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

Changxi Zhu, Mehdi Dastani, Shihan Wang

Keywords: Multi-task learning, Multi-agent deep reinforcement learning, Communication

Summary

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 method with communication that learns and performs multiple tasks simultaneously, with agents interacting through learnable communication protocols. Specifically, MCS adopts a task-invariant communication protocol 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 communicated messages to agents' policies for each task. 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.

Contribution(s)

1. We extend the scope of MADRL approaches with communication from single-task to multi-task settings. We therefore propose the first multi-task MADRL method with communication, MCS, aiming to learn and perform multiple tasks simultaneously with communicating agents.
   Context: The literature on multi-task MADRL does not consider communication, while the literature considering communication in MADRL focuses only on single-task settings.
2. We propose to use a task-invariant communication protocol that is shared across multiple tasks. We further enhance coordination among agents by maximizing the mutual information between communicated messages and agents' policies for each specific task, enabling informative messages during communication.
   Context: Prior work on single-task MADRL with communication does not employ a task-invariant communication protocol, resulting in communication overhead that increases with task complexity. Meanwhile, the multi-task MADRL literature adopts task-invariant architectures to enable a shared policy across tasks but do not address complex communication among agents.
3. We evaluate MCS in three challenging multi-agent benchmark environments, SMAC, AliceBob, and Google Research Football (GRF) in 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.
   Context: We adopt the same multi-task settings on SMAC as Tian et al. (2023). AliceBob and GRF are evaluated only in single-task settings in the literature, and we extend them to multi-task settings following a similar extension as in SMAC. Following Tian et al. (2023) and Li et al. (2025a), we evaluate methods in terms of both average win rate across tasks and per-task win rate. We compare MCS with single-task MADRL baselines since single-task settings can be viewed as instances of multi-task settings.

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

Changxi Zhu[1], **Mehdi Dastani**[1], **Shihan Wang**[1]
{c.zhu,m.m.dastani,s.wang2}@uu.nl

[1]**Department of Information and Computing Sciences, Utrecht University, The Netherlands**

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 method with communication that learns and performs multiple tasks simultaneously, with agents interacting through learnable communication protocols. Specifically, MCS adopts a task-invariant communication protocol 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 communicated messages to agents' policies for each task. 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.

1 Introduction

Communication is essential in multi-agent deep reinforcement learning (MADRL) for sharing information and enhancing coordination among multiple agents toward common goals (Zhu et al., 2024), especially in partially observable environments such as autonomous driving (Shalev-Shwartz et al., 2016), sensor networks (Pipattanasomporn et al., 2009), and multi-robot control (Kober et al., 2013). Recent works on learning communication in MADRL have investigated what information to communicate (Jiang & Lu, 2018; Yuan et al., 2022), when and with whom to communicate (Singh et al., 2019; Ding et al., 2020; Zhang et al., 2025) and how to integrate communication into policies (Das et al., 2019; Guan et al., 2022), thereby enabling learnable 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 (Zhang & Yang, 2022; Omidshafiei et al., 2017). This is achieved either via task-invariant architectures that exploit the shared structure across tasks to learn a single unified model (Hu et al., 2021; Tian et al., 2023), or via task representation learning that models environment dynamics to generalize to unseen tasks (Qin et al., 2024; Mahajan et al., 2024). However, despite these achievements, previous works on multi-task MADRL has not considered communication among agents within tasks. Instead, we aim at utilizing communication knowledge when learning multiple tasks simultaneously.

In this work, we extend the scope of MADRL approaches with communication 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 method with communication that leverages a task-invariant communication protocol, which unifies what, when, and how to communicate across different tasks.[1] In MCS, agents communicate with each other under varying task complexities while maintaining low communication overhead, enabling effective and efficient communication in multiple tasks.

The task-invariant communication protocol in MCS consists of the following processes, where messages are generated, transmitted, and aggregated using task-invariant architectures. First, the task-specific observations that vary across tasks are aligned and encoded into messages. Second, during communication, unnecessary messages are pruned to reduce communication overhead. Third, an aggregation function is employed to combine a variable number of received messages across tasks. Finally, the aggregated messages attend to relevant features in both the policy and critic networks, facilitating the learning of optimal agent behaviors. To enhance coordination among agents, we further introduce a predictor that maximizes the mutual information between communicated messages and agents' policies, while effectively handling varying action spaces across tasks. Then, agents in MCS can communicate and perform in multiple tasks with varying numbers of agents and different observation and action spaces.

We evaluate MCS on the well-known StarCraftII Multi-Agent Challenge (SMAC) benchmark (Samvelyan et al., 2019), a novel multi-task AliceBob environment, and an adapted multi-task version of Google Research Football (GRF) (Kurach et al., 2020). We assess both the average learning performance across tasks and the per-task learning performance. We compare MCS 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 (Omidshafiei et al., 2017), distilled single-task Q-functions into a unified Q-function. More recent Multi-task MADRL methods focus either on developing task-invariant architectures (Iqbal et al., 2021; Hu et al., 2021; Tian et al., 2023; Li et al., 2025a) or on learning task representations (Schäfer et al., 2023; Qin et al., 2024; Mahajan et al., 2024). The task-invariant architectures exploit shared task structure by unifying inputs across tasks using entity-based representations of observations and actions (de Witt et al., 2019). Specifically, REFIL (Iqbal et al., 2021) partitions agents into subgroups to capture shared local coordination patterns across tasks. UPDet (Hu et al., 2021) leverages a single unified Transformer architecture to handle varying entities across tasks. DT2GS (Tian et al., 2023) decomposes each task into multiple sub-tasks via latent variables generated by Transformer encoders. RIT (Li et al., 2025a) 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 (Schäfer et al., 2023), transition and reward functions (Qin et al., 2024), or task similarity measures (Mahajan et al., 2024).

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 tasks to learn generalized policies for unseen tasks (Zhang et al., 2023; Chen et al., 2024; Liu et al., 2025). Among them, ODIS (Zhang et al., 2023) identifies coordination skills for each agent from states and joint actions in the offline data. HiSSD (Liu et al., 2025) adopts a hierarchical policy that jointly learns common and task-specific skills in an offline dataset. In contrast, we consider online multi-task MADRL.

[1]The literature (Tian et al., 2023; Qin et al., 2024) uses related notions such as population-invariant and input-length-invariant. We use task-invariant communication protocol to emphasize invariance to both the number of agents and observation/action spaces.

