. Kaelbling, and T. Lozano-Perez, "From skills to symbols: Learning symbolic representations for abstract high-level planning," *J. Artif. Intell. Res.*, vol. 61, pp. 215–289, 2018.

[140] J. Y. Chen and M. Barnes, "Human–agent teaming for multirobot control: A review of human factors issues," *IEEE Transactions on Human-Machine Systems*, vol. 44, pp. 13–29, 2014.

[141] M. L. Schrum *et al.*, "Effects of social factors and team dynamics on adoption of collaborative robot autonomy," in *Proceedings of the 2021 ACM/IEEE International Conference on Human-Robot Interaction*, 2021, pp. 149–157.

[142] R. Paleja, "Mutual understanding in human-machine teaming," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2022, pp. 12 896–12 897.

[143] M. Natarajan *et al.*, "Human-robot teaming: Grand challenges," *Current Robotics Reports*, pp. 1–20, 2023.

[144] V. Raman and H. Kress-Gazit, "Explaining impossible high-level robot behaviors," *IEEE Trans. Robotics*, vol. 29, no. 1, pp. 94–104, 2013.

[145] S. Tellex, R. A. Knepper, A. Li, D. Rus, and N. Roy, "Asking for help using inverse semantics," in *Robotics: Science and Systems*, 2014.

[146] B. Hayes and J. Shah, "Improving robot controller transparency through autonomous policy explanation," *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*, pp. 303–312, 2017.

[147] V. Unhelkar, S. Li, and J. Shah, "Decision-making for bidirectional communication in sequential human-robot collaborative tasks," *2020 15th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 329–341, 2020.

[148] S. Rosenthal, S. P. Selvaraj, and M. Veloso, "Verbalization: Narration of autonomous robot experience," in *IJCAI*, 2016.

[149] T. Chakraborti, K. Talamadupula, Y. Zhang, and S. Kambhampati, "Interaction in human-robot societies," 2015.

[150] V. Lai and C. Tan, "On human predictions with explanations and predictions of machine learning models: A case study on deception detection," *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 2019.

[151] G. Bansal *et al.*, "Does the whole exceed its parts? the effect of ai explanations on complementary team performance," *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021.

[152] A. V. González, G. Bansal, A. Fan, R. Jia, Y. Mehdad, and S. Iyer, "Human evaluation of spoken vs. visual explanations for open-domain qa," *ArXiv*, vol. abs/2012.15075, 2020.

[153] A. Bussone, S. Stumpf, and D. O'Sullivan, "The role of explanations on trust and reliance in clinical decision support systems," *2015 International Conference on Healthcare Informatics*, pp. 160–169, 2015.

[154] Z. Zhang, Y. Genc, D. Wang, M. Ahsen, and X. Fan, "Effect of ai explanations on human perceptions of patient-facing ai-powered healthcare systems," *Journal of Medical Systems*, vol. 45, 2021.

[155] Y. Zhang, Q. Liao, and R. Bellamy, "Effect of confidence and explanation on accuracy and trust calibration in ai-assisted decision making," *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 2020.

[156] A. Anderson *et al.*, "Mental models of mere mortals with explanations of reinforcement learning," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 10, pp. 1–37, 2020.

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

[158] I. Grondman, L. Buşoniu, G. A. D. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, pp. 1291–1307, 2012.

[159] M. Babaeizadeh, I. Frosio, S. Tyree, J. Clemons, and J. Kautz, "Reinforcement learning through asynchronous advantage actor-critic on a GPU," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[160] G. Tesauro, "Temporal difference learning and td-gammon," *Communications of the ACM*, vol. 38, no. 3, pp. 58–68, 1995.

[161] R. S. Sutton, "Temporal credit assignment in reinforcement learning," Ph.D. dissertation, University of Massachusetts Amherst, 1984.

[162] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[163] J. Zhou *et al.*, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

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

[165] P. Tambwekar, A. Silva, N. Gopalan, and M. C. Gombolay, "Natural language specification of reinforcement learning policies through differentiable decision trees," *IEEE Robotics Autom. Lett.*, vol. 8, no. 6, pp. 3621–3628, 2023.

