Learning Efficient Diverse Communication for Cooperative Heterogeneous Teaming

Esmaeil Seraj[1,*], Zheyuan Wang[1,*], Rohan Paleja[1,*], Daniel Martin[1], Matthew Sklar[1], Anirudh Patel[2], Matthew Gombolay[1]
[1]Georgia Institute of Technology, [2]Sandia National Laboratory
[1]Atlanta, GA, USA, [2]Albuquerque, NM, USA
{eseraj3,pjohnwang,rpaleja3,dmartin1,msklar3}@gatech.edu,anipate@sandia.gov,matthew.gombolay@cc.gatech.edu

ABSTRACT

High-performing teams learn intelligent and efficient communication and coordination strategies to maximize their joint utility. These teams implicitly understand the different roles of heterogeneous team members and adapt their communication protocols accordingly. Multi-Agent Reinforcement Learning (MARL) seeks to develop computational methods for synthesizing such coordination strategies, but formulating models for heterogeneous teams with different state, action, and observation spaces has remained an open problem. Without properly modeling agent heterogeneity, as in prior MARL work that leverages homogeneous graph networks, communication becomes less helpful and can even deteriorate the cooperativity and team performance. We propose Heterogeneous Policy Networks (HetNet) to learn efficient and diverse communication models for coordinating cooperative heterogeneous teams. Building on heterogeneous graph-attention networks, we show that HetNet not only facilitates learning heterogeneous collaborative policies per existing agent-class but also enables end-to-end training for learning highly efficient binarized messaging. 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 an 8.1% to 434.7% performance improvement over the next-best baseline across multiple domains while simultaneously achieving a 200× reduction in the required communication bandwidth.

KEYWORDS

Cooperative MARL; Heterogeneous Teaming; Efficient Communication; Heterogeneous Graph Attention Networks

ACM Reference Format:
Esmaeil Seraj[1,*], Zheyuan Wang[1,*], Rohan Paleja[1,*], Daniel Martin[1], Matthew Sklar[1],, Anirudh Patel[2], Matthew Gombolay[1]. 2022. Learning Efficient Diverse Communication for Cooperative Heterogeneous Teaming. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 10 pages.

1 INTRODUCTION

High-performing human teams benefit from communication to build and maintain shared mental models to improve team effectiveness [24, 26]. Information sharing is key in building team cognition, and enables teammates to cooperate to successfully achieve shared goals [21, 26, 31]. Typical communication patterns across human teams widely differ based on the task or role the human assumes [43]. The field of MARL [7] has sought to develop agents that autonomously learn coordination and communication strategies to emulate high-performing human-human teams [5, 8, 46, 51, 52]. Yet, these approaches have fallen short in properly modeling heterogeneity and communication overhead in teaming [10, 11, 25, 53].

Heterogeneity in robots' design characteristics and their roles are introduced to leverage the relative merits of different agents and their capabilities [30, 32, 34, 35] 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 straightforward 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 [28, 53], 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 [2, 4, 32]. 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 [54]. More recent prior work, such as MAGIC [28], 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

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online. © 2022 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

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 (MA-HAC) 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 bandwidth 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.

2 RELATED WORK

MARL with Communication – Recently, the use of communication in MARL has been shown to enhance the collective performance of learning agents in cooperative MARL problems [9, 10, 18, 23, 29, 33, 38, 40, 50, 52, 53]. DIAL [13] and CommNet [40] displayed the capability to learn a discrete and continuous communication vectors, respectively. While DIAL considers the limited-bandwidth problem, neither of these approaches are readily applicable to composite teams or capable of performing attentional communication. TarMAC [10] achieves targeted communication through an attention mechanism which 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 [23]. SchedNet [20] explicitly addresses the bandwidth-related concerns. However, in SchedNet agents learn how to schedule themselves for accessing the communication channel, rather than learning the communication protocols from scratch. In our approach, we explicitly address the heterogeneous communication problem where agents learn diverse communication protocols and intermediate language representations to use among themselves for cooperation. Our model enables agents to perform attentional communication and sending limited-length digitized messages through class-specific encoder-decoder channels, addressing the limited-bandwidth issues.

