Learning Communication Skills in Multi-task Multi-agent Deep Reinforcement Learning

Changxi Zhu
Utrecht University
Utrecht, the Netherlands
c.zhu@uu.nl

Mehdi Dastani
Utrecht University
Utrecht, the Netherlands
m.m.dastani@uu.nl

Shihan Wang
Utrecht University
Utrecht, the Netherlands
s.wang2@uu.nl

ABSTRACT

In multi-agent deep reinforcement learning (MADRL), agents can communicate with one another to perform a task in a coordinated manner. When multiple tasks are involved, agents can also leverage knowledge from one task to improve learning in other tasks. In this paper, we propose Multi-task Communication Skills (MCS), a MADRL with communication method that learns and performs multiple tasks simultaneously, with agents interacting through learnable communication protocols. Specifically, MCS adopts a task-invariant message space that enables communication under varying numbers of agents and tasks with different observation and action spaces. To enhance coordinated behaviors among agents, we further correlate the task-invariant message space to task-specific policies of agents. We adapt three existing multi-agent benchmark environments to multi-task settings. Empirical results demonstrate that MCS achieves better performance than multi-task MADRL baselines without communication, as well as single-task MADRL baselines with and without communication.

KEYWORDS

multi-task learning, multi-agent deep reinforcement learning, communication

ACM Reference Format:
Changxi Zhu, Mehdi Dastani, and Shihan Wang. 2026. Learning Communication Skills in Multi-task Multi-agent Deep Reinforcement Learning. In *Proc. of the Adaptive and Learning Agents Workshop (ALA 2026), Paphos, Cyprus, https://alaworkshop2026.github.io/, May 25 – 26, 2026*, IFAAMAS, 10 pages.

1 INTRODUCTION

Communication is essential in multi-agent deep reinforcement learning (MADRL) for sharing information and enhancing coordination among multiple agents toward common goals [43], especially in partially observable environments such as autonomous driving [29], sensor networks [24], and multi-robot control [13]. Recent works on learning communication in MADRL have investigated what information to communicate [11, 37], when and with whom to communicate [4, 30, 42] and how to integrate communication into policies [2, 5], thereby enabling flexible and dynamic communication protocols. However, these approaches are limited to single-task settings, where both policies and communication protocols are developed and evaluated for only one task.

A different line of work in the MADRL literature focuses on multi-task learning, which aims to learn policies across multiple tasks and is referred to as multi-task MADRL [23, 41]. This is achieved either via task-invariant architectures [8, 33] that exploit the shared structure across tasks to learn a single unified model, or via task representation learning [19, 26] that explicitly models environment dynamics to generalize to unseen tasks. However, despite these achievements, previous works on multi-task MADRL have not considered communication among agents within tasks. In contrast, humans can leverage knowledge, in particular communication knowledge, to learn multiple tasks simultaneously.

In this work, we extend the scope of MADRL with communication approaches from single-task to multi-task settings. Integrating communication into multi-task MADRL is non-trivial. In multi-task settings, tasks may differ in complexity, such as in the number of agents and in their observation and action spaces. As a result, communicated messages that are typically encoded from local information (e.g., observations and actions) may vary across tasks, causing communication overhead to grow with task complexity. To address these challenges, we propose Multi-task Communication Skills (MCS), a multi-task MADRL with communication method that leverages a task-invariant message space, where agents encode their local information into a unified message representation across tasks. Built upon the task-invariant message space, agents in MCS communicate with each other under varying task complexities while maintaining low communication overhead, enabling effective and efficient communication across diverse tasks.

To establish the task-invariant message space in MCS, communicated messages are generated, transmitted, and aggregated using task-invariant architectures. In particular, the varying dimensionalities of task-specific observations are first aligned and encoded into messages with the same dimensionality. During communication, unnecessary messages are pruned to reduce communication overhead. As the number of agents may vary across tasks, agents may receive different numbers of messages. We therefore employ a shared aggregation function that can process a variable number of received messages across tasks. The aggregated messages are subsequently used to attend to relevant features in both the policy and critic networks. To enhance coordination, we introduce a predictor that maximizes the mutual information between messages and agents' policies, while effectively handling varying action spaces across tasks. Consequently, the communication protocol in MCS becomes task-invariant, unifying what, when, and how to communicate across different tasks.

We evaluate MCS on the well-known StarCraftII Multi-Agent Challenge benchmark [27], a novel multi-task AliceBob environment, and an adapted multi-task version of Google Research Football [14]. We assess both the average learning performance across tasks and the per-task learning performance. We compare MCS

Proc. of the Adaptive and Learning Agents Workshop (ALA 2026), Aydeniz, Delgrange, Mohammedalamen, Yang (eds.), May 25 – 26, 2026, Paphos, Cyprus, https://alaworkshop2026.github.io/. 2026.

with multi-task MADRL baselines without communication to evaluate the effectiveness of communication. We also compare MCS with single-task MADRL baselines with and without communication, as single-task settings can be viewed as instances of multi-task settings. The results show that MCS outperforms both multi-task and single-task MADRL baselines, benefiting from its task-invariant communication protocol, which enables effective learning and performing in multiple tasks.

2 RELATED WORK

Multi-task Learning in MADRL. Early approaches in multi-task MADRL, such as DEC-HDRQN [23], distilled single-task Q-functions into a unified Q-function. More recent Multi-task MADRL methods focus either on developing task-invariant architectures [8, 9, 15, 33] or on learning task representations [19, 26, 28]. The task-invariant architectures exploit shared task structure by unifying inputs across tasks using entity-based representations of observations and actions [3]. Specifically, REFIL [9] partitions agents into subgroups to capture shared local coordination patterns across tasks. UPDet [8] leverages a single unified Transformer architecture to handle varying entities across tasks. DT2GS [33] decomposes each task into multiple sub-tasks via latent variables generated by Transformer encoders. RIT [15] applies mask techniques to selectively disable network layers corresponding to invalid entities. On the other hand, task representations can be learned from a set of tasks based on trajectories [28], transition and reward functions [26], or task similarity measures [19].

The above research works focus on online multi-task MADRL. A related line of research investigates offline multi-task MADRL which utilizes offline data across multiple independent tasks to learn generalized policies for unseen tasks [1, 17, 38]. Among them, ODIS [38] identifies coordination skills for each agent from states and joint actions in the offline data. HiSSD [17] adopts a hierarchical policy that jointly learns common and task-specific skills in an offline dataset. In contrast, we consider online multi-task MADRL without assuming access to offline data or state information. Moreover, we focus on communication, which has not been explored in the literature on either offline or online multi-task MADRL.

Communication in MADRL. Existing literature on communication in MADRL primarily focuses on the single-task setting. While our method is the first work on multi-task setting and utilizes a different communication mechanism compared to the existing literature, we get inspiration from single-task MADRL with communication from two main perspectives: what to communicate and when to communicate. First, the messages content (what to communicate) is typically encoded from either partial observations [5, 30, 31, 34, 42] or intended actions [11, 12, 37]. Specifically, TGCNet [42] leverages a Transformer to encode observations as messages. ATOC [11] encodes both observations and action intentions. IS [12] encodes imagined trajectories capturing agents' future action plans. MAIC [37] encodes agents' local histories and teammates' intentions into messages. The literature further adopts information-theoretic objectives to generate agent-specific communication. Specifically, NDQ [34] reduces the uncertainty for receiver agents by maximizing the mutual information between messages and the receivers' policies. Similarly, COCOM [16] encourages agents to send messages that reduce uncertainty in teammates' decisions, while assuming access to global state. Moreover, MAIC [37] maximizes the mutual information between the sender's teammate model and the receiver's policy to produce tailored messages. These approaches do not consider a task-invariant message space as in our method. Moreover, they require learning models of teammates or the environment, whereas our proposed method does not.