[166] A. Oroojlooy and D. Hajinezhad, "A review of cooperative multi-agent deep reinforcement learning," *Appl. Intell.*, vol. 53, no. 11, pp. 13 677–13 722, 2023.

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

[168] C. Berner *et al.*, "Dota 2 with large scale deep reinforcement learning," *CoRR*, vol. abs/1912.06680, 2019. arXiv: 1912.06680.

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

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

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

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

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

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

[175] S. Wilson *et al.*, "The robotarium: Globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multi-robot systems," *IEEE Control Systems*, vol. 40, pp. 26–44, 2020.

[176] J. E. Mathieu, T. S. Heffner, G. F. Goodwin, E. Salas, and J. A. Cannon-Bowers, "The influence of shared mental models on team process and performance.," *Journal of applied psychology*, vol. 85, no. 2, p. 273, 2000.

[177] C. Yu, X. Wang, and Z. Feng, "Coordinated multiagent reinforcement learning for teams of mobile sensing robots," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, 2019, pp. 2297–2299.

[178] J. M. Catacora Ocana, F. Riccio, R. Capobianco, and D. Nardi, "Cooperative multi-agent deep reinforcement learning in soccer domains," in *Proceedings*

of the 18th International Conference on Autonomous Agents and MultiAgent Systems, 2019, pp. 1865–1867.

[179] Y. Du *et al.*, "Learning correlated communication topology in multi-agent reinforcement learning," in *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, 2021, pp. 456–464.

[180] H. Mao, Z. Gong, Y. Ni, and Z. Xiao, "Accnet: Actor-coordinator-critic net for" learning-to-communicate" with deep multi-agent reinforcement learning," *arXiv preprint arXiv:1706.03235*, 2017.

[181] H. Ravichandar, K. Shaw, and S. Chernova, "Strata: Unified framework for task assignments in large teams of heterogeneous agents.," *(JAAMAS)*, vol. 34, no. 2, p. 38, 2020.

[182] E. Seraj, L. Chen, and M. C. Gombolay, "A hierarchical coordination framework for joint perception-action tasks in composite robot teams," *IEEE Transactions on Robotics*, 2021.

[183] E. Seraj, A. Silva, and M. Gombolay, "Safe coordination of human-robot firefighting teams," *arXiv preprint arXiv:1903.06847*, 2019.

[184] M. J. Bays and T. A. Wettergren, "A solution to the service agent transport problem," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2015, pp. 6443–6450.

[185] I. F. Akyildiz and I. H. Kasimoglu, "Wireless sensor and actor networks: Research challenges," *Ad hoc networks*, vol. 2, no. 4, pp. 351–367, 2004.

[186] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artif. Intell.*, vol. 101, pp. 99–134, 1998.

[187] S. Levine, "Policy gradients," *CS 294-112: Deep Reinforcment Learning*, 2018.

[188] E. Seraj, X. Wu, and M. Gombolay, "Firecommander: An interactive, probabilistic multi-agent environment for joint perception-action tasks," *arXiv e-prints*, arXiv–2011, 2020.

[189] M. A. Finney, *FARSITE, Fire Area Simulator–model development and evaluation*. US Department of Agriculture, Forest Service, Rocky Mountain Research Station, 1998.

[190] S. W. Smith *et al.*, "The scientist and engineer's guide to digital signal processing," 1997.

[191] L. Pimentel, R. R. Paleja, Z. Wang, E. Seraj, J. Pagan, and M. C. Gombolay, "Scaling multi-agent reinforcement learning via state upsampling," 2010.

[192] C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of ppo in cooperative, multi-agent games," *arXiv preprint arXiv:2103.01955*, 2021.

[193] B. Griffin, *New report shows manufacturing output hit $35 trillion in 2017*, Jul. 2019.

[194] Microsmallcap.com, *Why tech could shakeup the $8.1 trillion global logistics industry*, Nov. 2018.