MARL with Graph Neural Networks (GNN) – Prior work on MARL have sought to utilize GNNs to model a communication structure among agents [1]. Deep Graph Network (DGN) [17] 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 [37], an effective communication topology is proposed by using hierarchical GNNs to propagate messages among groups and agents. G2ANet [23] proposed a game abstraction method combining a hard and a soft-attention mechanism to dynamically learn interactions between agents. More recently, MAGIC [28] introduced a scalable, attentional communication model for learning a centralized scheduler to determine when to communicate and how to process messages through graph-attention networks. While these prior work have successfully modeled multi-agent interactions, they are not designed to address heterogeneous teams directly. HetNet, on the other hand, is designed to capture the heterogeneity among agents and learn an efficient shared language across agents with different action and observation spaces to improve cooperativity.

Heterogeneity in Multi-agent Systems – In [6], 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 [6]. In [49], 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 [27] utilized GNNs and Deep Deterministic Q-network (DDQN) to facilitate coordination among heterogeneous agents (i.e., those with different state and action spaces). Going beyond this prior work, we build our HetNet model based upon an actor-critic framework and generalize the problem formulation for state-, action- and observation-space heterogeneities. Moreover, HetNet facilitates learning efficient binary representations of states as an intermediate language among agents of different types to improve cooperativity.

3 PRELIMINARIES

3.1 Problem Formulation

Founding on a standard Partially Observable MDP (POMDP) [19], we formulate a new problem setup termed as Multi-Agent Heterogeneous POMDP (MAH-POMDP), which can be represented by a 9-tuple $\langle C, \mathcal{N}, \{\mathcal{S}^{(i)}\}_{i\in C}, \{\mathcal{A}^{(i)}\}_{i\in C}, \{\Omega^{(i)}\}_{i\in C}, \{O^{i}\}_{i\in C}, r, \mathcal{T}, \gamma\rangle$. 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 $\{O^{(i)}\}_{i \in C} : o_t^{i_j} \sim O^{(i)}(\cdot|\bar{s})$. 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 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., $\arg\max_{\pi(\bar{s}) \in \Pi} \mathbb{E}_{\pi(\bar{s})}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | \pi(\bar{s})\right]$.

3.2 Actor-Critic (AC) Methods

Actor-Critic (AC) methods [3, 15] are an approach to RL that utilize function approximation, in which each agent j has a policy, $\pi_\theta^j(a|s)$, 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 [44], parameterized by ϕ, where $Q^\phi(s, a)$ approximately solves the credit-assignment problem [41]. By the policy gradient theorem [42], the expected reward maximization (i.e., the AC objective), $J(\theta)$, is maximized via $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta^j}\left[\nabla_\theta \log \pi_\theta^j(a_t^j|o_t^j) Q^\phi(o_t^j, a_t^j)\right]$, where a_t^j and o_t^j are the action and observation of agent j, respectively.

3.3 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 compute a new feature vector [17, 48, 55]. 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. 1. In Eq. 1, $\bar{W}_{att}$ 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 [45, 47].

$$\alpha_{jk} = \text{softmax}_k\left(\sigma'\left(\bar{W}_{att}^T\left[\omega\bar{h}_j \,\|\, \omega\bar{h}_k\right]\right)\right) \tag{1}$$

4 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 binarized 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.

4.1 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 [14, 20].

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.