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 focuses only on the single-task setting. While we focus on multi-task setting and employ a different communication protocol from those used in single-task setting, we draw inspiration from the single-task MADRL with communication literature 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 (Sukhbaatar et al., 2016; Singh et al., 2019; Wang et al., 2020; Guan et al., 2022; Zhang et al., 2025) or intended actions (Jiang & Lu, 2018; Kim et al., 2021; Yuan et al., 2022). Specifically, TGCNet (Zhang et al., 2025) leverages a Transformer to encode observations as messages. ATOC (Jiang & Lu, 2018) encodes both observations and action intentions. IS (Kim et al., 2021) encodes imagined trajectories capturing agents' future action plans. MAIC (Yuan et al., 2022) encodes agents' local histories and teammates' intentions into messages. The literature further adopts information-theoretic objectives to generate agent-specific message content. Specifically, NDQ (Wang et al., 2020) maximizes the mutual information between communicated messages and the receivers' policies for reducing the receivers' uncertainty. Similarly, COCOM (Li et al., 2025b) encourages agents to send messages that reduce uncertainty in teammates' decisions, while assuming access to global state. Moreover, MAIC (Yuan et al., 2022) 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 communication protocol as in our method. Moreover, they learn 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 (Zhang et al., 2019; 2020; Han et al., 2023), a binary classifier (Singh et al., 2019; Mao et al., 2020; Ding et al., 2020; Sun et al., 2024), or a communication graph (Das et al., 2019; Liu et al., 2020; Jiang et al., 2020; Niu et al., 2021; Hu et al., 2024; Zhang et al., 2025; Lee et al., 2025). Specifically, agents can communicate when their confidence is low, as in VBC (Zhang et al., 2019), or choose not to communicate when the previous or estimated messages are similar to the current messages, as in TMC (Zhang et al., 2020) and MBC (Han et al., 2023). Methods based on learnable binary classifiers, including IC3Net (Singh et al., 2019), GACML (Mao et al., 2020), I2C (Ding et al., 2020), and T2MAC (Sun et al., 2024), 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 (Das et al., 2019), G2ANet (Liu et al., 2020), DGN (Jiang et al., 2020), and MAGIC (Niu et al., 2021). More recently, CommFormer (Hu et al., 2024) and TGCNet (Zhang et al., 2025) prune messages based on hard attention mechanisms, while IWoL (Lee et al., 2025) restricts communication to agents within a certain distance. In contrast, our method employs soft attention mechanisms but integrates a threshold to prune unnecessary messages among agents.

3 Preliminaries

We extend the definition of multi-task MADRL (Omidshafiei et al., 2017) 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

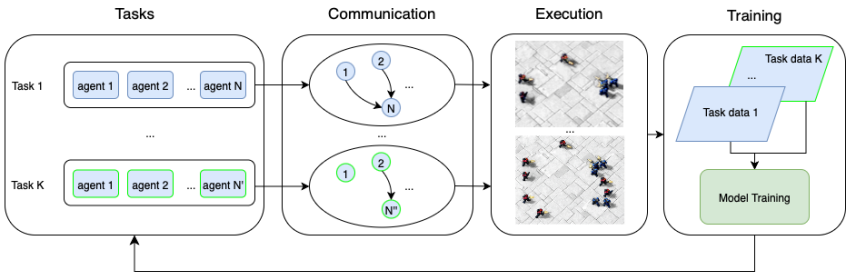


Figure 1: An overview of the MCS framework. Agents communicate and act in each task, and experience from multiple tasks are used to train a shared communication protocol across tasks.

in E. The observation function Ω maps the entity set to the set of observable entities, $\Omega : E \to \hat{E}$, 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 the other components of the Dec-POMDP of task k, where we have $I^k \subseteq \mathcal{I}$, $S^k \subseteq \mathcal{S}$, $O^k \subseteq \mathcal{O}$, and $A^k \subseteq \mathcal{A}$; for entities, we have $E^k \subseteq \mathcal{E}$ and $\hat{E}^k \subseteq \mathcal{E}$. For communication, we assume a message space $\mathcal{M}$ shared across tasks such that $M^1 = M^2 = ... = M^K = \mathcal{M}$.

In the partial 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 $m_i^k = f^{msg}(o_i^k; \theta_{enc}) \in \mathcal{M}$, 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 depends 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 generate, transmit, aggregate, and integrate messages in our proposed method MCS, while reducing communication overhead across tasks. MCS further enhances coordination by maximizing the mutual information between communicated messages and the policy distribution of agents for each task. An overview of the MCS framework is shown in Figure 1. During training, experiences across all tasks are jointly collected to learn a task-invariant communication protocol shared across tasks. During execution, the learned communication protocol can generate different communication graphs based on task-specific observations. As a result, MCS enables agents to communicate, execute, and learn multiple tasks simultaneously.

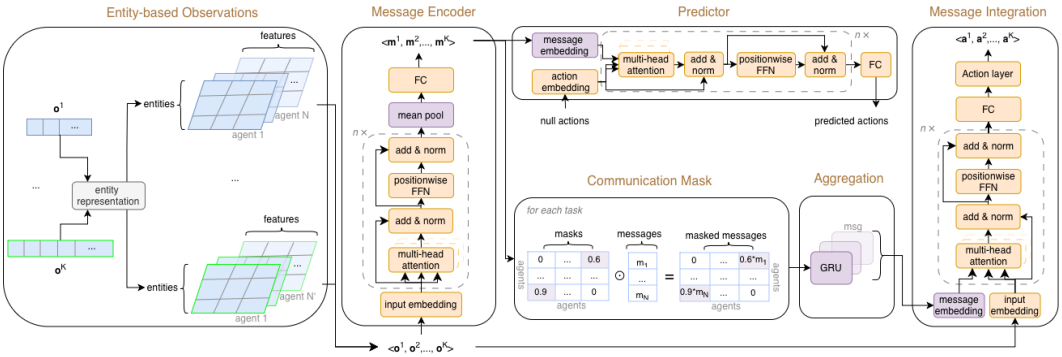


Figure 2: The communication protocol of MCS, including the architectures of all components. The yellow and purple rectangles include learnable parameters. Task-specific observations $\boldsymbol{o}^k$ are first transformed into entity-based representations and encoded into messages $\boldsymbol{m}^k$. During communication, these messages are pruned by masks $\boldsymbol{C}^k$ via column-wise multiplication, where rows and columns correspond to agents. The masked messages are then aggregated using a GRU and integrated into the policy. During training, the predictor takes the communicated messages $\boldsymbol{m}^k$ as query embeddings to predict the sender agents' used actions.

4.1 Entity-based Communication in Multi-task MADRL

We schematically illustrate the communication protocol of MCS in Figure 2. Inspired by entity-based representations used in multi-task MADRL without communication (Tian et al., 2023), 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). Such entity-based representations can unify observation representations across different tasks. However, in MADRL with communication, message dimensionality grows with the number of observed entities and total amount of communication further grows with the number of communicating agents. Thus, we conduct additional design 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. We explain the modules of MCS (Figure 2), detailing how agents encode, transmit, aggregate, and integrate messages in the following section.

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 $\boldsymbol{m}^k = \langle m_1^k, \ldots, m_N^k \rangle = \langle f^{msg}(o_1^k; \theta_{enc}), \ldots, f^{msg}(o_N^k; \theta_{enc}) \rangle$ and N is the number of agents. We abbreviate $\boldsymbol{m}^k = f^{msg}(\boldsymbol{o}^k; \theta_{enc})$.

Communication Mask. We further prune unnecessary messages while leveraging an attention mechanism to measure the importance of communication between agents. By introducing a com-

munication threshold, we construct a communication mask that prevents redundant messages and scales the remaining ones according to their importance. In particular, we employ an additive attention mechanism (Zhang et al., 2025; Lee et al., 2025) in the multi-task setting to produce scores $\alpha_{i,j}^k \in [0,1]$, which quantify the importance of sender agent i communicating with receiver agent j ($j \neq i$) 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. We then define communication mask $C_{i,j}^k \in [0,1]$ based on the scores $\alpha_{i,j}^k$:[2]

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

where $\hat{\alpha} \in [0,1]$ is a predefined communication threshold. A larger value of $\hat{\alpha}$ indicates a lower communication overhead from the receiver's perspective. We denote masked messages received by receiver agent j from sender agent i in 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.

