
LEARNING COMMUNICATION SKILLS IN MULTI-TASK MULTI-AGENT DEEP REINFORCEMENT LEARNING

Changxi Zhu

Department of Information and Computing Sciences
Utrecht University
Utrecht, Netherlands
c.zhu@uu.nl

Mehdi Dastani

Department of Information and Computing Sciences
Utrecht University
Utrecht, Netherlands
m.m.dastani@uu.nl

Shihao Wang

Department of Information and Computing Sciences
Utrecht University
Utrecht, 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. MCS employs a Transformer encoder to encode task-specific observations into a shared message space, capturing shared communication skills among agents. To enhance coordination among agents, we introduce a prediction network that correlates messages with the actions of sender agents in each task. We adapt three multi-agent benchmark environments to multi-task settings, where the number of agents as well as the observation and action spaces vary across tasks. Experimental 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

1 Introduction

Communication is essential in multi-agent deep reinforcement learning (MADRL) for enabling information sharing and enhancing collaboration among multiple agents toward common goals [1], especially in partially observable environments such as autonomous driving [2], sensor networks [3], and multi-robot control [4]. Recent works on learning communication in MADRL have investigated what information to communicate [5, 6], when and with whom to communicate [7, 8, 9] and how to integrate communication into policies [10, 11], thereby enabling flexible and dynamic communication. However, these approaches are limited to single-task settings, where both policies and communication protocols are designed and evaluated for only one task.

A different line of work in the MADRL literature focuses on multi-task learning, which aims to generalize policies across multiple tasks [12]. This is achieved either via task-invariant architectures [13, 14] that exploit the shared structure across tasks to learn a single unified model, or via task representation learning [15, 16] that explicitly models environment dynamics to generalize to unseen tasks. However, despite these achievements, previous works on multi-task learning in MADRL has 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. The integration of communication into multi-task MADRL is non-trivial as it involves complex processes regarding what, when, and how to communicate. Moreover, tasks may differ in observation and action spaces, making message generation challenging. To address these challenges, we propose encoding task-specific observations into a shared message space, capturing common communication skills that can generalize across tasks, therefore enabling effective and efficient communication in Multi-task Multi-Agent Deep Reinforcement Learning (Multi-task MADRL).

In this paper, we propose Multi-task Communication Skills (MCS), a multi-task MADRL with communication method. MCS leverages task-invariant architectures to learn shared communication protocols across tasks, thereby improving coordination among agents and accelerate learning performance in multi-task settings. During communication in each task, messages are encoded by Transformer encoders and exchanged over a learned communication graph, where unnecessary messages are pruned. The received messages are then aggregated through a shared aggregation function and used to attend to important features in policy and critic networks. To further enhance coordination, we introduce a prediction network that maximizes mutual information between messages and actions to encourage informative communication. To evaluate MCS, we conduct experiments on the well-known SMAC benchmark [17], a novel multi-task AliceBob environment, and an adapted multi-task version of Google Research Football [18]. We evaluate both the average performance across tasks and the per-task performance. Our results show that MCS significantly outperforms multi-task MADRL baselines without communication, as well as single-task MADRL baselines both with and without communication. We further conduct ablation studies to assess the effects of pruning unnecessary messages and of the prediction network on learning performance. We also analyze the sensitivity of hyperparameters and the learned representations of messages. As a result, agents in MCS can communicate and perform efficiently and effectively across multiple tasks with varying numbers of agents and diverse observation and action spaces.

2 Related Work

Multi-task Learning in MADRL Early approaches in multi-task MADRL, such as DEC-HDRQN [19], distilled single-task Q-networks into a unified Q-network through a distillation process. More recent methods focus either on task-invariant architectures [20, 13, 14, 21] or on learning task representations [22, 15, 16]. multi-task MADRL methods based on task-invariant architectures exploit shared task structure by unifying inputs across tasks using entity-based representations of observations and actions [23]. Specifically, REFIL [20] partitions agents into subgroups to capture shared local coordination patterns across tasks. UPDet [13] leverages a single unified Transformer architecture to handle varying entities across tasks. DT2GS [14] decomposes each task into multiple sub-tasks via latent variables generated by Transformer encoders. RIT [21] 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 [22], transition and reward functions [15], or task similarity measures [16].