4.2 Heterogeneous Communication Model

GNNs previously used in MARL operate on homogeneous graphs to learn a universal feature update and communication scheme for all agents [17, 23, 28, 37], 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 3.1, we directly model each agent class $i \in C$ 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

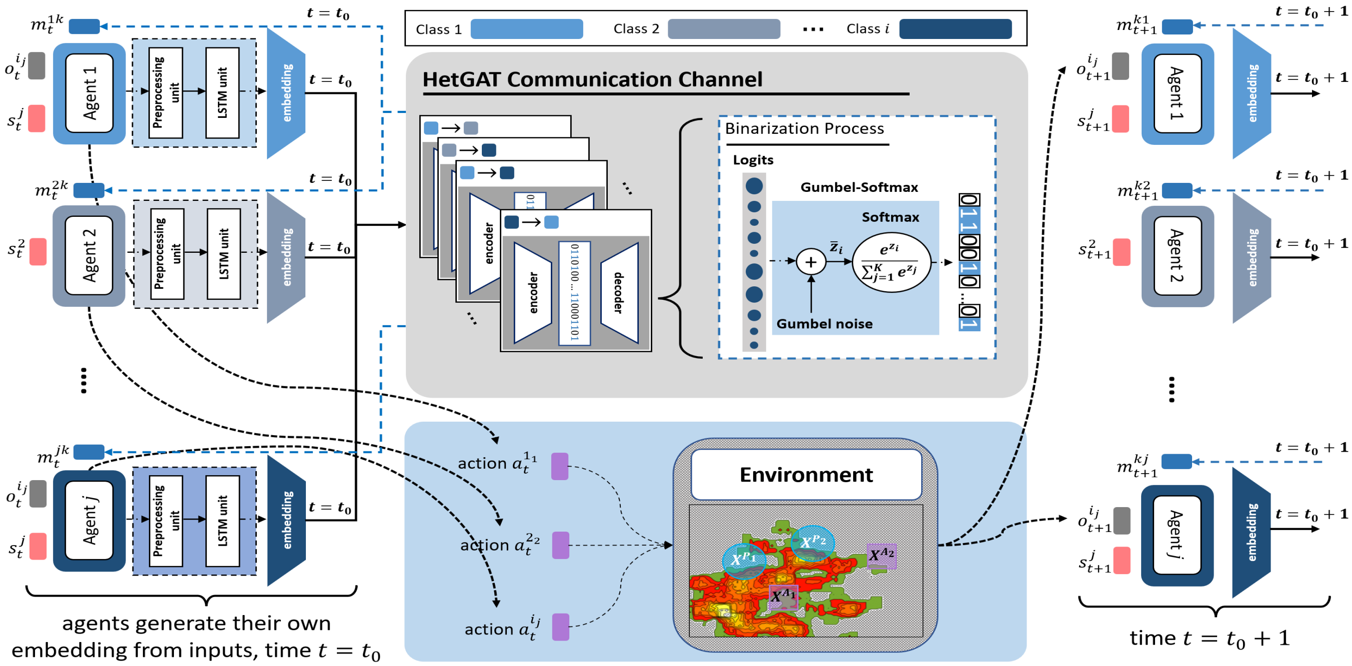


Figure 1: Overview of our multi-agent heterogeneous attentional communication architecture in a CTDE paradigm. At each time point $t = t_0$, each agent j of class i generates a local embedding from its own inputs, by passing its input data through class-specific preprocessing units (i.e., a CNN or a 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.

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 Fig. 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 Fig. 1, the generated embeddings are then passed into a HetGAT communication channel including a class-specific encoder-decoder network and a Gumbel-Softmax [16] unit to generate a binarized message, m_t, for an agent, j.

4.3 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. Fig. 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 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 the communication edge that an agent is connected