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 remaining 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:

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

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 $\tilde{h} \in \{1, \ldots, \tilde{H}\}$ first computes 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 $\mathcal{V}_j^{(\tilde{h})} = o_j^k W_{\mathcal{V}}^{(\tilde{h})}$. These embeddings are then used to obtain hidden representation z_j^k for critics and policies. We have:

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

where $W_{\mathcal{Q}}$, $W_{\mathcal{K}}$, $W_{\mathcal{V}}$, and W_z are learnable weight matrices shared across agents and tasks, and $||$ denotes concatenation. Notably, the query $\mathcal{Q}j^{(\tilde{h})}$ (using messages) and the key $\mathcal{K}j^{(\tilde{h})}$ (using entity-based observations) first produce weighting scores $\mu_j^{(\tilde{h})}$ over entities, which indicate the relevance of

[2] We use soft differentiable samples via the Gumbel-Softmax to generate scores α_{ij}^k.

the received message to each observable entity of receiver j. These weighting scores are then used to combine the entity-based representations $\mathcal{V}_j^{(\tilde{h})}$ to form z_j^k, enabling receiver j to focus on the most relevant entities during communication. With hidden representation z_j^k, a final action layer is used to decide actions for agents, following Hu et al. (2021) and Tian et al. (2023).

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 $\bar{m}_i^k = f^{agg}\Big(\boldsymbol{C}_i^k \odot f^{msg}(\boldsymbol{o}^k; \theta_{enc})\Big)$. During centralized training, the encoder parameters θ_{enc}, policy parameters θ_π, and critic parameters ϕ are shared across agents and tasks.

4.2 Enhancing Coordination Among Agents

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 $\boldsymbol{m}^k \in \mathcal{M}$ with the sender agents' actions $\boldsymbol{a}^k \in A^k$ used in the environment, thereby enhancing coordinated action selection among agents for each task k. We provide the network architecture of the predictor q_{pred} in Figure 2. Specifically, the predictor q_{pred} leverages a Transformer decoder, which predicts sender agents' actions $\boldsymbol{a}^k$ based on communicated messages $\boldsymbol{m}^k$ in the forward pass. In the backward pass, we train the predictor by maximizing the mutual information $I(A^k; \mathcal{M})$ between the actions and communicated messages for each task k. However, computing the mutual information $I(A^k; \mathcal{M})$ is intractable. Therefore, we seek to maximize its variational lower bound (known as Barber-Agakov lower bound) (Poole et al., 2019). By replacing the conditional distribution of actions given messages with a variational distribution $q_{pred}(\boldsymbol{a}^k \mid \boldsymbol{m}^k)$, we have:

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

where $H(\boldsymbol{a}^k) = -\mathbb{E}_{p(\boldsymbol{a}^k)}[\log p(\boldsymbol{a}^k)]$ 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 maximizes a variational lower bound on 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 batches. 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:

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

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

Algorithm 1 Multi-task Communication Skills

1: **Input**: Batch size B, number of tasks K, episodes L, steps T.
2: **Initialize**: Parameters of encoder θ_{enc}, policy θ_π, and critic 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 aggregated messages $(\bar{\boldsymbol{m}}_t^1, ..., \bar{\boldsymbol{m}}_t^K)$ in Equation 4
11: Collect rewards $(r_t^1, ..., r_t^K)$ and insert experience $(\boldsymbol{o}_t^k, \boldsymbol{a}_t^k, r_t^k)$ 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 through end-to-end training
21: **end for**

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.

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 masks defined in Equations 1 and 2 and the aggregation function defined 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 11). 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 batches (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 (Yang et al., 2023), StarCraftII Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019),

and Google Research Football (GRF) (Kurach et al., 2020).[3] 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. (2023), we construct multi-task settings for each environment in Section 5.1. We compare the following methods across these multiple tasks:

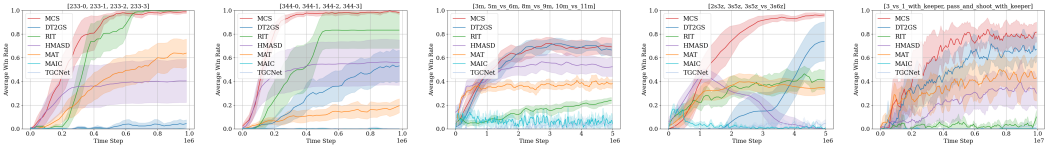
- **Multi-task MADRL methods with communication**. Our proposed method, MCS, is the first in this branch, which is a Transformer-based approach that allows communication among agents for better coordination in multiple tasks.
- **Multi-task MADRL methods without communication**. In this branch, DT2GS (Tian et al., 2023) and RIT (Li et al., 2025a) 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 (Zhang et al., 2025) and MAIC (Yuan et al., 2022). 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 (Wen et al., 2022) is the SOTA method on many SMAC tasks, which employs a Transformer to encode observations for generating action sequences. HMASD (Yang et al., 2023) 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. This results in two essential perspectives for comparison: Multi-task with vs. without communication, and Multi-task vs. Single-task. By following the literature (Tian et al., 2023), we evaluate both the average win rate across the K tasks, defined as $\frac{1}{K}\sum_{k=1}^{K} WinRate(k)$, where $WinRate(k)$ denotes the win rate evaluated on task k, and the per-task win rate $WinRate(k)$ evaluated on each individual task. The win rate $WinRate(k)$ for each task is computed by running 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, which are sufficient to demonstrate converged performance. Moreover, we allocate sufficient computational budget (1/2 A100 GPU) to MCS and baselines for each seed. The hyperparameters for each environment are adopted either from published sources or obtained through fine-tuning, provided in Appendix B. To enable a fair comparison, critical hyperparameters (e.g., learning rates) are shared across methods. For all environments, we set the same 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 how different thresholds $\hat{\alpha}$ affect the learning performance in Section 5.2. Detailed implementation of the entity-based representations are provided in Appendix A. We also report the computational time of all methods in Appendix B, showing that MCS introduces no significant additional runtime. In all figures, the shaded regions indicate the 95% confidence interval around the average performance.

5.1 Evaluation Domains

The AliceBob environment is 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

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



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

Figure 3: Average win-rate across multiple tasks on AliceBob (a–b), SMAC (c–d), and GRF (e).