The above research works focus on online multi-task MADRL. A related line of research [24, 25, 26] investigates offline multi-task MADRL which utilizes offline data across multiple independent tasks to learn generalized policies for unseen tasks. Among them, ODIS [24] identifies coordination skills for individual agents from states and joint actions in the offline data. HiSSD [26] adopts a hierarchical policy that jointly learns common and task-specific skills in an offline dataset. However, we consider online MADRL without assuming the availability of offline data or state information. Moreover, existing literature on both online and offline multi-task MADRL does not model communication and interactions among agents within tasks, which our work explicitly addresses.

Communication in MADRL Existing research 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 method 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 content of messages (what to communicate) is generally encoded information derived from either observations [27, 7, 28, 11, 9] or intended actions [5, 29, 6]. Specifically, TGCNet [9] leverages a Transformer encoder to encode local observations as messages. ATOC [5] learns an intention model that encodes both local observations and action intentions. IS [29] encodes imagined trajectories capturing agents’ future action plans. MAIC [6] uses a teammate model to predict teammates’ intentions and generate agent-specific messages. These approaches use an intention model during both training and execution. In contrast, our method learns messages that correlate with actions using a prediction network, which is used only in training.

The decision of whether to communicate or not can be determined based on confidence or distance measures [30, 31, 32], a binary classifier [7, 33, 8, 34], or a communication graph [10, 35, 36, 37, 38, 9]. Specifically, agents can communicate when their confidence is low, as in VBC [30], or choose not to communicate when the previous or estimated messages are similar to the current messages, as in TMC [31] and MBC [32]. Methods based on learnable binary classifiers, including IC3Net [7], GACML [33], I2C [8], and T2MAC [34], assign communication labels using a threshold during

training. Most similar to our work, attention mechanisms are often employed to construct communication graphs, enabling dynamic and flexible communication. Specifically, TarMAC [10] formulates communication as an attention-based message aggregation process. G2ANet [35] employs a two-layer attention mechanism for selective message aggregation. DGN [36] uses multi-head attention as a convolution kernel to aggregate information among neighboring agents. MAGIC [37] introduces a hard attention to dynamically construct communication graphs, and CommFormer [38] leverages attention to allocate credit to received messages within a bi-level optimization framework. More recently, TGCNet [9] learns a dynamic and directed communication graph using a multi-key gated mechanism with multiple hard attention modules. In contrast, our method also employs attention mechanisms but further integrates a threshold to prune unnecessary messages.

3 Preliminaries

We extend the definition of multi-task MADRL [19] by incorporating entities and communication.

Definition 1. An Entity-Based Multi-Task Multi-Agent Reinforcement Learning with communication problem \mathcal{MT} is defined as:

$$\mathcal{MT} := \{\mathcal{T}_k | k = 1, 2, \dots, K\}, \quad \mathcal{T}_k := \langle I, E, S, A, O, M, \Omega, P, R, \gamma \rangle$$

where each task \mathcal{T}_k 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} , actions \mathcal{A} , and messages \mathcal{M} 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 $I^k \subseteq \mathcal{I}$, $E^k \subseteq \mathcal{E}$, $\hat{E}^k \subseteq \mathcal{E}$, $S^k \subseteq \mathcal{S}$, $O^k \subseteq \mathcal{O}$, $A^k \subseteq \mathcal{A}$, and $M^k \subseteq \mathcal{M}$.

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 $m_i^k \sim f(o_i^k; \theta_{enc})$ parametrized by θ_{enc} . A joint policy parametrized by θ_π is then defined as $\pi(\mathbf{a}^k | \mathbf{o}^k, \mathbf{m}^k; \theta_\pi)$, where joint observations $\mathbf{o}^k = \langle o_1^k, \dots, o_N^k \rangle$ and joint messages $\mathbf{m}^k = \langle m_1^k, \dots, m_N^k \rangle$, with $N = |I^k|$ denoting the number of agents in task k . The goal of multi-task MADRL with communication is to learn both the encoder f and the policy π that maximize the average expected return across all K tasks:

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

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 $V(\mathbf{o}^k, \mathbf{m}^k; \phi)$ with parameters ϕ as:

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

4 Methods

In this section, we present Multi-task Communication Skills (MCS), a multi-task MADRL with communication method that learns and performs multiple tasks simultaneously. An overview of MCS architecture is shown in Figure 1. Each task consists of a set of agents that exchange messages through a learned communication graph. Based on received messages and local observations, agents select and execute actions in the environment. During training, experiences from all agents and tasks are collected jointly to enable centralized training. After training, agents deploy their learned policies in each individual task. MCS can handle communication with varying numbers of agents, observation spaces, and action spaces, thereby enabling generalization to multiple different environments. The entire framework, including communication, policy, and value networks, is trained end-to-end.