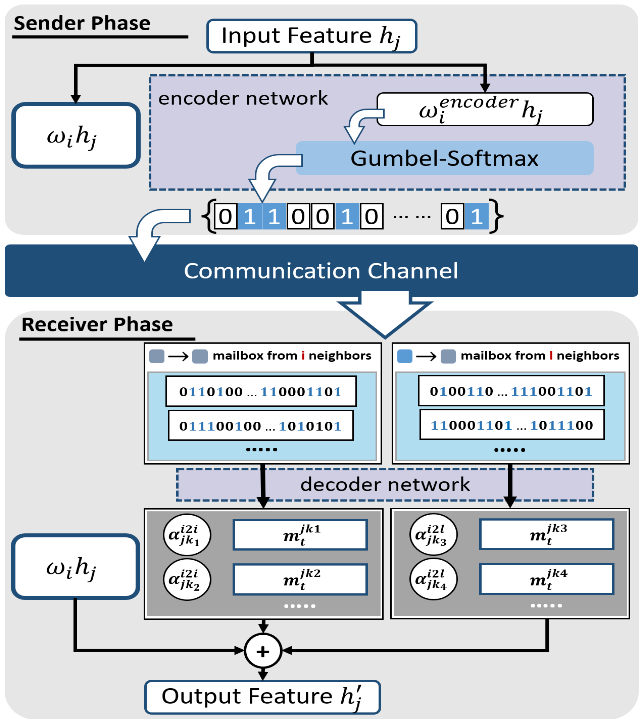


Figure 2: The sender and receiver phases of the feature update process in a HetGAT layer for one agent, j, of class i.

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. 2, 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. 3 and, $\Delta_l(j)$ include agent j's neighbors that belong to class l.

$$\text{Class } (i): \quad \bar{h}'_j = \sigma\Big(\omega_i \bar{h}_j + \sum_{l \in C} \sum_{k \in N_l(j)} \alpha_{jk}^{i2l} m_t^{jk}\Big) \tag{2}$$

$$m_t^{jk} = \omega_i^{dec}(\text{GumbelSoftmax}(\omega_l^{enc} h_k)) \tag{3}$$

Note that we have $l = i$ for intra-class communication. When computing attention coefficients in a heterogeneous graph, we adapt Eq. 1 into Eq. 4 to account for heterogeneous channels.

$$\alpha_{jk}^{i2l} = \text{softmax}_k\left(\sigma'\left(\vec{W}_{att}^T\left[\omega_i \bar{h}_j \,\|\, \omega_{i2l} \bar{h}_k\right]\right)\right) \tag{4}$$

As discussed in Section 4.2, 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. 5. 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. 4.

$$\text{SEN}: \quad \bar{h}'_s = \sigma\Big(\omega_s \bar{h}_s + \sum_{i \in C} \sum_{j \in N^{(i)}} \alpha_{sj}^{i2sen} \omega_{i2sen} \bar{h}_j\Big) \tag{5}$$

To stabilize the learning process, we adapt the multi-head extension of the attention mechanism [45] 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.

4.4 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 TRAINING AND EXECUTION

5.1 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 3.2, shown in Eq. 6. We leverage an on-policy training paradigm for MAHAC.

$$\nabla_{\theta^i} J(\theta^i) = \frac{1}{N} \sum_{j=1}^{N^{(i)}} \sum_{t=1}^{T} \nabla_{\theta^i} \log \pi^i \left(\bar{a}_t^{i_j} | \bar{o}_t^{i_j}, \bar{m}_t\right)\left(\left(\sum_{t'=t}^{T} \gamma^{t'-t} r^{i_j}(\bar{s}^{i_j}, \bar{a}^{i_j})\right) - b(t)\right) \tag{6}$$

In Eq. 6, $\bar{a}_t^{i_j}$ 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 [22][1].

5.2 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.

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

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 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 1 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.

Algorithm 1: 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

6 EMPIRICAL EVALUATION

6.1 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. Please refer to the supplementary for environment and model details.

Predator-Prey (PP) [38] – For the homogeneous domain, we adopt the Predator-Prey (PP) [38] 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 all belong to the same class (i.e., identical state, observation and action spaces).

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 *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).

FireCommander (FC) [36] – In the second heterogeneous domain, the FireCommander [36], two classes of *perception* and *action* agents must collaborate as a composite team to extinguish a propagating firespot. At each timestep, the firespot propagates to a new location according to the FARSITE [12] model, while the previous location is still on fire. All firespots are initially hidden to agents and need to be discovered before being extinguished. As such, *perception* agents are tasked to scan the environment to detect the firespots while *action* agents (no observation inputs) are required to move and extinguish a firespot that has been discovered by a *perception* agent before. Note that since firespots propagate, both *perception* and *action* agents need to continue to explore the map and collaborate until all firespots are extinguished.