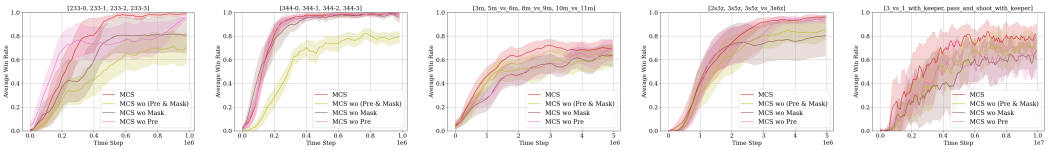
button of the same color. We extend this environment to a multi-task setting where each task consists of either diamonds or food, and either buttons or keys, following the literature (Tian et al., 2023). For example, task 0 may involve 3 pairs of entities $\{red\,diamond,\ red\,button\}, \{blue\,diamond,\ blue\,button\}, \{pink\,diamond,\ pink\,button\}$ 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 $\{344\text{–}0, 344\text{–}1, 344\text{–}2, 344\text{-}4\}$ by introducing an additional agent (Charlie) as well as an additional color and entity (e.g., blue flower). To increase task complexity, diamonds and trees are randomly placed at the top of the environment, while keys and buttons are randomly placed at the bottom at the beginning of each episode. Agents are also spawned at random positions, requiring effective coordination 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 (Tian et al., 2023). 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 in 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 3, with the corresponding statistical analysis provided in Appendix B. As shown in Figure 3, MCS achieves much higher average win rates than other methods in `AliceBob-344`, `Stalker-Zealot`, and `Football` 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 $\sim$0.6M steps). As the number of agents increases, RIT's domain mask scales quadratically, which increases learning complexity and may explain its lower average win rate than MCS 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.



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

Figure 4: Ablations of MCS. When ablating Pre and Mask, we keep the common components fixed.

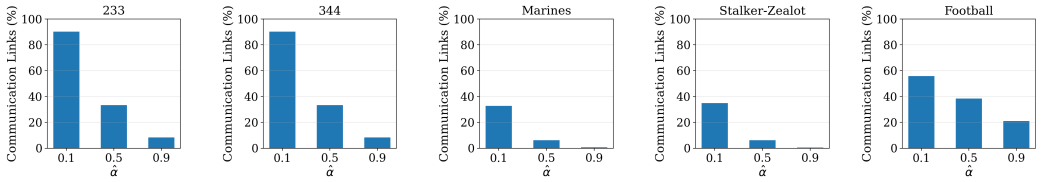
We compute per-task win rate for multi-task and single-task methods in Appendix B. Notably, for all tasks in `AliceBob-233`, `AliceBob-344`, `Stalker-Zealot`, and `Football`, MCS consistently achieves comparable or higher win rates compared to other methods. In `Marine` series, DT2GS outperforms MCS in one task $10m\_vs\_11m$, 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 4, 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`, 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 of MADRL agents with communication in multi-task settings.

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`, where mean aggregation ignores inter-agent dependencies and Transformer aggregation can increase the depth and learning complexity of models. 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 important role. Detailed ablation results are provided in Appendix B.

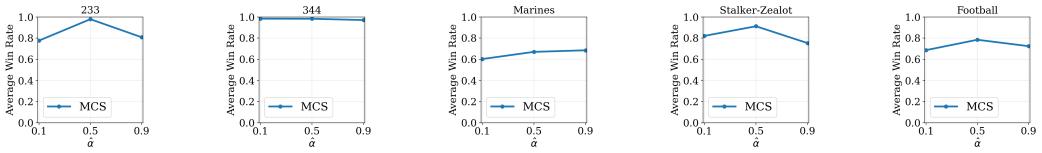
Communication Threshold. In MCS, we use a communication threshold ($\hat{\alpha}$) to reduce unnecessary communication among agents and improve communication efficiency. To quantify this effect, we analyze the percentage of allowed communication links between pairs of agents relative to all possible communication links. We report the average number of communication links over the last 20 evaluation episodes in Figure 5. As shown in the figure, increasing $\hat{\alpha}$ consistently reduces the number of active communication links, indicating improved communication efficiency. The reduction is particularly pronounced in SMAC compared with the other environments, potentially because this environment favors more structured communication patterns.

We further investigate the effect of the threshold $\hat{\alpha}$ on learning performance, where we set $\hat{\alpha} \in \{0.1, 0.5, 0.9\}$. A larger threshold $\hat{\alpha}$ corresponds to lower communication overhead. Figure 6



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

Figure 5: The percentage of communication links among agents under different values of $\hat{\alpha}$.



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

Figure 6: Average win rate achieved by MCS under different values of threshold $\hat{\alpha}$ in all environments.

reports the average win rate over the last 20 episodes under different values of $\hat{\alpha}$. As shown, in `AliceBob-344`, all values of $\hat{\alpha}$ achieve a similar learning performance, demonstrating robustness to different communication overhead. For `AliceBob-233`, `Stalker-Zealot`, and `Football`, a larger threshold ($\hat{\alpha} = 0.9$) can decrease the learning performance, indicating that these environments less benefit from limited communication. In contrast, in `Marine`, MCS tends to perform better with larger values of $\hat{\alpha}$, where communication is much reduced. Overall, a robust configuration is $\hat{\alpha} = 0.5$, suggesting that moderate communication overhead 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 those produced by DT2GS, which also employs a Transformer to encode observations for policy learning but does not incorporate communication. The detailed analysis is provided in Appendix B. Notably, in AliceBob, both methods learn similar representations across tasks, while MCS forms more a compact message representation. In SMAC and Football, the message representations in MCS better captures task differences, such as changes in coordination strategies when the number of agents changes. Overall, across 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 may explain its superior empirical performance.

6 Conclusion

We propose Multi-task Communication Skills (MCS), a multi-task MADRL method with communication that leverages a task-invariant communication protocol that is shared under tasks with varying numbers of agents, observation spaces, and action spaces. We further introduce a predictor that maximizes the mutual information between communicated messages and agents' actions to promote coordination for each task. 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 communication graphs.

References

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

Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. Tarmac: Targeted multi-agent communication. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pp. 1538–1546, 2019.

Christian Schröder de Witt, Jakob N. Foerster, Gregory Farquhar, Philip H. S. Torr, Wendelin Boehmer, and Shimon Whiteson. Multi-agent common knowledge reinforcement learning. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 9924–9935, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/f968fdc88852a4a3a27a81fe3f57bfc5-Abstract.html.

Ziluo Ding, Tiejun Huang, and Zongqing Lu. Learning individually inferred communication for multi-agent cooperation. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020.

Scott Fujimoto, Wei-Di Chang, Edward J. Smith, Shixiang Gu, Doina Precup, and David Meger. For SALE: state-action representation learning for deep reinforcement learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *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*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/c20ac0df6c213db6d3a930fe9c7296c8-Abstract-Conference.html.

Cong Guan, Feng Chen, Lei Yuan, Chenghe Wang, Hao Yin, Zongzhang Zhang, and Yang Yu. Efficient multi-agent communication via self-supervised information aggregation. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *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*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/075b2875e2b671ddd74aeec0ac9f0357-Abstract-Conference.html.

Shuai Han, Mehdi Dastani, and Shihan Wang. Model-based sparse communication in multi-agent reinforcement learning. In Noa Agmon, Bo An, Alessandro Ricci, and William Yeoh (eds.), *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023, London, United Kingdom, 29 May 2023 - 2 June 2023*, pp. 439–447. ACM, 2023. DOI: 10.5555/3545946.3598669. URL https://dl.acm.org/doi/10.5555/3545946.3598669.

Shengchao Hu, Li Shen, Ya Zhang, and Dacheng Tao. 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, 2024. URL https://openreview.net/forum?id=Qox9rO0kN0.

Siyi Hu, Fengda Zhu, Xiaojun Chang, and Xiaodan Liang. 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, 2021. URL https://openreview.net/forum?id=v9c7hr9ADKx.