The decision of whether to communicate or not can be determined based on confidence or distance measures [6, 39, 40], a binary classifier [4, 20, 30, 32], or a communication graph [2, 7, 10, 18, 22, 42]. Specifically, agents can communicate when their confidence is low, as in VBC [39], or choose not to communicate when the previous or estimated messages are similar to the current messages, as in TMC [40] and MBC [6]. Methods based on learnable binary classifiers, including IC3Net [30], GACML [20], I2C [4], and T2MAC [32], assign communication labels using a threshold during training. Most similar to our work, attention mechanisms are often employed to construct communication graphs, such as in TarMAC [2], G2ANet [18], DGN [10], and MAGIC [22]. Recently, CommFormer [7] and TGCNet [42] prune unnecessary messages based on hard attention. In contrast, our method employs soft attention mechanisms but further integrates a threshold to prune unnecessary messages.

3 PRELIMINARIES

We extend the definition of multi-task MADRL [23] by incorporating entities and communication.

Definition 3.1. An Entity-Based Multi-Task Multi-Agent Reinforcement Learning with communication problem $\mathcal{MT}$ is defined as:

$$\mathcal{MT} := \{\mathcal{T}_k | k = 1, 2, ..., K\}$$

where each task $\mathcal{T}_k = \langle I, E, S, A, O, M, \Omega, P, R, \gamma \rangle$ is a Dec-POMDP tuple augmented with an additional message set M and entity set E. The Dec-POMDP components are a set of agents I, a set of environment states S, a set of joint actions A, a set of joint observations O, an observation function Ω, a transition function P, a reward function R, and a discount factor γ. We assume that agents are a subset of entities, i.e., $I \subseteq E$, and states are represented as an entity-based matrix, i.e., $S \subseteq \mathbb{R}^{|E| \times D^e}$, where D^e denotes the feature dimensionality, which remains the same for each entity in E. The observation function Ω maps the entity set to the set of observable entities, $\Omega : E \rightarrow \hat{E}$, which is used to construct observations $O \subseteq \mathbb{R}^{|\hat{E}| \times D^e}$ for $\hat{E} \subseteq E$. To enable a shared policy across tasks, we unify the varying sets of agents, entities, states, actions, observations, and messages by introducing the union sets of agents $\mathcal{I}$, entities $\mathcal{E}$, states $\mathcal{S}$, observations $\mathcal{O}$, and actions $\mathcal{A}$ across tasks. Since each individual task can have its own state space, we use $S^k \subseteq \mathcal{S}$ to denote the state space of task k. We follow the same conventional notation for all components of the Dec-POMDP of task k, where we have $I^k \subseteq \mathcal{I}$, $E^k \subseteq \mathcal{E}$, $\hat{E}^k \subseteq \mathcal{E}$, $S^k \subseteq \mathcal{S}$, $O^k \subseteq \mathcal{O}$, and $A^k \subseteq \mathcal{A}$. For communication, we assume a task-invariant message space $\mathcal{M}$ shared across tasks such that $M^1 = M^2 = ... = M^K = \mathcal{M}$.

In the partially observable environment of task k, agents may not observe the state $s^k \in S^k$ of the environment. Each agent i receives a local observation $o_i^k \in O^k$ and encodes it into a message

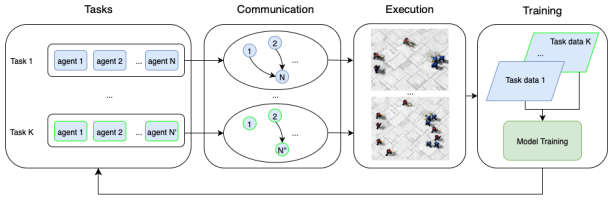


Figure 1: An overview of the MCS architecture. The experience from multiple tasks are used to train a shared model.

$m_i^k = f^{msg}(o_i^k; \theta_{enc}) \in \mathcal{M}$, where message function f^{msg} is parameterized by θ_{enc}. A joint policy parametrized by θ_π is then defined as $\pi(\boldsymbol{a}^k \mid \boldsymbol{o}^k, \boldsymbol{m}^k; \theta_\pi)$, where joint observations $\boldsymbol{o}^k = \langle o_i^k, ..., o_N^k \rangle$ and joint messages $\boldsymbol{m}^k = \langle m_1^k, ..., m_N^k \rangle$, with $N = |I^k|$ denoting the number of agents in task k. The goal of multi-task MADRL with communication is to learn a shared message encoder f^{msg} and policy π that jointly maximize the average return across K tasks:

$$\mathcal{L}_{\mathcal{MT}} = \max_{\theta_\pi, \theta_{enc}} \frac{1}{K} \sum_{k=1}^{K} \mathbb{E}_{s^k \sim p^k,\, \boldsymbol{a}^k \sim \pi} \left[\sum_{t=0}^{T} \gamma^t R^k(s_t^k, \boldsymbol{a}_t^k) \right]$$

where T is the length of episodes. Here, the policy π is conditioned on local observations and messages. We follow the centralized training and decentralized execution (CTDE) paradigm to use a centralized value function as the critic to guide the training of decentralized policies. In practice, the centralized value function is learned from joint observations and messages to estimate expected return. For task k, we define a centralized observation-based value function $V(\boldsymbol{o}^k, \boldsymbol{m}^k; \phi)$ with parameters ϕ as:

$$V(\boldsymbol{o}^k, \boldsymbol{m}^k; \phi) = \mathbb{E}_{s^k \sim p^k,\, \boldsymbol{a}^k \sim \pi} \left[\sum_{t=0}^{T} \gamma^t R^k(s_t^k, \boldsymbol{a}_t^k) \,\middle|\, \boldsymbol{o}^k, \boldsymbol{m}^k, \phi \right]$$

4 METHODS

In this section, we describe how agents in MCS generate, transmit, aggregate, and integrate messages based on task-invariant message space, while reducing communication overhead across tasks. An overview of the MCS architecture is shown in Figure 1. During training, experiences across all tasks are jointly collected to learn a shared communication protocol. During execution, the learned communication protocol generates different communication graphs according to task-specific observations. The entire framework, including the message encoder, policy network, and critic network, is trained end-to-end.

4.1 Entity-based Communication in Multi-task MADRL

The network architecture of MCS is schematically illustrated in Figure 2. Inspired by entity-based representations used in multi-task MADRL without communication [33], we first transform raw observations into entity-based representations $o_i^k \in \mathbb{R}^{|\hat{E}^k| \times D^e}$ for each agent i with observable entities $\hat{E}^k$ in each task k (as defined in Section 3). Without additional design, entity-based observations can increase communication overhead quadratically, as message dimensionality grows with the number of observed entities and the total amount of communication further grows with the number of communicating agents. To reduce communication overhead, we first apply mean pooling during message encoding to prevent the growth in message dimensionality. We further prune unnecessary messages using a communication mask. To accommodate a varying number of received messages (due to different numbers of agents across tasks), we adopt a task-invariant aggregation function that produces an invariant size of input for the policy and critic networks, enabling effective and efficient training across tasks.