6.2 Baselines

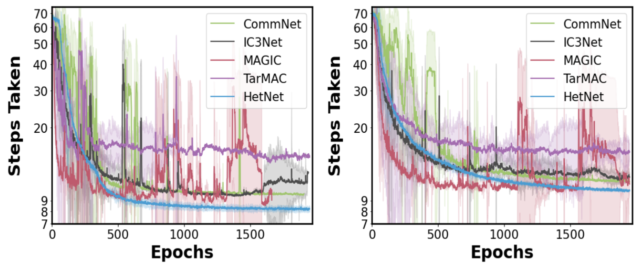
We benchmark two variants of our framework, i.e. HetNet-Binary and HetNet-Real, against four end-to-end communicative MARL baselines: (1) CommNet [40], (2) IC3Net [38], (3) TarMAC [10] and, (4) MAGIC [28]. For our HetNet-Real variant, we remove the binarization process (i.e., Gumbel-Softmax) and the 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 Section 6.3). We note that, for all four baselines, i.e. CommNet [40], IC3Net [38], TarMAC [10] and, MAGIC [28], 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`).

6.3 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

Table 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 [10]	-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 [38]	-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 [40]	-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 [28]	-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**



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

Figure 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.

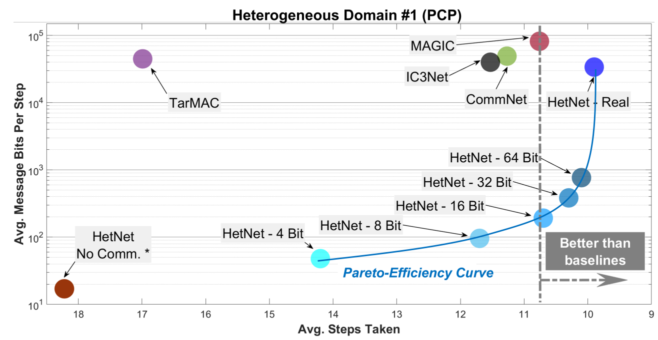


Figure 4: Communicated bits per round of communication vs. performance in PCP for different methods. HetNet facilitates binarized messages among agents which requires significantly less CB as compared to real-valued baselines.

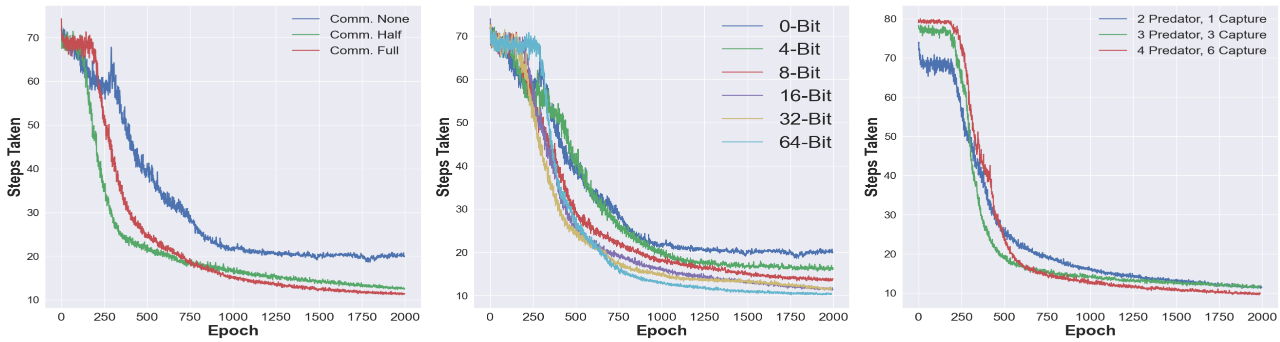
method (Section 6.3.2). We then present evidence to support the effects of communication on collaboration performance (Section 6.3.3) as well as to determine the sensitivity of HetNet to key variables such as number of agents (Section 6.3.4). Additionally, we investigate the effects of the critic structures proposed in Section 5.2 on HetNet's performance (Section 6.3.5).