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

Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. In *Advances in Neural Information Processing Systems 31 (NIPS)*, pp. 7265–7275, 2018.

Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu. Graph convolutional reinforcement learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HkxdQkSYDB.

Woojun Kim, Jongeui Park, and Youngchul Sung. Communication in multi-agent reinforcement learning: Intention sharing. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=qpsl2dR9twy.

Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *Int. J. Robotics Res.*, 32(11):1238–1274, 2013. DOI: 10.1177/0278364913495721.

Karol Kurach, Anton Raichuk, Piotr Stanczyk, Michal Zajac, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, and Sylvain Gelly. Google research football: A novel reinforcement learning environment. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 4501–4510. AAAI Press, 2020. DOI: 10.1609/AAAI.V34I04.5878. URL https://doi.org/10.1609/aaai.v34i04.5878.

Dongsu Lee, Daehee Lee, Yaru Niu, Honguk Woo, Amy Zhang, and Ding Zhao. Unifying agent interaction and world information for multi-agent coordination. In *NeurIPS 2025 Workshop: Second Workshop on Aligning Reinforcement Learning Experimentalists and Theorists*, 2025.

Chao Li, Shaokang Dong, Shangdong Yang, Yujing Hu, Tianyu Ding, Wenbin Li, and Yang Gao. Multi-task multi-agent reinforcement learning with interaction and task representations. *IEEE Trans. Neural Networks Learn. Syst.*, 36(7):13431–13445, 2025a. DOI: 10.1109/TNNLS.2024.3475216. URL https://doi.org/10.1109/TNNLS.2024.3475216.

Dapeng Li, Na Lou, Zhiwei Xu, Bin Zhang, and Guoliang Fan. Efficient communication in multi-agent reinforcement learning with implicit consensus generation. In Toby Walsh, Julie Shah, and Zico Kolter (eds.), *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pp. 23240–23248. AAAI Press, 2025b. DOI: 10.1609/AAAI.V39I22.34490. URL https://doi.org/10.1609/aaai.v39i22.34490.

Sicong Liu, Yang Shu, Chenjuan Guo, and Bin Yang. 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, 2025. URL https://openreview.net/forum?id=HR1ujVR0ig.

Yong Liu, Weixun Wang, Yujing Hu, Jianye Hao, Xingguo Chen, and Yang Gao. Multi-agent game abstraction via graph attention neural network. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, pp. 7211–7218, 2020.

Mridul Mahajan, Georgios Tzannetos, Goran Radanovic, and Adish Singla. 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*, pp. 4733–4741. ijcai.org, 2024. URL https://www.ijcai.org/proceedings/2024/523.

Hangyu Mao, Zhengchao Zhang, Zhen Xiao, Zhibo Gong, and Yan Ni. Learning agent communication under limited bandwidth by message pruning. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pp. 5142–5149, 2020.

Leland McInnes and John Healy. UMAP: uniform manifold approximation and projection for dimension reduction. *CoRR*, abs/1802.03426, 2018. URL http://arxiv.org/abs/1802.03426.

Yaru Niu, Rohan R. Paleja, and Matthew C. Gombolay. Multi-agent graph-attention communication and teaming. In *20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 964–973, 2021.

Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P. How, and John Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2681–2690. PMLR, 2017. URL http://proceedings.mlr.press/v70/omidshafiei17a.html.

M. Pipattanasomporn, H. Feroze, and S. Rahman. Multi-agent systems in a distributed smart grid: Design and implementation. In *2009 IEEE/PES Power Systems Conference and Exposition*, pp. 1–8, 2009. DOI: 10.1109/PSCE.2009.4840087.

Ben Poole, Sherjil Ozair, Aäron van den Oord, Alexander A. Alemi, and George Tucker. On variational bounds of mutual information. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5171–5180. PMLR, 2019. URL http://proceedings.mlr.press/v97/poole19a.html.

Rongjun Qin, Feng Chen, Tonghan Wang, Lei Yuan, Xiaoran Wu, Yipeng Kang, Zongzhang Zhang, Chongjie Zhang, and Yang Yu. Multi-agent policy transfer via task relationship modeling. *Sci. China Inf. Sci.*, 67(8), 2024. DOI: 10.1007/S11432-023-3862-1. URL https://doi.org/10.1007/s11432-023-3862-1.

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. The starcraft multi-agent challenge. In Edith Elkind, Manuela Veloso, Noa Agmon, and Matthew E. Taylor (eds.), *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019*, pp. 2186–2188. International Foundation for Autonomous Agents and Multiagent Systems, 2019. URL http://dl.acm.org/citation.cfm?id=3332052.

Lukas Schäfer, Filippos Christianos, Amos Storkey, and Stefano V. Albrecht. Learning task embeddings for teamwork adaptation in multi-agent reinforcement learning. In *NeurIPS Workshop on Generalization in Planning*, 2023.

Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *CoRR*, abs/1610.03295, 2016.

Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

Shagun Sodhani, Amy Zhang, and Joelle Pineau. Multi-task reinforcement learning with context-based representations. In *International conference on machine learning*, pp. 9767–9779. PMLR, 2021.

Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 2244–2252, 2016.

Chuxiong Sun, Zehua Zang, Jiabao Li, Jiangmeng Li, Xiao Xu, Rui Wang, and Changwen Zheng. T2MAC: targeted and trusted multi-agent communication through selective engagement and evidence-driven integration. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (eds.), *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*, pp. 15154–15163. AAAI Press, 2024. DOI: 10.1609/AAAI.V38I13.29438.

Zikang Tian, Ruizhi Chen, Xing Hu, Ling Li, Rui Zhang, Fan Wu, Shaohui Peng, Jiaming Guo, Zidong Du, Qi Guo, and Yunji Chen. Decompose a task into generalizable subtasks in multi-agent reinforcement learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *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*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/f7d3cef7ff579f2f903c8f458e730cae-Abstract-Conference.html.

Tonghan Wang, Jianhao Wang, Chongyi Zheng, and Chongjie Zhang. Learning nearly decomposable value functions via communication minimization. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

Muning Wen, Jakub Grudzien Kuba, Runji Lin, Weinan Zhang, Ying Wen, Jun Wang, and Yaodong Yang. Multi-agent reinforcement learning is a sequence modeling problem. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *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*, 2022.

Mingyu Yang, Yaodong Yang, Zhenbo Lu, Wengang Zhou, and Houqiang Li. Hierarchical multi-agent skill discovery. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *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*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/c276c3303c0723c83a43b95a44a1fcbf-Abstract-Conference.html.

Lei Yuan, Jianhao Wang, Fuxiang Zhang, Chenghe Wang, Zongzhang Zhang, Yang Yu, and Chongjie Zhang. Multi-agent incentive communication via decentralized teammate modeling. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pp. 9466–9474. AAAI Press, 2022. DOI: 10.1609/AAAI.V36I9.21179. URL https://doi.org/10.1609/aaai.v36i9.21179.

Fuxiang Zhang, Chengxing Jia, Yi-Chen Li, Lei Yuan, Yang Yu, and Zongzhang Zhang. 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, 2023. URL https://openreview.net/forum?id=53FyUAdP7d.

Sai Qian Zhang, Qi Zhang, and Jieyu Lin. Efficient communication in multi-agent reinforcement learning via variance based control. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pp. 3230–3239, 2019.