Mean Pooling. Given entity-based observations, we employ a Transformer-based message encoder f^{msg}, shared by N agents and K tasks, to generate messages. The message encoder f^{msg} first embeds observations into an input embedding, followed by n standard Transformer blocks with multi-head attention. The multi-head attention outputs a hidden representation with shape $\mathbb{R}^{|\hat{E}^k| \times D^h}$ for each agent i in task k, where D^h is the hidden dimensionality. To handle the varying sizes of entities across tasks and obtain low-dimensional messages, we apply mean pooling to aggregate features over entities, followed by a fully connected layer. This produces a message for agent i in task k, denoted as $m_i^k = f^{msg}(o_i^k; \theta_{enc}) \in \mathbb{R}^{D^m}$, where D^m is the dimensionality of messages that remains constant across tasks. As a result, the message encoder f^{msg} maps task- and agent-specific observations into a shared message space $\mathbb{R}^{D^m}$, enabling the learned message representations to generalize across agents and tasks. We denote the joint messages in task k as $\boldsymbol{m}^k = \langle m_1^k, \ldots, m_N^k \rangle = \langle f^{msg}(o_1^k; \theta_{enc}), \ldots, f^{msg}(o_N^k; \theta_{enc}) \rangle$. This can be abbreviated as $\boldsymbol{m}^k = f^{msg}(\boldsymbol{o}^k; \theta_{enc})$.

Communication Mask. We further prune unnecessary messages while leveraging an attention mechanism to measure the importance of communication between agents. By introducing a communication threshold, we construct a communication mask that prevents redundant messages and scales the remaining ones according to their importance. In particular, inspired by Zhang et al. [42], we employ an additive attention mechanism to produce scores $\alpha_{i,j}^k \in [0, 1]$, which quantify the importance of agent i communicating with agent j in task k:

$$\alpha_{i,j}^k = \text{Gumbel-Softmax}\Big(v^\top \tanh\left(W_q m_i^k + W_k m_j^k\right)\Big), \tag{1}$$

where $v^\top$, W_q, and W_k are learnable parameters shared across agents and tasks. The communication mask $C_{i,j}^k \in [0, 1]$ is defined based on the scores [1]:

$$C_{i,j}^k = \begin{cases} \alpha_{i,j}^k, & \text{if } \alpha_{i,j}^k > \hat{\alpha} \\ 0, & \text{otherwise} \end{cases}, \quad i \neq j \tag{2}$$

where $\hat{\alpha} \in [0, 1]$ is a predefined threshold. We denote gated messages received by agent j from agent i for task k as $\tilde{m}_{i,j}^k = C_{i,j}^k\, m_i^k$. All messages received by agent j in task k are therefore denoted as $\tilde{\boldsymbol{m}}_j^k = \langle \tilde{m}_{1,j}^k, \ldots, \tilde{m}_{N,j}^k \rangle = \boldsymbol{C}_j^k \odot \boldsymbol{m}^k$, where $\boldsymbol{C}_j^k = \langle C_{1,j}^k, \ldots, C_{N,j}^k \rangle$ is the communication masks used by receiver agent j. In this way, the masks of redundant messages are set to 0 and therefore will not be considered in message integration.

[1]We use soft differentiable samples via the Gumbel-Softmax to generate scores $\alpha_{i,j}^k$.

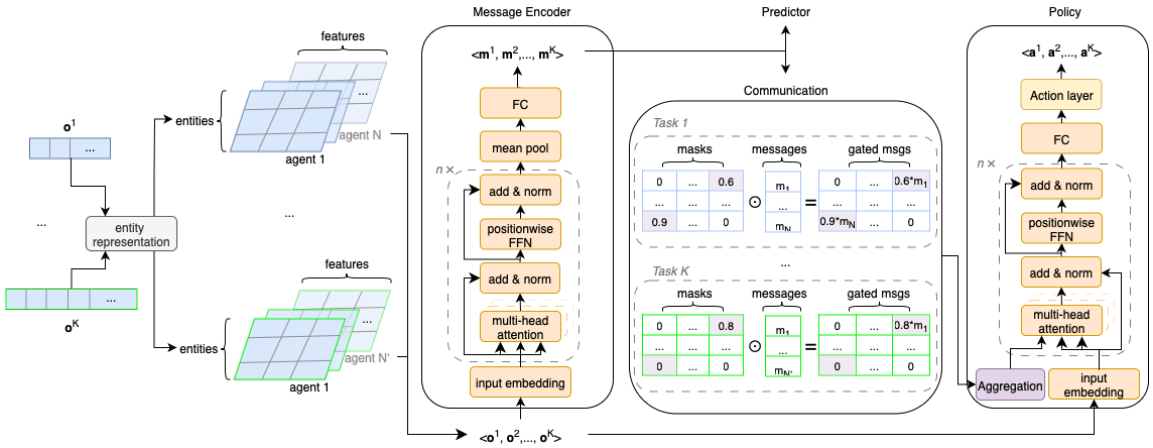


Figure 2: The network structure of MCS. Task-specific observations o^k are represented in an entity-based form and then encoded into messages $\boldsymbol{m}^k$. During communication, messages are pruned using masks C^k, applied through column-wise multiplication. Then, messages are aggregated and integrated into the policy network.

Task-invariant Aggregation. The number of received messages can vary across tasks, which changes the input size of the policy network. To address this, we use a task-invariant architecture capable of aggregating a variable number of received messages while allowing efficient gradient backpropagation. We thus employ a GRU as an aggregation function f^{agg}. The GRU produces fixed-dimensional representations regardless of the number of received messages across tasks. Moreover, the GRU scales linearly with the number of received messages, thereby providing computational efficiency. For each task k, the aggregation function produces an aggregated message $\bar{m}_j^k$ upon received messages $\tilde{\boldsymbol{m}}_j^k$ (i.e., $\bar{m}_j^k = f^{agg}(\tilde{\boldsymbol{m}}_j^k)$) as follows:

$$\begin{aligned} h_{j,\ell}^k &= \mathrm{GRU}(\tilde{m}_{\ell,j}^k,\ h_{j,\ell-1}^k), \qquad \ell = 1, \ldots, N \\ \bar{m}_j^k &= \frac{1}{N}\sum_{\ell} h_{j,\ell}^k \in R^{D^h}, \end{aligned} \tag{3}$$

where $h_{j,\ell}^k$ is the hidden state of receiver agent j in task k when processing the ℓ-th received message.

Message Integration. The aggregated message $\bar{m}_j^k$ is then used to help receiver agents focus on important and relevant features in task-specific observations when deciding an action. Specifically, we use $\bar{m}_j^k$ in the query embedding in a standard multi-head attention module within the Transformer-based policy network. For $\tilde{H}$ heads with $d_{\text{head}} = D^h/\tilde{H}$, each head $\tilde{h} \in \{1, \ldots, \tilde{H}\}$ computes:

$$\begin{aligned} \mu_j^{(\tilde{h})} &= \mathrm{softmax}\left(\frac{\mathcal{Q}_j^{(\tilde{h})}\mathcal{K}_j^{(\tilde{h})\top}}{\sqrt{d_{\text{head}}}}\right) \in \mathbb{R}^{1\times|\hat{E}^k|}, \\ z_j^k &= \left[\|_{\tilde{h}=1}^{\tilde{H}} \mu_j^{(\tilde{h})}\mathcal{V}_j^{(\tilde{h})}\right] W_z \ \in \ \mathbb{R}^{1\times D^h} \end{aligned} \tag{4}$$

where query embedding $\mathcal{Q}_j^{(\tilde{h})} = (\bar{m}_j^k)^\top W_{\mathcal{Q}}^{(\tilde{h})}$, key embedding $\mathcal{K}_j^{(\tilde{h})} = o_j^k W_{\mathcal{K}}^{(\tilde{h})}$, and value embedding $\mathcal{V}_j^{(\tilde{h})} = o_j^k W_{\mathcal{V}}^{(\tilde{h})}$. In Equation 4, $W_{\mathcal{Q}}$, $W_{\mathcal{K}}$, $W_{\mathcal{V}}$, and W_z are learnable weight matrices shared across agents and tasks. Notably, the query embedding $\mathcal{Q}j^{(\tilde{h})}$ and the key embedding $\mathcal{K}j^{(\tilde{h})}$ are used to produce weighting scores $\mu_j^{(\tilde{h})}$ over entities, which indicate the relevance of the received message to each observable entity of receiver j. These weighting scores are then used to combine the entity-based representations $\mathcal{V}_j^{(\tilde{h})}$, enabling receiver j to focus on the most relevant entities during communication. The outputs from all heads are concatenated to form z_j^k, which is used to decide the receiver's actions. Note that the final action layer follows Hu et al. [8] and Tian et al. [33].