[195] D. Chapman, "Planning for conjunctive goals," *Artificial Intelligence*, vol. 32, no. 3, pp. 333–377, 1987.

[196] G. J. Peter, "Hands-on graduate courses in lean manufacturing (lm) emphasizing green and total productive maintenance (tpm)," in *ASME 2010 International Mechanical Engineering Congress and Exposition*, American Society of Mechanical Engineers Digital Collection, 2010, pp. 357–365.

[197] P. M. Sanderson, "The human planning and scheduling role in advanced manufacturing systems: An emerging human factors domain," *Human Factors*, vol. 31, no. 6, pp. 635–666, 1989.

[198] B. L. MacCarthy, J. R. Wilson, and S. Crawford, "Human performance in industrial scheduling: A framework for understanding," *Human Factors and Ergonomics in Manufacturing & Service Industries*, vol. 11, no. 4, pp. 299–320, 2001.

[199] L. Chen, R. R. Paleja, M. Ghuy, and M. C. Gombolay, "Joint goal and strategy inference across heterogeneous demonstrators via reward network distillation," *2020 15th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 659–668, 2020.

[200] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *NIPS*, 2016.

[201] D. S. Brown, W. Goo, P. Nagarajan, and S. Niekum, "Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations," in *ICML*, 2019.

[202] L. Bottou *et al.*, "Counterfactual reasoning and learning systems: The example of computational advertising," *J. Mach. Learn. Res.*, vol. 14, no. 1, pp. 3207–3260, Jan. 2013.

[203] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web," in *Proceedings of the 7th International World Wide Web Conference*, Brisbane, Australia, 1998, pp. 161–172.

[204] R. Jin, H. Valizadegan, and H. Li, "Ranking refinement and its application to information retrieval," in *Proceedings of the 17th International Conference on World Wide Web*, ser. WWW '08, Beijing, China: ACM, 2008, pp. 397–406, ISBN: 978-1-60558-085-2.

[205] T. Pahikkala, E. Tsivtsivadze, A. Airola, J. Boberg, and T. Salakoski, "Learning to rank with pairwise regularized least-squares," *SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*, Jan. 2007.

[206] H. E. Robbins, "A stochastic approximation method," *Annals of Mathematical Statistics*, vol. 22, pp. 400–407, 2007.

[207] 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*, JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[208] P. Tambwekar, A. Silva, N. Gopalan, and M. C. Gombolay, "Specifying and interpreting reinforcement learning policies through simulatable machine learning," 2021.

[209] C. Olah *et al.*, "The building blocks of interpretability," *Distill*, vol. 3, no. 3, e10, 2018.

[210] L. A. Hendricks, R. Hu, T. Darrell, and Z. Akata, "Generating counterfactual explanations with natural language," *arXiv preprint arXiv:1806.09809*, 2018.

[211] L. Anne Hendricks, R. Hu, T. Darrell, and Z. Akata, "Grounding visual explanations," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 264–279.

[212] S. Paepcke and L. Takayama, "Judging a bot by its cover: An experiment on expectation setting for personal robots," in *Proceedings of the 5th ACM/IEEE International Conference on Human-robot Interaction*, ser. HRI '10, Osaka, Japan: IEEE Press, 2010, pp. 45–52, ISBN: 978-1-4244-4893-7.

[213] A. Gawande, *The Checklist Manifesto: How to Get Things Right*. Henry Holt and Company, 2010, ISBN: 9781429953382.

[214] A. B. Haynes *et al.*, "A surgical safety checklist to reduce morbidity and mortality in a global population," *New England Journal of Medicine*, vol. 360, no. 5, pp. 491–499, 2009, PMID: 19144931. eprint: https://doi.org/10.1056/NEJMsa0810119.

[215] M. Natarajan and M. Gombolay, "Effects of anthropomorphism and accountability on trust in human robot interaction," *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, 2020.

[216] J. M. Jumper *et al.*, "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, pp. 583–589, 2021.

[217] J. Cui, W. Macke, H. Yedidsion, A. Goyal, D. Urielli, and P. Stone, "Scalable multiagent driving policies for reducing traffic congestion," *ArXiv*, vol. abs/2103.00058, 2021.