Sai Qian Zhang, Qi Zhang, and Jieyu Lin. Succinct and robust multi-agent communication with temporal message control. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33 (NIPS)*, 2020.

Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Trans. Knowl. Data Eng.*, 34(12): 5586–5609, 2022. DOI: 10.1109/TKDE.2021.3070203. URL https://doi.org/10.1109/TKDE.2021.3070203.

Zhuohui Zhang, Bin He, Bin Cheng, and Gang Li. Bridging training and execution via dynamic directed graph-based communication in cooperative multi-agent systems. In Toby Walsh, Julie Shah, and Zico Kolter (eds.), *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pp. 23395–23403. AAAI Press, 2025. DOI: 10.1609/AAAI.V39I22.34507. URL https://doi.org/10.1609/aaai.v39i22.34507.

Changxi Zhu, Mehdi Dastani, and Shihan Wang. A survey of multi-agent deep reinforcement learning with communication. *Autonomous Agents and Multi-Agent Systems*, 38(4), 2024. DOI: 10.1007/s10458-023-09633-6.

Supplementary Materials

The following content was not necessarily subject to peer review.

A Details of Entity-based Representations

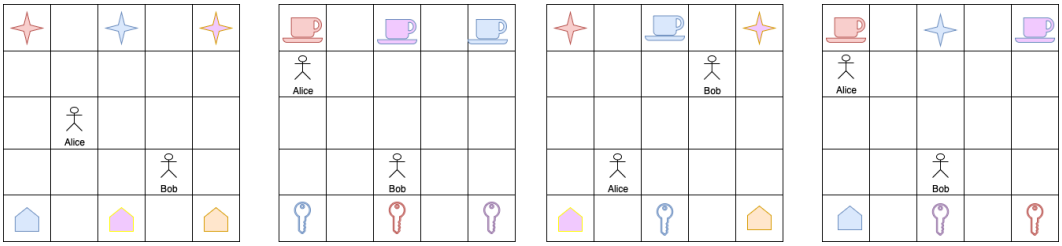


Figure 7: An illustration of `AliceBob-233` series. Agents, diamonds/food, and buttons/keys are randomly placed on the map in each episode.

Figure 8: SMAC `Stalker-Zealot` series with 2s3z (Left), 3s5z (Middle), and 3s5z_vs_3s6z (Right).

Figure 9: `Football` series with 3 vs 1 with keeper (Left) and Pass and shot with keeper (Right).

AliceBob. The AliceBob environment was originally proposed for single-task MADRL without considering entity-based representations. Following the approaches of Hu et al. (2021) and Tian et al. (2023), we construct an entity-based representation and adapt the AliceBob environment to multi-task settings. In the entity-based representation, entities consist of RL agents, buttons/keys, and diamonds/food. For each entity, we extract partially observable features, including the relative distance from each agent to all other entities and a one-hot representation of the observed entity type. To reduce learning complexity, we further enrich each agent's observation by incorporating features of the surrounding grid cells using the same representation scheme. The resulting entity representation thus comprises: (i) features of the agent itself, (ii) features of other agents, (iii) features of diamonds/food, (iv) features of keys/buttons, and (v) features of the surrounding grids. For example, in the `AliceBob-233` series, the observation shape becomes 16×17 where 16 is the total number of entities (8 original entities plus 8 surrounding grids), and 17 is the number of features (2 dimensions for the relative distance and 15 dimensions for the one-hot encoding). The reward function is defined as follows: agents receive a reward of 1 whenever a valid pair (diamond–button or food–key) is completed, a reward of 5 when all pairs are completed (regarded as a win), a penalty of 0.5 for agent collisions, and a penalty of 0.1 per step. We provide an example of the multi-task setting in the `AliceBob-233` series in Figure 7, where two agents must coordinate to collect

Table 1: Critical hyperparameters used in different environments.

Hyperparameter	AliceBob	SMAC	GRF
lr	5e-4	5e-4	5e-4
$critic\_lr$	5e-4	5e-4	5e-4
$opti\_eps$	1e-5	1e-5	1e-5
PPO epochs	8	8	10
mini-batches	10	8	2
$clip\_param$	0.2	0.2	0.2
entropy coefficient	0.01	0.01	0.01
max grad norm	10	10	10
γ	0.99	0.99	0.99
GAE λ	0.95	0.95	0.95
hidden size	64	64	64
message dimension	10	10	10
batch size	32	32	32
number of steps	1e6	5e6	1e7
evaluation episodes	32	32	32

diamonds/food and press buttons/keys separately. The type of entities is determined before the start of the game, while their positions are randomly generated at the beginning of each episode to create dynamic targets. Agents can leverage the knowledge acquired from collecting diamonds or pressing buttons in one task to improve their learning performance in another task.

SMAC. We use the same multi-task environment as used by Hu et al. (2021) and Tian et al. (2023), where entities consist of the agent itself, allied agents, and enemies, as illustrated in Figure 8. The feature set includes visibility, distance to the agent, relative x and y positions, health, shield, and unit type of each entity. Then, for each entity, a total of 16 feature dimensions are considered.[4]

GRF. We adapt the Google Research Football (Kurach et al., 2020) environment to a multi-task setting with entity-based representations. Starting from the raw features in `simple115v2`, we group them into four macro-entities: left-team locations, left-team directions, right-team locations, and right-team directions. Ball position and direction are appended to the appropriate entity depending on possession, and a one-hot encoding of the active player is included. This results in 43 feature dimensions per entity, with the total number of features across all entities remaining similar to the original raw representation, thereby preserving the overall complexity of the environment. We illustrate two tasks from the `Football` series $\{3\_vs\_1\_with\_keeper, pass\_and\_shoot\_with\_keeper\}$, as shown in Figure 9. In $3\_vs\_1\_with\_keeper$, three left-team players start in the right half, competing against one right-team defender and the goalkeeper. In $pass\_and\_shoot\_with\_keeper$, two left-team players start in the right half against one right-team defender and the goalkeeper. An episode terminates when (a) the maximum duration (200 steps) is reached, (b) the ball goes out of bounds, (c) a team scores, or (d) possession changes.

B Hyperarameters and Additional Results

Hyperparameters. The critical hyperparameters shared by actor-critic methods in AliceBob, SMAC, and GRF are summarized in Table 1. For SMAC environment, we use the same hyperparameters as in DT2GS (Tian et al., 2023), which have already been fine-tuned. For AliceBob environment, we adopt similar values but use a slightly larger mini-batch size to obtain more samples for training. For the GRF environment, we use the same hyperparameters as those employed in the single-task setting.

Computational Time. For each method, experimental seeds were run in parallel on a cluster, with each seed using 32 CPU cores and $1/2$ of an NVIDIA A100 GPU. Table 2 reports the

[4]Implementation available at: https://github.com/Felixvillas/DT2GS

Table 2: Average wall-clock runtime (**hours**) per method across tasks under each multi-task series.