In MCS, messages are used not only in the policy network but also as additional inputs to the critic networks. Then, each sender agent's messages are updated through gradient backpropagation from both the receiver agents' policy networks and the critic networks, thereby guiding the generation of messages that are most beneficial for communication. Within CTDE paradigm, we optimize the following objective across all agents and tasks:

$$\mathcal{L}(\theta_{enc}, \theta_\pi, \phi) = \frac{1}{K}\sum_{k}^{K}\frac{1}{N}\sum_{i}^{N}\mathbb{E}_{\boldsymbol{o}^k, a_i^k}[\log \pi(a_i^k|o_i^k, \bar{m}_i^k; \theta_\pi)V(\boldsymbol{o}^k, \bar{\boldsymbol{m}}^k; \phi)] \tag{5}$$

where $\bar{\boldsymbol{m}}^k = \langle \bar{m}_1^k, \ldots, \bar{m}_N^k \rangle$ denotes the joint aggregated messages of agents, and $\bar{m}_i^k = f^{agg}\Big(C_i^k \odot f^{msg}(\boldsymbol{o}^k; \theta_{enc})\Big)$. During centralized training, the encoder parameters θ_{enc}, policy parameters θ_π, and critic parameters ϕ are shared across agents and tasks.

4.2 Prediction Network for Coordination

With our proposed message encoder, communication is used to share encoded task-specific observations among agents, thereby broadening each agent's view of the environment. Despite this, sharing encoded observations does not explicitly account for coordination, as agents may still need to align their action selections to achieve coordinated behaviors. To address this, we further introduce a predictor q_{pred} that correlates the generated messages $\boldsymbol{m}^k$ with the sender agents' actions $\boldsymbol{a}^k$, thereby enhancing coordinated

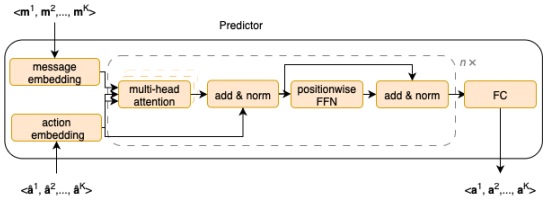


Figure 3: The network structure diagram of the predictor.

action selection among agents. Concretely, we employ an additional Transformer decoder as the predictor q_{pred} (Figure 3), which is used only during training and discarded at execution. Within multi-head attention, the predictor q_{pred} takes received messages as query embeddings. The key and value embeddings are derived from initialized actions denoted as $\tilde{\boldsymbol{a}}$ (e.g., null actions). To handle varying action dimensionalities across tasks, we apply zero-padding to actions. The predicted action distribution is then used for maximizing the mutual information $I(A^k; M^k)$ between the actions $\boldsymbol{a}^k \in A^k$ used in the environment and communicated messages $\boldsymbol{m}^k \in M^k$ for each task k. However, computing the mutual information $I(A^k; M^k)$ is intractable. Therefore, we maximize its variational lower bound (known as Barber-Agakov lower bound) [25]. By replacing the conditional distribution of actions given messages with a variational distribution $q_{pred}(\boldsymbol{a}^k \mid \boldsymbol{m}^k)$, we have:

$$\begin{aligned} I(A^k; M^k) &= \mathbb{E}_{p(\boldsymbol{a}^k, \boldsymbol{m}^k)}\left[\log \frac{p(\boldsymbol{a}^k \mid \boldsymbol{m}^k)}{p(\boldsymbol{a}^k)}\right] \\ &= \mathbb{E}_{p(\boldsymbol{a}^k, \boldsymbol{m}^k)}\left[\log \frac{q_{pred}(\boldsymbol{a}^k \mid \boldsymbol{m}^k)\, p(\boldsymbol{a}^k \mid \boldsymbol{m}^k)}{p(\boldsymbol{a}^k)\, q_{pred}(\boldsymbol{a}^k \mid \boldsymbol{m}^k)}\right] \\ &= \mathbb{E}_{p(\boldsymbol{a}^k, \boldsymbol{m}^k)}\left[\log \frac{q_{pred}(\boldsymbol{a}^k \mid \boldsymbol{m}^k)}{p(\boldsymbol{a}^k)}\right] + \\ &\mathbb{E}_{p(\boldsymbol{m}^k)}\Big[\mathrm{KL}\big(p(\boldsymbol{a}^k \mid \boldsymbol{m}^k) \,\|\, q_{pred}(\boldsymbol{a}^k \mid \boldsymbol{m}^k)\big)\Big] \\ &\geq \mathbb{E}_{p(\boldsymbol{a}^k, \boldsymbol{m}^k)}\big[\log q_{pred}(\boldsymbol{a}^k \mid \boldsymbol{m}^k)\big] \;+\; H(\boldsymbol{a}^k) \end{aligned} \tag{6}$$

where $H(\boldsymbol{a}^k) \equiv -\mathbb{E}_{p(\boldsymbol{a}^k)}[\log p(\boldsymbol{a}^k)]$ is the entropy of action distribution. The last inequality is due to the non-negativity of the KL divergence. Since the entropy term is non-negative, maximizing the first term is equal to maximizing the mutual information. In practice, we estimate $\mathbb{E}_{p(\boldsymbol{a}^k, \boldsymbol{m}^k)}\big[\log q_{pred}(\boldsymbol{a}^k \mid \boldsymbol{m}^k)\big]$ using sampled batch data. Given samples $\{(\boldsymbol{a}_b^k, \boldsymbol{m}_b^k, \boldsymbol{o}_b^k)\}_{k \in \{1,\ldots,K\}, b \in \{1,\ldots,B\}}$ for K tasks and B batch-size, we maximize the following log-likelihood:

$$\mathcal{L}_{pred}(\theta_{enc}) = \frac{1}{K}\frac{1}{B}\sum_{k=1}^{K}\sum_{b=1}^{B} \log q_{pred}\Big(\boldsymbol{a}_b^k \,\Big|\, f^{msg}(\boldsymbol{o}_b^k; \theta_{enc})\Big) \tag{7}$$

where message encoder f^{msg} first encodes sampled observations into messages. Then, the predictor q_{pred} estimates an action distribution based on the current messages. The loss $\mathcal{L}_{pred}$ is computed using the sampled actions and the estimated action distribution. Then, gradients are backpropagated from the predictor q_{pred} to the message encoder f^{msg}, encouraging f^{msg} to generate messages that reflect the sender agents' intended behaviors and align agents' action selections.