[218] C. Katrakazas, M. A. Quddus, W.-H. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transportation Research Part C-emerging Technologies*, vol. 60, pp. 416–442, 2015.

[219] R. Abe, "Introducing autonomous buses and taxis: Quantifying the potential benefits in japanese transportation systems," *Transportation Research Part A: Policy and Practice*, 2019.

[220] A. Banerjee and R. Padhi, "Nonlinear guidance and autopilot design for lunar soft landing," in *2018 AIAA Guidance, Navigation, and Control Conference*, 2018, p. 1872.

[221] B. Kim, "Interactive and interpretable machine learning models for human machine collaboration," 2015.

[222] G. Ciravegna *et al.*, "Logic explained networks," *Artif. Intell.*, vol. 314, p. 103 822, 2023.

[223] N. Frosst and G. E. Hinton, "Distilling a neural network into a soft decision tree," *ArXiv*, vol. abs/1711.09784, 2017.

[224] M. Wu, M. C. Hughes, S. Parbhoo, M. Zazzi, V. Roth, and F. Doshi-Velez, "Beyond sparsity: Tree regularization of deep models for interpretability," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and*

Eighth AAAI Symposium on Educational Advances in Artificial Intelligence, ser. AAAI'18/IAAI'18/EAAI'18, New Orleans, Louisiana, USA: AAAI Press, 2018, pp. 1670–1678, ISBN: 978-1-57735-800-8.

[225] O. Bastani, Y. Pu, and A. Solar-Lezama, "Verifiable reinforcement learning via policy extraction," *Advances in neural information processing systems*, vol. 31, 2018.

[226] O. Nachum, S. S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," *Advances in neural information processing systems*, vol. 31, 2018.

[227] A. Ghose and B. Ravindran, "Interpretability with accurate small models," *Frontiers in Artificial Intelligence*, vol. 3, 2020.

[228] H. Lakkaraju, S. H. Bach, and J. Leskovec, "Interpretable decision sets: A joint framework for description and prediction," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16, San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 1675–1684, ISBN: 9781450342322.

[229] D. Hein, S. Limmer, and T. A. Runkler, "Interpretable control by reinforcement learning," *ArXiv*, vol. abs/2007.09964, 2020.

[230] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[231] D. Hafner, T. P. Lillicrap, M. Norouzi, and J. Ba, "Mastering atari with discrete world models," 2021.

[232] G. V. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, pp. 303–314, 1989.

[233] R. R. Selmic and F. L. Lewis, "Neural-network approximation of piecewise continuous functions: Application to friction compensation," *IEEE transactions on neural networks*, vol. 13 3, pp. 745–51, 2002.

[234] H. Chen, H. Zhang, S. Si, Y. Li, D. Boning, and C.-J. Hsieh, "Robustness verification of tree-based models," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[235] C. Szegedy *et al.*, "Intriguing properties of neural networks," *CoRR*, vol. abs/1312.6199, 2013.

[236] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, 2016.

[237] G. Katz, C. W. Barrett, D. L. Dill, K. D. Julian, and M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," *ArXiv*, vol. abs/1702.01135, 2017.

[238] H. Salman, G. Yang, H. Zhang, C.-J. Hsieh, and P. Zhang, "A convex relaxation barrier to tight robustness verification of neural networks," in *Neural Information Processing Systems*, 2019.

[239] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 5026–5033.

[240] I. Parberry, *Introduction to Game Physics with Box2D*. CRC Press, 2017.

[241] W. Gibaut *et al.*, "Neurosymbolic AI and its taxonomy: A survey," *CoRR*, vol. abs/2305.08876, 2023. arXiv: 2305.08876.

[242] P. A. Lopez *et al.*, "Microscopic traffic simulation using sumo," in *2018 21st international conference on intelligent transportation systems (ITSC)*, IEEE, 2018, pp. 2575–2582.