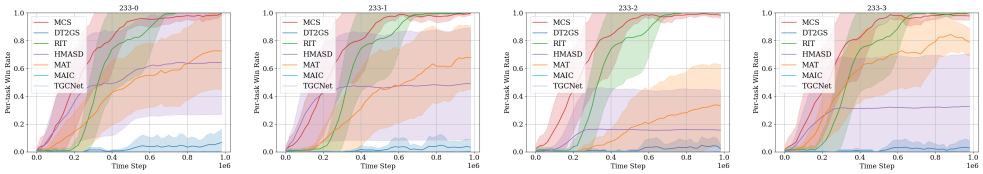
Method	AliceBob-233	AliceBob-344	Marine	Stalker-Zealot	Football
MCS	**4**	**4**	**23**	**10**	**13**
DT2GS	6	6	7	9	10
RIT	7	8	67	81	116
HMASD	1	1	6	6	8
MAT	1	1	6	8	8
MAIC	4	4	6	9	29
TGCNet	4	4	13	12	25

wall-clock training time for each method on each multi-task series; values are averaged over the tasks within a series. In AliceBob, the single-task methods HMASD and MAT are most compute-efficient, typically finishing within ≤ 1 hour per run. DT2GS is moderately heavier (6h), while MCS/MAIC/TGCNet sit around 4–5h. RIT is the slowest even on small tasks (7–8h). In SMAC (`Marine` and `Stalker-Zealot`) and `Football`, runtimes spread more widely. RIT scales poorly (tens to > 100h), suggesting unfavorable difficulty in complex environments. MCS remains reasonably efficient on most tasks, but requires longer training time on `Marines` (23h), where the series includes four tasks and up to 26 communicating agents in total, which may increase the computational cost. MAIC is fast on Marines (6h) yet slow on `Football` (29h). Overall, the proposed MCS does not significantly increase runtime compared to other multi-task MADRL baselines on all multi-task series, though it is slightly slower than DT2GS on `Marines`. Single-task baselines (HMASD/MAT) achieve much lower per-series runtime as they train only one task at the same time.

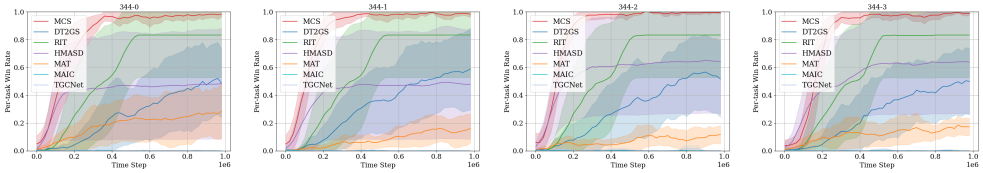
Per-task Win Rate. We further report the per-task performance of each method across all multi-task series, as shown in Figure 10 and Table 3. Note that the absence of visible baselines in Figure 10 is due to their much lower win rates. As illustrated in Figures 10a–10b, MCS consistently achieves faster convergence and significantly higher win rates across all tasks in the AliceBob environment. In SMAC `Marines` (Figure 10c), MCS outperforms DT2GS in the $3m$ and $5m\_vs\_6m$ tasks, while DT2GS surpasses MCS in $10m\_vs\_11m$, potentially due to the benefits of task decomposition in the task. Despite these differences, MCS and DT2GS achieve similar average performance across tasks (as shown in the table). For single-task baselines, methods such as HMASD achieve high win rates in $5m\_vs\_6m$ and $10m\_vs\_11m$. However, their performance varies significantly across tasks, resulting in lower overall average performance. In the `Stalker-Zealots` series, MCS consistently outperforms all baselines in terms of learning performance. In `Football` series, MCS achieves a significantly higher win rate in the $3\_vs\_1\_with\_keeper$ task. In the $pass\_and\_shoot\_with\_keeper$ task, the single-task baseline MAT performs similar to MCS. However, MAT's performance drops significantly in $3\_vs\_1\_with\_keeper$, where MCS maintains a consistent high win rate.

Statistical Analysis. Following the reinforcement learning and multi-task learning literature (Fujimoto et al., 2023; Sodhani et al., 2021), we conduct a two-sided Welch's t-test, i.e., a Student's t-test under an unequal-variance assumption, with significance level of 0.05. As shown in Table 3, we highlight MCS and methods that are not statistically significantly different from MCS, and underline the methods that achieve the highest average win rates for each multi-task series. In `AliceBob-233`, MCS and RIT perform significantly better than the other methods, while in `AliceBob-344`, MCS achieves the highest average win rate. In SMAC `Marines` series, MCS and DT2GS are significantly better than the other methods. In both SMAC `Stalker-Zealots` and `Football` series, MCS significantly outperforms all other methods. Overall, the statistical analysis shows that MCS consistently achieves performance that is either comparable to or significantly better than the other methods across all multi-task series.

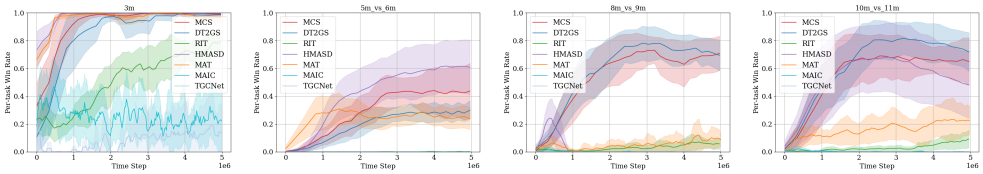
Ablations on Message Aggregation. We investigate alternative message aggregation functions beyond GRU, including mean aggregation and Transformer-based aggregation. Intuitively, mean aggregation ignores inter-agent dependencies, while Transformers are more difficult to train due to the increased depth and complexity of models. As we can see in Table 4, both alternatives (i.e.,



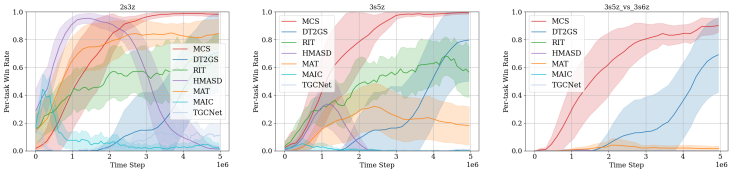
(a) AliceBob-233



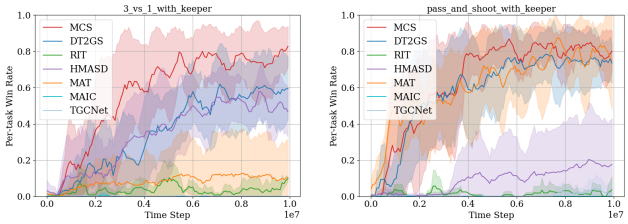
(b) AliceBob-344



(c) Marine



(d) Stalker Zealots



(e) Football

Figure 10: Per-task win rate curves across different environments: AliceBob-233/344, SMAC Marine/Stalker Zealots, and Football. The shaded area captures a 95% confidence interval.

Mean and Transformer) exhibit performance degradation compared to GRU used in our proposed method MCS in `AliceBob-233` and `Stalker-Zealot`. In addition, we examine different sampling strategies for training the GRU aggregation: a fixed order, where agents are sampled in a predetermined sequence, and a random order (adopted in MCS), where agents are sampled randomly. As shown in Table 4, in the two-agent case (i.e., `AliceBob-233`), random order allows each agent to communicate first, which improves learning performance compared to using a fixed order. For scenarios with more than two agents (e.g., `AliceBob-344` and `Stalker-Zealot`), the performance gap becomes marginal, as the communication topology plays a more dominant role.