6.3.1 Baseline Comparison. Fig. 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 at convergence by each of the baselines in PP, PCP and FC domains. The results of this test are presented in Table 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 Fig. 4). The heterogeneous policies learned by our model set the SOTA for learning challenging cooperative behaviors for composite teams.

6.3.2 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 Fig. 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 [39]). HetNet-Binary with 64 and 32-bits messages, respectively, achieve more than 100× and 200× lower CB while showing better performance than real-valued baselines.

6.3.3 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 Fig. 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 Fig. 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.

6.3.4 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

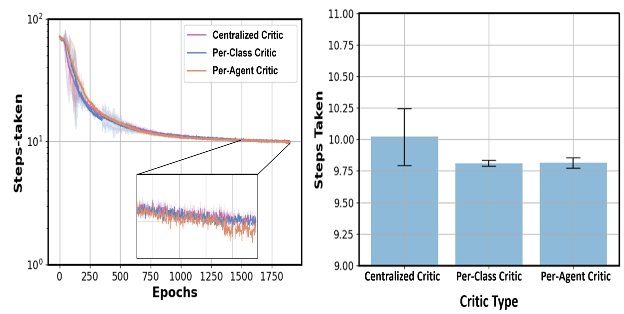


(a) Communication range. **(b) Message dimensionality.** **(c) Scalability to num. of agents.**

Figure 5: Analyzing HetNet's performance with and without communication (Fig. 5a) and across different binary message dimensions (Fig. 5b) in the PCP domain. Communication policy learned by HetNet improves the cooperativity among agents and the performance improves with larger message sizes. Fig. 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.

predator and *capture* agents, to evaluate the scalability to different team sizes. The results of this experiment are presented in Fig. 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.

6.3.5 Ablation Study #4: Effects of the Critic Structures. Finally, we investigate the utility and performance of the three critic structures proposed in Section 5.2 on HetNet's performance in the PCP domain. We utilized our HetNet-Real variant for this experiment. Fig. 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 Fig. 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 → 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 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.

7 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.

ACKNOWLEDGMENTS

This work is supported by the Naval Research Lab under grant N00173-21-1-G009, Lockheed Martin Corporation under grant GR0-0000509, Office of Naval Research (ONR) under grant N00014-19-1-2076, and supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

REFERENCES

[1] Dhaval Adjodah, Dan Calacci, Abhimanyu Dubey, Anirudh Goyal, Peter Krafft, Esteban Moro, and Alex Pentland. 2019. Communication topologies between learning agents in deep reinforcement learning. *arXiv preprint arXiv:1902.06740* (2019).

[2] Ian F Akyildiz and Ismail H Kasimoglu. 2004. Wireless sensor and actor networks: research challenges. *Ad hoc networks* 2, 4 (2004), 351–367.

[3] Mohammad Babaeizadeh, Iuri Frosio, Stephen Tyree, Jason Clemons, and Jan Kautz. 2017. 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. https://openreview.net/forum?id=r1VGvBcxl

[4] Matthew J Bays and Thomas A Wettergren. 2015. A solution to the service agent transport problem. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 6443–6450.

[5] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. 2019. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680* (2019).

[6] Maricela Bravo, José A. Reyes-Ortiz, José Rodríguez, and Blanca Silva-López. 2015. Multi-agent Communication Heterogeneity. In *2015 International Conference on Computational Science and Computational Intelligence (CSCI)*. 583–588. https://doi.org/10.1109/CSCI.2015.167

[7] L. Busoniu, R. Babuka, and B. D. Schutter. 2008. A Comprehensive Survey of Multiagent Reinforcement Learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38 (2008), 156–172.

[8] Jim Martin Catacora Ocana, Francesco Riccio, Roberto Capobianco, and Daniele Nardi. 2019. Cooperative multi-agent deep reinforcement learning in soccer domains. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. 1865–1867.

[9] Tianshu Chu, Sandeep Chinchali, and Sachin Katti. 2019. Multi-agent Reinforcement Learning for Networked System Control. In *International Conference on Learning Representations*.