[243] J. A. Damico and D. A. Wolfe, "Extended tables of the exact distribution of a rank statistic for all treatments multiple comparisons in one-way layout designs," *Communications in Statistics-Theory and Methods*, vol. 16, no. 8, pp. 2343–2360, 1987.

[244] P. Stone, G. Kaminka, S. Kraus, and J. Rosenschein, "Ad hoc autonomous agent teams: Collaboration without pre-coordination," in *AAAI*, 2010.

[245] R. Mirsky, W. Macke, A. Wang, H. Yedidsion, and P. Stone, "A penny for your thoughts: The value of communication in ad hoc teamwork," in *IJCAI*, 2020.

[246] B. A. A. White, A. Eklund, T. Mcneal, A. Hochhalter, and A. Arroliga, "Facilitators and barriers to ad hoc team performance," *Baylor University Medical Center Proceedings*, vol. 31, pp. 380–384, 2018.

[247] J. Shah, "Human-robot teaming using shared mental models," 2012.

[248] S. Nikolaidis, P. A. Lasota, R. Ramakrishnan, and J. Shah, "Improved human–robot team performance through cross-training, an approach inspired

by human team training practices," *The International Journal of Robotics Research*, vol. 34, pp. 1711–1730, 2015.

[249] J. Boyd, "The essence of winning and losing," 2012.

[250] N. Sebanz, H. Bekkering, and G. Knoblich, "Joint action: Bodies and minds moving together," *Trends in Cognitive Sciences*, vol. 10, pp. 70–76, Mar. 2006.

[251] D. Gunning, "Darpa's explainable artificial intelligence (xai) program," *Proceedings of the 24th International Conference on Intelligent User Interfaces*, 2019.

[252] S. Mohammed, K. Hamilton, M. Sánchez-Manzanares, and R. Rico, "Team cognition," in *The Wiley Blackwell Handbook of the Psychology of Team Working and Collaborative Processes*. John Wiley & Sons, Ltd, 2017, ch. 16, pp. 369–392, ISBN: 9781118909997. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118909997.ch16.

[253] W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, and K. Müller, "Explainable ai: Interpreting, explaining and visualizing deep learning," *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, 2019.

[254] Y. Wang, M. Huang, X. Zhu, and L. Zhao, "Attention-based lstm for aspect-level sentiment classification," in *EMNLP*, 2016.

[255] M. Endsley, "Situation awareness misconceptions and misunderstandings," *Journal of Cognitive Engineering and Decision Making*, vol. 9, pp. 32–4, 2015.

[256] M. Endsley, "Situation awareness global assessment technique (sagat)," in *Proceedings of the IEEE 1988 National Aerospace and Electronics Conference*, 1988, 789–795 vol.3.

[257] C. Amato, G. Chowdhary, A. Geramifard, N. K. Ure, and M. J. Kochenderfer, "Decentralized control of partially observable markov decision processes using belief space macro-actions," *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5962–5969, 2015.

[258] A. Silva, M. Gombolay, T. W. Killian, I. D. J. Jimenez, and S.-H. Son, "Optimization methods for interpretable differentiable decision trees applied to reinforcement learning," in *AISTATS*, 2020.

[259] S. Seo, L. R. Kennedy-Metz, M. Zenati, J. Shah, R. Dias, and V. Unhelkar, "Towards an ai coach to infer team mental model alignment in healthcare," *ArXiv*, vol. abs/2102.08507, 2021.

[260] L. Sanneman and J. Shah, "A situation awareness-based framework for design and evaluation of explainable ai," *Explainable, Transparent Autonomous Agents and Multi-Agent Systems*, vol. 12175, pp. 94–110, 2020.

[261] D. Sirkin, N. Martelaro, M. Johns, and W. Ju, "Toward measurement of situation awareness in autonomous vehicles," *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, 2017.

[262] D. McFarlane and K. Latorella, "The scope and importance of human interruption in human-computer interaction design," *Human–Computer Interaction*, vol. 17, pp. 1–61, 2002.

[263] G. Mark, D. Gudith, and U. Klocke, "The cost of interrupted work: More speed and stress," in *CHI*, 2008.