Analysis of Message Representations. We 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

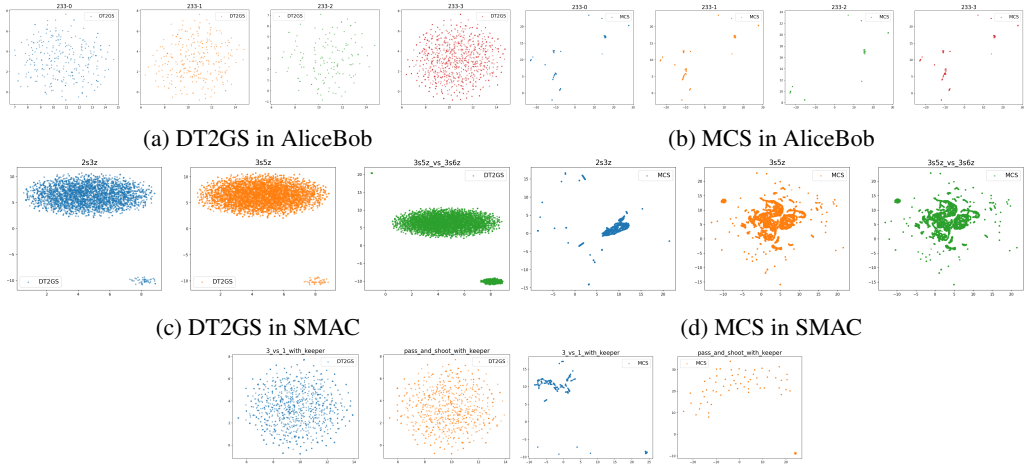
Table 3: Per-task win rates and average win rates for all benchmark multi-task series, where ± denotes the 95% confidence interval rounded to two decimals. The highest average performance is underlined. MCS is highlighted in blue, and methods that are not statistically significantly different from MCS are highlighted in orange, according to Welch's t-test at a significance level of 0.05. Columns correspond to individual tasks, and the rightmost column reports the average across tasks.

AliceBob-233					
Method	**233-0**	**233-1**	**233-2**	**233-3**	**Avg**
MCS	0.97 ± 0.02	0.98 ± 0.01	0.98 ± 0.01	0.98 ± 0.01	0.98 ± 0.00
DT2GS	0.04 ± 0.10	0.03 ± 0.08	0.03 ± 0.07	0.03 ± 0.06	0.03 ± 0.03
RIT	0.99 ± 0.03	0.98 ± 0.03	0.98 ± 0.03	0.98 ± 0.03	0.99 ± 0.01
HMASD	0.63 ± 0.52	0.48 ± 0.55	0.16 ± 0.40	0.32 ± 0.52	0.40 ± 0.20
MAT	0.64 ± 0.34	0.60 ± 0.34	0.28 ± 0.40	0.77 ± 0.23	0.57 ± 0.14
MAIC	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
TGCNet	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
AliceBob-344					
Method	**344-0**	**344-1**	**344-2**	**344-3**	**Avg**
MCS	0.97 ± 0.01	0.98 ± 0.01	0.99 ± 0.01	0.98 ± 0.00	0.98 ± 0.00
DT2GS	0.43 ± 0.36	0.52 ± 0.35	0.49 ± 0.33	0.44 ± 0.30	0.47 ± 0.14
RIT	0.83 ± 0.43	0.83 ± 0.43	0.83 ± 0.43	0.83 ± 0.43	0.83 ± 0.17
HMASD	0.47 ± 0.54	0.48 ± 0.55	0.64 ± 0.52	0.63 ± 0.51	0.56 ± 0.22
MAT	0.25 ± 0.24	0.14 ± 0.11	0.09 ± 0.06	0.16 ± 0.05	0.16 ± 0.06
MAIC	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
TGCNet	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.01	0.00 ± 0.00	0.00 ± 0.00
Marine					
Method	**10m_vs_11m**	**3m**	**5m_vs_6m**	**8m_vs_9m**	**Avg**
MCS	0.67 ± 0.10	0.99 ± 0.01	0.40 ± 0.14	0.68 ± 0.11	0.68 ± 0.04
DT2GS	0.78 ± 0.19	0.97 ± 0.07	0.27 ± 0.04	0.74 ± 0.14	0.69 ± 0.05
RIT	0.06 ± 0.04	0.72 ± 0.09	0.00 ± 0.00	0.05 ± 0.04	0.21 ± 0.02
HMASD	0.61 ± 0.32	1.00 ± 0.00	0.56 ± 0.28	0.00 ± 0.00	0.54 ± 0.09
MAT	0.22 ± 0.15	0.98 ± 0.01	0.25 ± 0.09	0.09 ± 0.09	0.39 ± 0.04
MAIC	0.00 ± 0.00	0.25 ± 0.31	0.00 ± 0.00	0.00 ± 0.00	0.06 ± 0.05
TGCNet	0.00 ± 0.00	0.12 ± 0.32	0.00 ± 0.00	0.00 ± 0.00	0.03 ± 0.07
Stalker-Zealots					
Method	**2s3z**	**3s5z**	**3s5z_vs_3s6z**	**Avg**	
MCS	0.96 ± 0.02	0.96 ± 0.04	0.82 ± 0.09	0.91 ± 0.03	
DT2GS	0.33 ± 0.30	0.39 ± 0.33	0.30 ± 0.29	0.34 ± 0.15	
RIT	0.62 ± 0.21	0.62 ± 0.16	0.00 ± 0.00	0.41 ± 0.08	
HMASD	0.39 ± 0.15	0.00 ± 0.00	0.00 ± 0.00	0.13 ± 0.05	
MAT	0.83 ± 0.11	0.24 ± 0.19	0.02 ± 0.02	0.37 ± 0.07	
MAIC	0.03 ± 0.03	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.01	
TGCNet	0.11 ± 0.08	0.00 ± 0.00	0.00 ± 0.00	0.04 ± 0.02	
Football					
Method	**academy_3_vs_1_with_keeper**	**academy_pass_and_shoot_with_keeper**	**Avg**		
MCS	0.76 ± 0.21	0.81 ± 0.08	0.78 ± 0.10		
DT2GS	0.59 ± 0.25	0.74 ± 0.14	0.67 ± 0.12		
RIT	0.05 ± 0.05	0.03 ± 0.03	0.04 ± 0.02		
HMASD	0.50 ± 0.17	0.18 ± 0.68	0.34 ± 0.21		
MAT	0.11 ± 0.33	0.79 ± 0.10	0.45 ± 0.16		
MAIC	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00		
TGCNet	0.00 ± 0.00	0.03 ± 0.07	0.01 ± 0.04		

Table 4: Additional ablation results on different aggregation methods and GRU sampling strategies.

Task (Win Rate %)	Mean	Transformer	GRU (Ours)	Fixed Order	Random Order (Ours)
`AliceBob-233`	50	63	98	80 (0.6M)	92 (0.6M)
`AliceBob-344`	99	99	99	95	99
`Stalker-Zealot`	74	90	92	91	92

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 (McInnes & Healy, 2018) to project the high-dimensional latent representations into a two dimensional space,



(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 11: Latent representations of DT2GS and MCS projected by UMAP.

as shown in Figure 11, 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 different 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.