Algorithm 1 Multi-task Communication Skills

1: **Input**: Batch size B, number of tasks K, episodes L, steps T.
2: **Initialize**: Encoder parameters θ_{enc}, policy parameters θ_π, and critic parameters ϕ. Replay buffer $\mathcal{B}^k$ for each task k.
3: **for** $l = 0, 1, ..., L-1$ **do**
4: **for** $t = 0, 1, ..., T-1$ **do**
5: Collect observations $(\boldsymbol{o}_t^1, ..., \boldsymbol{o}_t^K)$ for K tasks
6: Generate messages $(\boldsymbol{m}_t^1, ..., \boldsymbol{m}_t^K)$ with message encoder
7: **for** $k = 1, ..., K$ **do**
8: Communicate and aggregate messages into $\bar{\boldsymbol{m}}_t^k$ based on Equations 1, 2, 3
9: **end for**
10: Decide actions $(\boldsymbol{a}_t^1, ..., \boldsymbol{a}_t^K)$ based on $(\bar{\boldsymbol{m}}_t^1, ..., \bar{\boldsymbol{m}}_t^K)$ in Equation 4 and collect the rewards $(r_t^1, ..., r_t^K)$
11: Insert $(\boldsymbol{o}_t^k, \boldsymbol{a}_t^k, r_t^k)$ into buffer $\mathcal{B}^k$
12: **end for**
13: **for** $k = 1, ..., K$ **do**
14: Sample a random minibatch of B steps from $\mathcal{B}^k$
15: Generate messages $\boldsymbol{m}^k = f^{msg}(\boldsymbol{o}^k)$
16: Generate predicted action distribution $q_{pred}(\boldsymbol{a}^k | \boldsymbol{m}^k)$
17: Generate values $V(o_1^k, \bar{m}_1^k), ..., V(o_N^k, \bar{m}_N^k)$
18: **end for**
19: Calculate loss $-\mathcal{L}_{\mathcal{MT}}$ based on Equation 8
20: Update critics, actors, and messages with gradient descent
21: **end for**

4.3 The Overall Learning Objective

The overall learning objective $\mathcal{L}_{\mathcal{MT}}$ of multi-task MADRL with communication is defined as:

$$\mathcal{L}_{\mathcal{MT}} = \mathcal{L}(\theta_{enc}, \theta_\pi, \phi) + \beta\, \mathcal{L}_{pred}(\theta_{enc}), \tag{8}$$

where β is a coefficient that balances the influence of the predictor. $\mathcal{L}_{\mathcal{MT}}$ ensures that communicated messages are both informative and beneficial for action selections of receiver agents, and that both the policies and critics are learned based on communication.

We further introduce Algorithm 1 to illustrate how agents learn the message encoder, policies, and critics in multi-task MADRL. At each time step t of an episode, agents in task k observe $\boldsymbol{o}_t^k = \langle o_{1,t}^k, \ldots, o_{N,t}^k \rangle$, which are used to generate messages $\boldsymbol{m}_t^k$ (lines 5-6). Based on the communication graph defined in Equations 1 and 2 and the aggregation function in Equation 3, messages are communicated and aggregated into $\bar{\boldsymbol{m}}_t^k$ at time step t (lines 7-9), which are then used to produce actions $\boldsymbol{a}_t^k$ (line 10). The actions $\boldsymbol{a}_t^k$ are executed in the environment, and rewards r_t^k are collected (line 10). The resulting transitions of observations, actions, and rewards are stored in task-specific replay buffers (line 11). During training, mini-batches are sampled from these buffers and the loss defined in Equation 8 is computed based on sampled experience (lines 13-19), Then, gradients are backpropagated through the critics, actors, predictors, aggregation function, and message encoders in an end-to-end manner (line 20).

5 EXPERIMENTS

We evaluate our proposed method MCS in three challenging multi-agent benchmark environments: AliceBob [36], StarCraftII Multi-Agent Challenge (SMAC) [27], and Google Research Football (GRF) [14] [2]. These environments were originally designed for single-task MADRL and consist of multiple cooperative tasks with challenges in coordinating agents' behaviors. Following Tian et al. [33], we construct multi-task settings for each environment. We then compare the following methods across these multiple tasks:

- **Multi-task MADRL methods with communication**. Our proposed method, MCS, is the first in this branch, which is a Transformer-based approach that allows communication among agents for better coordination in multiple tasks.
- **Multi-task MADRL methods without communication**. In this branch, DT2GS [33] and RIT [15] are two SOTA methods. DT2GS is a policy-based method that decompose tasks into subtasks under entity-based observations. RIT is a value-based method that uses a domain mask to filter out unobservable entities. We compare MCS with DT2GS and RIT to evaluate the benefit of using communication under multi-task settings, without relying on task decomposition or domain knowledge.
- **Single-task MADRL methods with communication**. In this branch, we consider TGCNet [42] and MAIC [37]. TGCNet is the SOTA method that employs a hard attention mechanism to generate a communication graph. In contrast to MCS, TGCNet does not explicitly account for coordination among agents. On the other hand, MAIC is a representative method that promotes coordination but requires learning a teammate model, whereas MCS does not. Notably, neither TGCNet nor MAIC considers the multi-task setting.
- **Single-task MADRL methods without communication**. MAT [35] is the SOTA method on many SMAC tasks, which employs a Transformer to encode observations for generating action sequences. HMASD [36] represents the SOTA method in leveraging latent skills for agent coordination. We compare MCS with MAT to evaluate the benefit of our proposed Transformer-based communication mechanism, and with HMASDA to assess the effectiveness of the predictor in promoting coordination.

The comparison examines two key aspects: (i) whether training is conducted in a multi-task or single-task setting, and (ii) whether communication is enabled among agents. This results in two solid perspectives for comparison: Multi-task with vs. without communication, and Multi-task vs. Single-task. By following the literature [33], we evaluate both the average win rate across the K tasks, defined as $Avg = \frac{1}{K}\sum_{k=1}^{K} WinRate(k)$, where $WinRate(k)$ denotes the win rate on task k, and the per-task win rate $WinRate(k)$ for each individual task. The win rate $WinRate(k)$ for each task is computed by running several evaluation episodes at fixed intervals during training. For comparison, we consider two aspects regarding the performance of MADRL methods: learning performance (final win rate) and learning efficiency (how fast it converges). The reported win rate are averaged over 6 random seeds for AliceBob and SMAC, and 4 random seeds for GRF. Moreover, we allocate sufficient computational budget (1/2 A100 GPU) to each method for each seed. The hyperparameters for each environment are adopted either from published sources or obtained through fine-tuning, provided in the Appendix. For all environments, we set the coefficient $\beta = 0.1$ (introduced in Equation 8) for MCS. Regarding the threshold $\hat{\alpha}$ (introduced in Equation 2), we use $\hat{\alpha} = 0.5$ for all environments except SMAC `Marine`, where we set $\hat{\alpha} = 0.9$. We further analyze the impact of β and $\hat{\alpha}$ on learning performance in Section 5.2. Detailed implementation of the entity-based representations are provided in the Appendix. We also report the computational time of all methods in the Appendix, showing that the proposed MCS introduces no significant additional runtime.

[2]The source code is available at https://github.com/changxizhu/MCS.