[10] Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. 2019. Tarmac: Targeted multi-agent communication. In *International Conference on Machine Learning*. PMLR, 1538–1546.

[11] Yali Du, Bo Liu, Vincent Moens, Ziqi Liu, Zhicheng Ren, Jun Wang, Xu Chen, and Haifeng Zhang. 2021. Learning Correlated Communication Topology in Multi-Agent Reinforcement learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. 456–464.

[12] Mark A Finney. 1998. *FARSITE, Fire Area Simulator–model development and evaluation.* Number 4. US Department of Agriculture, Forest Service, Rocky Mountain Research Station.

[13] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. 2016. Learning to Communicate with Deep Multi-Agent Reinforcement Learning. *Advances in Neural Information Processing Systems* 29 (2016), 2137–2145.

[14] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[15] Ivo Grondman, Lucian Buşoniu, Gabriel A. D. Lopes, and Robert Babuka. 2012. A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42 (2012), 1291–1307.

[16] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* (2016).

[17] Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu. 2020. Graph Convolutional Reinforcement Learning. In *International Conference on Learning Representations (ICLR)*.

[18] Jiechuan Jiang and Zongqing Lu. 2018. Learning Attentional Communication for Multi-Agent Cooperation. *Advances in Neural Information Processing Systems* 31 (2018), 7254–7264.

[19] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. 1998. Planning and Acting in Partially Observable Stochastic Domains. *Artif. Intell.* 101 (1998), 99–134.

[20] Daewoo Kim, Sangwoo Moon, David Hostallero, Wan Ju Kang, Taeyoung Lee, Kyunghwan Son, and Yung Yi. 2018. Learning to Schedule Communication in Multi-agent Reinforcement Learning. In *International Conference on Learning Representations*.

[21] Sachin Konan, Esmaeil Seraj, and Matthew Gombolay. 2022. Iterated Reasoning with Mutual Information in Cooperative and Byzantine Decentralized Teaming. arXiv:2201.08484 [cs.MA]

[22] Sergey Levine. 2018. Policy gradients. *CS 294-112: Deep Reinforcment Learning* (2018).

[23] Yong Liu, Weixun Wang, Yujing Hu, Jianye Hao, Xingguo Chen, and Yang Gao. 2020. Multi-agent game abstraction via graph attention neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 7211–7218.

[24] Jean MacMillan, Elliot E Entin, and Daniel Serfaty. 2004. Communication overhead: The hidden cost of team cognition. (2004).

[25] Hangyu Mao, Zhibo Gong, Yan Ni, and Zhen Xiao. 2017. ACCNet: Actor-coordinator-critic net for" Learning-to-communicate" with deep multi-agent reinforcement learning. *arXiv preprint arXiv:1706.03235* (2017).

[26] John E Mathieu, Tonia S Heffner, Gerald F Goodwin, Eduardo Salas, and Janis A Cannon-Bowers. 2000. The influence of shared mental models on team process and performance. *Journal of applied psychology* 85, 2 (2000), 273.

[27] Douglas De Rizzo Meneghetti and Reinaldo Augusto da Costa Bianchi. 2020. Towards Heterogeneous Multi-Agent Reinforcement Learning with Graph Neural Networks. *arXiv preprint arXiv:2009.13161* (2020).

[28] Yaru Niu, Rohan Paleja, and Matthew Gombolay. 2021. Multi-Agent Graph-Attention Communication and Teaming. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. 964–973.

[29] Emanuele Pesce and Giovanni Montana. 2020. Improving coordination in small-scale multi-agent deep reinforcement learning through memory-driven communication. *Machine Learning* (2020), 1–21.

[30] Harish Ravichandar, Kenneth Shaw, and Sonia Chernova. 2020. STRATA: unified framework for task assignments in large teams of heterogeneous agents. *(JAAMAS)* 34, 2 (2020), 38.

[31] Eduardo Salas, Terry L Dickinson, Sharolyn A Converse, and Scott I Tannenbaum. 1992. Toward an understanding of team performance and training. (1992).