[264] J. Westbrook, M. Raban, S. Walter, and H. E. Douglas, "Task errors by emergency physicians are associated with interruptions, multitasking, fatigue and working memory capacity: A prospective, direct observation study," *BMJ Quality & Safety*, vol. 27, pp. 655–663, 2018.

[265] A. Goldman, "Theory of mind," 2012.

[266] M. S. Chmielewski and T. A. Morgan, "Five-factor model of personality," in *Encyclopedia of Behavioral Medicine*. New York, NY: Springer New York, 2013, pp. 803–804, ISBN: 978-1-4419-1005-9.

[267] G. Hoffman, "Evaluating fluency in human–robot collaboration," *IEEE Transactions on Human-Machine Systems*, vol. 49, pp. 209–218, 2019.

[268] A. Aron, E. Aron, M. Tudor, and G. Nelson, "Close relationships as including other in the self," *Journal of Personality and Social Psychology*, vol. 60, pp. 241–253, 1991.

[269] C. Bartneck, D. Kulic, E. Croft, and S. Zoghbi, "Godspeed questionnaire series," 2019.

[270] S. Hart and L. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research," *Advances in psychology*, vol. 52, pp. 139–183, 1988.

[271] C. Liu and M. Tomizuka, "Algorithmic safety measures for intelligent industrial co-robots," *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3095–3102, 2016.

[272] M. J. Mataric, "Robots for the people, by the people: Personalizing human-machine interaction," *Sci. Robotics*, vol. 3, no. 21, 2018.

[273] J. L. Wright, S. G. Lakhmani, and J. Y. C. Chen, "Bidirectional communications in human-agent teaming: The effects of communication style and feedback," *International Journal of Human–Computer Interaction*, vol. 38, pp. 1972–1985, 2022.

[274] A. Butchibabu, C. Sparano-Huiban, L. Sonenberg, and J. Shah, "Implicit coordination strategies for effective team communication," *Human factors*, vol. 58, no. 4, pp. 595–610, 2016.

[275] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013. arXiv: 1312.5602.

[276] K. Guerin, C. S. Lea, C. Paxton, and G. Hager, "A framework for end-user instruction of a robot assistant for manufacturing," *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6167–6174, 2015.

[277] C. Paxton, A. T. Hundt, F. Jonathan, K. Guerin, and G. Hager, "Costar: Instructing collaborative robots with behavior trees and vision," *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 564–571, 2017.

[278] C. Paxton, F. Jonathan, A. T. Hundt, B. Mutlu, and G. Hager, "Evaluating methods for end-user creation of robot task plans," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6086–6092, 2018.

[279] D. Fogli, L. Gargioni, G. Guida, and F. Tampalini, "A hybrid approach to user-oriented programming of collaborative robots," *Robotics Comput. Integr. Manuf.*, vol. 73, p. 102 234, 2022.

[280] B. Beyret, A. Shafti, and A. Faisal, "Dot-to-dot: Explainable hierarchical reinforcement learning for robotic manipulation," *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5014–5019, 2019.

[281] B. Sarkar, A. Talati, A. Shih, and S. Dorsa, "Pantheonrl: A marl library for dynamic training interactions," in *Proceedings of the 36th AAAI Conference on Artificial Intelligence (Demo Track)*, 2022.

[282] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv: Learning*, 2018.

[283] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, "Pruning and quantization for deep neural network acceleration: A survey," *Neurocomputing*, vol. 461, pp. 370–403, 2021.

[284] S. Karten, M. Tucker, H. Li, S. Kailas, M. Lewis, and K. P. Sycara, "Interpretable learned emergent communication for human-agent teams," *IEEE Transactions on Cognitive and Developmental Systems*, 2022.

[285] J. Wainer, D. J. Feil-Seifer, D. A. Shell, and M. J. Mataric, "Embodiment and human-robot interaction: A task-based perspective," in *RO-MAN 2007 - The 16th IEEE International Symposium on Robot and Human Interactive Communication*, 2007, pp. 872–877.