5.1 Evaluation Domains

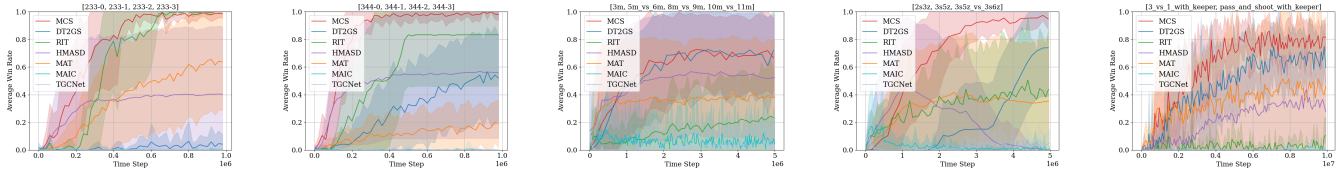
The AliceBob environment was originally created to show how agents coordinate to collect diamonds in a grid world, where a diamond can only be collected if another agent is simultaneously standing on the button of the same color. We extend this environment to a multi-task setting where each task consists of either diamonds or food, and either buttons or keys. For example, task 0 may involve 3 pairs of entities $\{red\ diamond,\ red\ button\}$, $\{blue\ diamond,\ blue\ button\}$, $\{pink\ diamond,\ pink\ button\}$ with 2 agents $\{Alice, Bob\}$. We denote this configuration as 233-0, indicating 2 agents, 3 diamonds/foods, and 3 buttons/keys with task index 0. By varying entity types and associated colors (therefore different IDs), we obtain the `AliceBob-233` series: {233-0, 233−1, 233-2, 233−4}. To examine scalability, we further construct the `AliceBob-344` series {344−0, 344−1, 344−2, 344-4} by introducing an additional agent (Charlie) as well as an additional color and entity (e.g., blue flower). To increase task complexity, entities including agents are randomly placed in the environment at the beginning of each episode, requiring effective coordination for the agents to reach the correct and changing targets.

In SMAC, RL agents must defeat enemies, requiring effective cooperation strategies. We construct two series of tasks: `Marine` series $\{3m, 5m_vs_6m, 8m_vs_9m, 10m_vs_11m\}$ and `Stalker-Zealot` series $\{2s3z, 3s5z, 3s5z_vs_3s6z\}$, following the multi-task MADRL literature [33]. In these series, the number of agents, as well as the observation and action spaces, vary across tasks, increasing the difficulty of learning a shared communication protocol.

GRF is challenging because of high stochasticity and sparse rewards. In GRF, opponents are controlled by expert agents, which significantly increase the complexity of the state−action space. GRF has not been used in multi-task MADRL literature, even without communication, and we extend the single-task environment to multi-task settings. To ensure tasks are comparable and jointly learnable, we select two maps requiring shooting strategies to show that MCS can solve multiple maps in GRF. Then, we construct a `Football` series: $\{3_vs_1_with_keeper,\ pass_and_shoot_with_keeper\}$, which involve different numbers of RL agents attempting to score from the edge of the field.

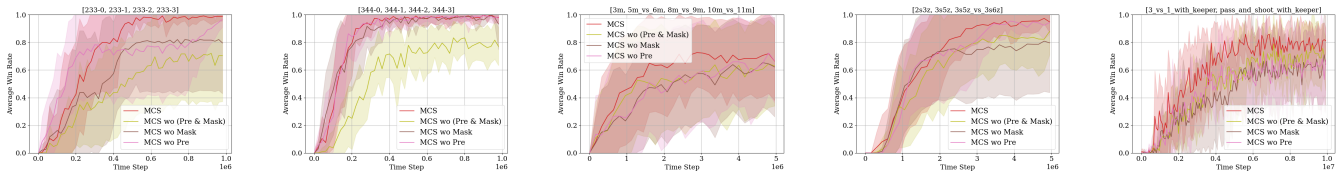
5.2 Experimental Results

Win Rates of Tasks. We report the average win rate across tasks for both multi-task and single-task MADRL methods in Figure 4. As shown, MCS achieves much higher average win rates than other methods in `AliceBob-344`, `Stalker-Zealot`, and `Football`



(a) AliceBob-233 (b) AliceBob-344 (c) Marine (d) Stalker–Zealot (e) Football

Figure 4: Average win-rate across multiple tasks on AliceBob (a–b), SMAC (c–d), and GRF (e).



(a) AliceBob-233 (b) AliceBob-344 (c) Marine (d) Stalker-Zealot (e) Football

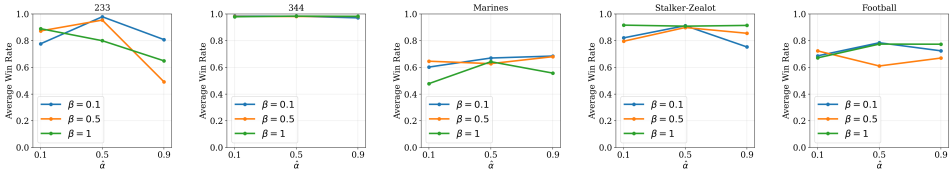
Figure 5: Ablations of MCS on AliceBob (a–b), SMAC (c–d), and GRF (e).

series. In `AliceBob-233`, MCS outperforms all the other methods except RIT. Note that RIT employs a domain mask that encodes entities from all tasks. This domain knowledge enables RIT to match the final learning performance of MCS in `AliceBob-233`. However, MCS, which does not rely on domain knowledge, achieves faster learning during the early training stage (before ~0.6M steps). As the number of agents increases, RIT's domain mask scales quadratically, increasing learning complexity, which explains why MCS outperforms RIT in `AliceBob-344`. In `Marine`, MCS achieves a similar average win rate to DT2GS. Compared to single-task MADRL with and without communication methods, MCS consistently achieves higher average win rates in all series. In particular, MADRL communication methods such as TGCNet and MAIC rely on training with a single batch, which prevents them from efficiently exploiting successful episodes, e.g., when agents must collect all diamonds or food for a win. In `Stalker-Zealot`, HMASD exhibits a significant performance drop at around 1M timesteps, suggesting convergence to a poor local optimum. Per-task win rate for both multi-task and single-task methods are provided in the Appendix. Notably, across all tasks in `AliceBob-233`, `AliceBob-344`, `Stalker-Zealot`, and `Football`, MCS consistently achieves comparable or higher win rates compared to other methods. In `Marine` series, DT2GS outperforms MCS in $10m_vs_11m$, where DT2GS can decompose the task into several sub-tasks and reduce learning complexity. Despite this, MCS achieves similar or even higher win rates than DT2GS in other tasks in `Marine`. The learning performance across all series demonstrates that MCS benefits from learning a shared communication protocol, enabling it to effectively learn and perform multiple tasks simultaneously.

Ablations. We conduct ablation experiments on MCS to assess the contributions of the communication mask (Equation 2) and the predictor (Equation 7), investigating whether pruning unnecessary messages and encouraging the correlation between messages and actions improves learning performance. As shown in Figure 5, MCS achieves higher average win rates compared to MCS without the predictor (Pre) and the communication mask (Mask), denoted as MCS (Pre & Mask), across all environments. Removing the communication mask in MCS (i.e., MCS wo Mask) decreases the learning performance in all series except for `AliceBob-344`. In `AliceBob-344`, MCS wo Mask achieves a similar final average win rate to MCS, whereas MCS exhibits faster convergence. On the other hand, removing the predictor (i.e., MCS wo Pre) will slow down the convergence (compared with MCS) in `AliceBob-233`, `Marine`, and `Stalker-Zealot` series. In `AliceBob-344` series, MCS wo Pre achieves a similar converged performance as MCS. In `Football`, however, removing the predictor in MCS decreases the learning performance. The results demonstrate that the combination of the communication mask and the predictor in MCS can accelerate or improve the learning performance.