[32] Esmaeil Seraj, Letian Chen, and Matthew C Gombolay. 2021. A hierarchical coordination framework for joint perception-action tasks in composite robot teams. *IEEE Transactions on Robotics* (2021).

[33] Esmaeil Seraj and Matthew Gombolay. 2020. Coordinated control of uavs for human-centered active sensing of wildfires. In *2020 American Control Conference (ACC)*. IEEE, 1845–1852.

[34] Esmaeil Seraj, Andrew Silva, and Matthew Gombolay. 2019. Safe coordination of human-robot firefighting teams. *arXiv preprint arXiv:1903.06847* (2019).

[35] Esmaeil Seraj, Zheyuan Wang, Rohan Paleja, Matthew Sklar, Anirudh Patel, and Matthew Gombolay. 2021. Heterogeneous graph attention networks for learning diverse communication. *arXiv preprint arXiv:2108.09568* (2021).

[36] Esmaeil Seraj, Xiyang Wu, and Matthew Gombolay. 2020. FireCommander: An Interactive, Probabilistic Multi-agent Environment for Joint Perception-Action Tasks. *arXiv e-prints* (2020), arXiv–2011.

[37] Junjie Sheng, Xiangfeng Wang, Bo Jin, Junchi Yan, Wenhao Li, Tsung-Hui Chang, Jun Wang, and Hongyuan Zha. 2020. Learning structured communication for multi-agent reinforcement learning. *arXiv preprint arXiv:2002.04235* (2020).

[38] Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. 2018. Learning when to communicate at scale in multiagent cooperative and competitive tasks. *arXiv preprint arXiv:1812.09755* (2018).

[39] Steven W Smith et al. 1997. The scientist and engineer's guide to digital signal processing. (1997).

[40] Sainbayar Sukhbaatar, Rob Fergus, et al. 2016. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*. 2244–2252.

[41] Richard Stuart Sutton. 1984. *Temporal credit assignment in reinforcement learning.* Ph.D. Dissertation. University of Massachusetts Amherst.

[42] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction.* MIT press.

[43] Howard Taylor. 2007. *The effects of interpersonal communication style on task performance and well being.* Ph.D. Dissertation. University of Buckingham.

[44] Gerald Tesauro. 1995. Temporal difference learning and TD-Gammon. *Commun. ACM* 38, 3 (1995), 58–68.

[45] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations* (2018).

[46] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575, 7782 (2019), 350–354.

[47] Zheyuan Wang and Matthew Gombolay. 2020. Learning scheduling policies for multi-robot coordination with graph attention networks. *IEEE Robotics and Automation Letters* 5, 3 (2020), 4509–4516.

[48] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.

[49] Shixun Xiong, Qingxian Wu, and Yuhui Wang. 2019. Distributed Coordination of Heterogeneous Multi-Agent Systems with Output Feedback Control. In *2019 IEEE International Conference on Unmanned Systems and Artificial Intelligence (ICUSAI)*. IEEE, 106–111.

[50] Xing Xu, Rongpeng Li, Zhifeng Zhao, and Honggang Zhang. 2021. Stigmergic Independent Reinforcement Learning for Multiagent Collaboration. *IEEE Transactions on Neural Networks and Learning Systems* (2021).

[51] Chao Yu, Xin Wang, and Zhanbo Feng. 2019. Coordinated Multiagent Reinforcement Learning for Teams of Mobile Sensing Robots. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*.

2297–2299.
[52] Chongjie Zhang and Victor R Lesser. 2013. Coordinating multi-agent reinforcement learning with limited communication.. In *International Conference on Autonomous Agents and Multiagent Systems*. 1101–1108.
[53] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. 2021. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control* (2021), 321–384.
[54] Sai Qian Zhang, Qi Zhang, and Jieyu Lin. 2019. Efficient Communication in Multi-Agent Reinforcement Learning via Variance Based Control. *Advances in Neural Information Processing Systems* 32 (2019), 3235–3244.
[55] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1 (2020), 57–81.