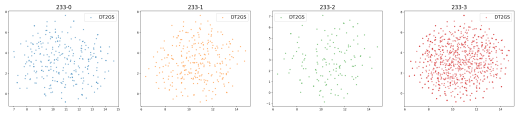
We also investigate alternative message aggregation mechanisms beyond the GRU used in MCS, including mean aggregation and Transformer-based aggregation, as well as different sampling strategies for training the GRU: a fixed order of sampled agents and a random order of sampled agents (used in MCS). Both alternative aggregation methods (i.e., Mean and Transformer) exhibit performance degradation compared to GRU on `AliceBob-233` and `Stalker-Zealot`. Moreover, the performance gap between the two sampling strategies becomes marginal when the number of agents exceeds two, as the communication topology plays a more critical role. Detailed ablation results are provided in the Appendix.

Hyperparameters. We further analyze the effects of the threshold $\hat{\alpha}$ and the coefficient β. Specifically, we evaluate $\hat{\alpha} \in \{0.1, 0.5, 0.9\}$ and $\beta \in \{0.1, 0.5, 1\}$. A larger threshold $\hat{\alpha}$ corresponds to lower communication overhead, while a larger coefficient β emphasizes more on regularizing communicated messages. Figure 6 reports the average win rate over the last 20 episodes under different combinations of $\hat{\alpha}$ and β. As shown, in `AliceBob-344`, all combinations of $\hat{\alpha}$ and β achieve a similar learning performance, demonstrating robustness

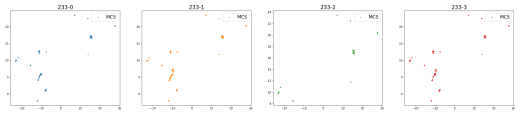


(a) AliceBob-233 (b) AliceBob-344 (c) Marine (d) Stalker-Zealot (e) Football

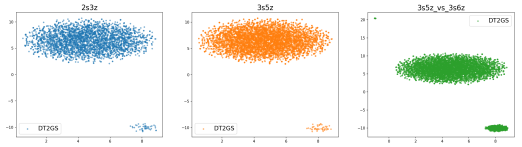
Figure 6: Average win rate for different combinations of $\hat{\alpha}$ and β achieved by MCS in all environments.



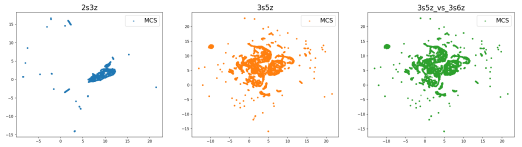
(a) DT2GS in AliceBob



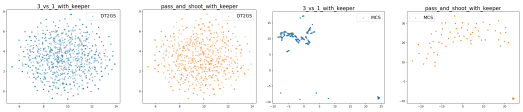
(b) MCS in AliceBob



(c) DT2GS in SMAC



(d) MCS in SMAC



(e) DT2GS in GRF

(f) MCS in GRF

Figure 7: Latent representations of DT2GS and MCS.

to these hyperparameters. For `AliceBob-233`, `Stalker-Zealot`, and `Football`, a larger threshold ($\hat{\alpha} = 0.9$) can degrade the learning performance, indicating that these environments benefit less from overly limited communication. In contrast, in `Marine`, MCS tends to perform better with larger values of $\hat{\alpha}$, particularly when $\beta = 0.1$ or $\beta = 0.5$, where communication overhead is much reduced while being less regularized. Overall, a robust configuration is $\hat{\alpha} = 0.5$ and $\beta = 0.1$, suggesting that a moderate communication level combined with mild regularization yields consistently high average win rate.

Analysis of Message Representations. We further investigate communicated messages learned by MCS, which correspond to the latent representations generated by our proposed Transformer-based message encoder. To examine the impact of communication on learning, we compare the latent representations produced by MCS with DT2GS, which also employs a Transformer to encode observations for policy learning but does not incorporate communication. For visualization, we record the latent representations from the last three episodes of MCS and DT2GS. We then apply UMAP [21] to project the high-dimensional latent representations into a two dimensional space, as shown in Figure 7, where each point corresponds to an agent. In AliceBob, where different tasks differ only in the type of entities, both DT2GS and MCS learn similar latent representations across tasks. However, MCS produces more compact message representations, indicating a shared communication pattern among agents, which enhances their coordination. In SMAC, DT2GS tends to learn similar representations across all tasks, while task $2c3s$ should intuitively differ from the others due to the smaller number of agents and enemies involved. In contrast, MCS can capture this difference and learns a distinct representation for $2c3s$, likely because fewer agents participate in communication. In `Football`, MCS also distinguishes between $3_vs_1_with_keeper$ and $pass_and_shoot_with_keeper$, indicating that agents may adopt different shooting strategies as the number of agents and their starting positions vary across the two tasks. Furthermore, in all tasks, DT2GS tends to spread the representations in the latent space, treating agents independently without capturing their interactions. In contrast, MCS forms clusters that reflect structured representations, indicating meaningful inter-agent dependencies, which is consistent with its superior empirical performance.

6 CONCLUSION

We propose Multi-task Communication Skills (MCS), a multi-task MADRL with communication method that learns a shared communication protocol across tasks with varying numbers of agents, observation spaces, and action spaces. We introduce a predictor that maximizes the mutual information between messages and actions to promote coordination. Empirical results show that MCS outperforms multi-task MADRL methods without communication and single-task MADRL methods with or without communication across several benchmark multi-task environments. Moreover, MCS exhibits meaningful patterns in the latent message representations. In future work, we aim to develop more adaptive strategies that dynamically prune unnecessary messages for each task. We will also investigate how the differences across tasks influence the learned message representation.

REFERENCES

[1] Jiayu Chen, Tian Lan, and Vaneet Aggarwal. 2024. Variational offline multi-agent skill discovery. *arXiv preprint arXiv:2405.16386* (2024).

[2] Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. 2019. TarMAC: Targeted Multi-Agent Communication. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*. 1538–1546.

[3] Christian Schröder de Witt, Jakob N. Foerster, Gregory Farquhar, Philip H. S. Torr, Wendelin Boehmer, and Shimon Whiteson. 2019. Multi-Agent Common Knowledge Reinforcement Learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 9924–9935.

[4] Ziluo Ding, Tiejun Huang, and Zongqing Lu. 2020. Learning Individually Inferred Communication for Multi-Agent Cooperation. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.).

[5] Cong Guan, Feng Chen, Lei Yuan, Chenghe Wang, Hao Yin, Zongzhang Zhang, and Yang Yu. 2022. Efficient Multi-agent Communication via Self-supervised Information Aggregation. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (Eds.).

[6] Shuai Han, Mehdi Dastani, and Shihan Wang. 2023. Model-based Sparse Communication in Multi-agent Reinforcement Learning. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023, London, United Kingdom, 29 May 2023 - 2 June 2023*, Noa Agmon, Bo An, Alessandro Ricci, and William Yeoh (Eds.). ACM, 439–447.

[7] Shengchao Hu, Li Shen, Ya Zhang, and Dacheng Tao. 2024. Learning Multi-Agent Communication from Graph Modeling Perspective. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

[8] Siyi Hu, Fengda Zhu, Xiaojun Chang, and Xiaodan Liang. 2021. UPDeT: Universal Multi-agent RL via Policy Decoupling with Transformers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

[9] Shariq Iqbal, Christian A Schroeder De Witt, Bei Peng, Wendelin Böhmer, Shimon Whiteson, and Fei Sha. 2021. Randomized entity-wise factorization for multi-agent reinforcement learning. In *International Conference on Machine Learning*. PMLR, 4596–4606.

[10] Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu. 2020. Graph Convolutional Reinforcement Learning. In *8th International Conference on Learning Representations (ICLR)*.

[11] Jiechuan Jiang and Zongqing Lu. 2018. Learning Attentional Communication for Multi-Agent Cooperation. In *Advances in Neural Information Processing Systems 31 (NIPS)*. 7265–7275.

[12] Woojun Kim, Jongeui Park, and Youngchul Sung. 2021. Communication in Multi-Agent Reinforcement Learning: Intention Sharing. In *9th International Conference on Learning Representations (ICLR)*.

[13] Jens Kober, J. Andrew Bagnell, and Jan Peters. 2013. Reinforcement learning in robotics: A survey. *Int. J. Robotics Res.* 32, 11 (2013), 1238–1274.

[14] Karol Kurach, Anton Raichuk, Piotr Stanczyk, Michal Zajac, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, and Sylvain Gelly. 2020. Google Research Football: A Novel Reinforcement Learning Environment. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*. 4501–4510.

[15] Chao Li, Shaokang Dong, Shangdong Yang, Yujing Hu, Tianyu Ding, Wenbin Li, and Yang Gao. 2025. Multi-Task Multi-Agent Reinforcement Learning With Interaction and Task Representations. *IEEE Trans. Neural Networks Learn. Syst.* 36, 7 (2025), 13431–13445.

[16] Dapeng Li, Na Lou, Zhiwei Xu, Bin Zhang, and Guoliang Fan. 2025. Efficient Communication in Multi-Agent Reinforcement Learning with Implicit Consensus Generation. In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, Toby Walsh, Julie Shah, and Zico Kolter (Eds.). AAAI Press, 23240–23248.

[17] Sicong Liu, Yang Shu, Chenjuan Guo, and Bin Yang. 2025. Learning Generalizable Skills from Offline Multi-Task Data for Multi-Agent Cooperation. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

[18] Yong Liu, Weixun Wang, Yujing Hu, Jianye Hao, Xingguo Chen, and Yang Gao. 2020. Multi-Agent Game Abstraction via Graph Attention Neural Network. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*. 7211–7218.

[19] Mridul Mahajan, Georgios Tzannetos, Goran Radanovic, and Adish Singla. 2024. Learning Embeddings for Sequential Tasks Using Population of Agents. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*. ijcai.org, 4733–4741.

[20] Hangyu Mao, Zhengchao Zhang, Zhen Xiao, Zhibo Gong, and Yan Ni. 2020. Learning agent communication under limited bandwidth by message pruning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 5142–5149.

[21] Leland McInnes and John Healy. 2018. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *CoRR* abs/1802.03426 (2018).

[22] Yaru Niu, Rohan R. Paleja, and Matthew C. Gombolay. 2021. Multi-Agent Graph-Attention Communication and Teaming. In *20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 964–973.

[23] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P. How, and John Vian. 2017. Deep Decentralized Multi-task Multi-Agent Reinforcement Learning under Partial Observability. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017 (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 2681–2690.

[24] Manisa Pipattanasomporn, Hassan Feroze, and Saifur Rahman. 2009. Multi-agent systems in a distributed smart grid: Design and implementation. In *2009 IEEE/PES Power Systems Conference and Exposition*. IEEE, 1–8.

[25] Ben Poole, Sherjil Ozair, Aäron van den Oord, Alexander A. Alemi, and George Tucker. 2019. On Variational Bounds of Mutual Information. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 5171–5180.

[26] Rongjun Qin, Feng Chen, Tonghan Wang, Lei Yuan, Xiaoran Wu, Yipeng Kang, Zongzhang Zhang, Chongjie Zhang, and Yang Yu. 2024. Multi-agent policy transfer via task relationship modeling. *Sci. China Inf. Sci.* 67, 8 (2024).

[27] Mikayel Samvelyan, Tabish Rashid, Christian Schröder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob N. Foerster, and Shimon Whiteson. 2019. The StarCraft Multi-Agent Challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019*, Edith Elkind, Manuela Veloso, Noa Agmon, and Matthew E. Taylor (Eds.). 2186–2188.

[28] Lukas Schafer, Filippos Christianos, Amos Storkey, et al. 2023. Learning Task Embeddings for Teamwork Adaptation in Multi-agent Reinforcement Learning [C/OL]. In *NeurIPS 2023 Workshop on Generalization in Planning*.

[29] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. 2016. Safe, Multi-Agent, Reinforcement Learning for Autonomous Driving. *CoRR* abs/1610.03295 (2016). arXiv:1610.03295 http://arxiv.org/abs/1610.03295

[30] Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. 2019. 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*.

[31] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. 2016. Learning Multiagent Communication with Backpropagation. In *Advances in Neural Information Processing Systems 29 (NIPS)*. 2244–2252.

[32] Chuxiong Sun, Zehua Zang, Jiabao Li, Jiangmeng Li, Xiao Xu, Rui Wang, and Changwen Zheng. 2024. T2MAC: Targeted and Trusted Multi-Agent Communication through Selective Engagement and Evidence-Driven Integration. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (Eds.). AAAI Press, 15154–15163.

[33] Zikang Tian, Ruizhi Chen, Xing Hu, Ling Li, Rui Zhang, Fan Wu, Shaohui Peng, Jiaming Guo, Zidong Du, Qi Guo, and Yunji Chen. 2023. Decompose a Task into Generalizable Subtasks in Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (Eds.).

[34] Tonghan Wang, Jianhao Wang, Chongyi Zheng, and Chongjie Zhang. 2020. Learning Nearly Decomposable Value Functions Via Communication Minimization. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

[35] Muning Wen, Jakub Grudzien Kuba, Runji Lin, Weinan Zhang, Ying Wen, Jun Wang, and Yaodong Yang. 2022. Multi-Agent Reinforcement Learning is a Sequence Modeling Problem. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (Eds.).

[36] Mingyu Yang, Yaodong Yang, Zhenbo Lu, Wengang Zhou, and Houqiang Li. 2023. Hierarchical Multi-Agent Skill Discovery. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (Eds.).

[37] Lei Yuan, Jianhao Wang, Fuxiang Zhang, Chenghe Wang, Zongzhang Zhang, Yang Yu, and Chongjie Zhang. 2022. Multi-Agent Incentive Communication via Decentralized Teammate Modeling. In *Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI-22)*.

[38] Fuxiang Zhang, Chengxing Jia, Yi-Chen Li, Lei Yuan, Yang Yu, and Zongzhang Zhang. 2023. Discovering Generalizable Multi-agent Coordination Skills from Multi-task Offline Data. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

[39] Sai Qian Zhang, Qi Zhang, and Jieyu Lin. 2019. Efficient Communication in Multi-Agent Reinforcement Learning via Variance Based Control. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 3230–3239.

[40] Sai Qian Zhang, Qi Zhang, and Jieyu Lin. 2020. Succinct and Robust Multi-Agent Communication With Temporal Message Control. In *Advances in Neural Information Processing Systems 33 (NIPS)*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.).

[41] Yu Zhang and Qiang Yang. 2022. A Survey on Multi-Task Learning. *IEEE Trans. Knowl. Data Eng.* 34, 12 (2022), 5586–5609.

[42] Zhuohui Zhang, Bin He, Bin Cheng, and Gang Li. 2025. Bridging Training and Execution via Dynamic Directed Graph-Based Communication in Cooperative Multi-Agent Systems. In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, Toby Walsh, Julie Shah, and Zico Kolter (Eds.). AAAI Press, 23395–23403.

[43] Changxi Zhu, Mehdi Dastani, and Shihan Wang. 2024. A survey of multi-agent deep reinforcement learning with communication. *Autonomous Agents and Multi-Agent Systems* 38, 4 (2024).