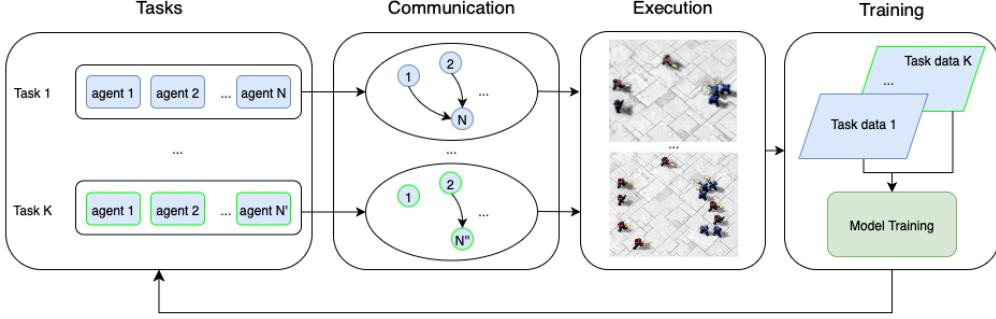


Figure 1: An overview of the MCS architecture. Agents communicate and act in each task, and data from multiple tasks are used to train a shared model across tasks.

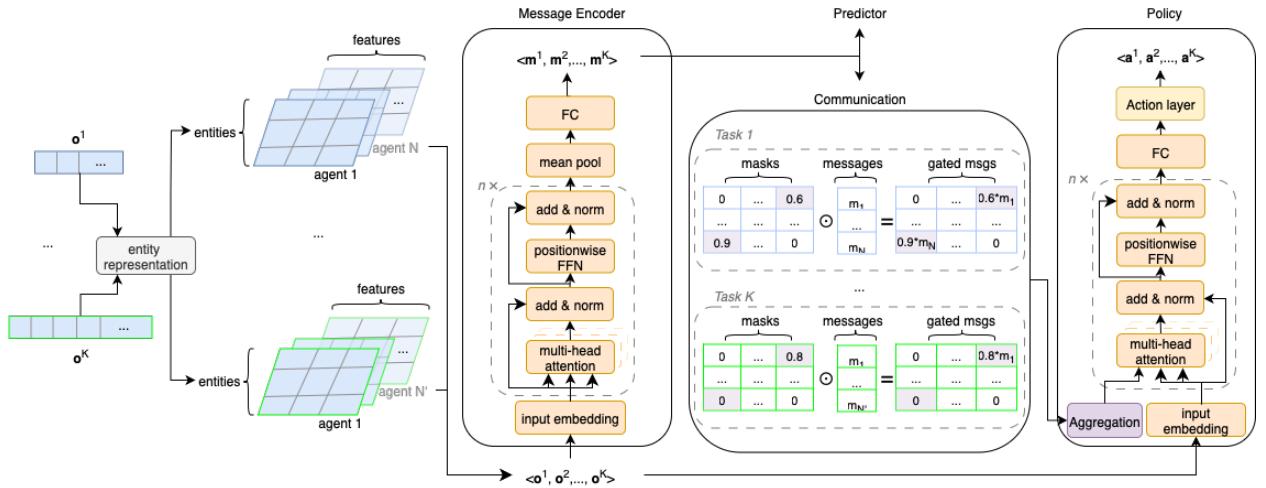


Figure 2: The network structure of MCS. Task-specific observations \mathbf{o}^k are represented in an entity-based form and then encoded into messages \mathbf{m}^k . During communication, messages are pruned using masks \mathbf{C}^k , applied through column-wise multiplication. Then, messages are aggregated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

4.1 Entity-based Communication in multi-task MADRL

We first introduce how messages are generated, communicated, and integrated into policies in MCS. The network structure used in MCS is schematically illustrated in Figure 2. We start with an entity-based representation across multiple tasks. Due to the varying dimensionality of the observation and action spaces, the network's inputs and outputs shall be aligned when learning a shared communication protocol and policy across tasks. In entity-based representations, the observation \mathbf{o}_i^k for each agent i in task k is represented as $\mathbf{o}_i^k \in \mathbb{R}^{|\hat{E}^k| \times D^e}$, where observable entities \hat{E}^k may vary across tasks. However, the feature dimensionality D^e for each entity remains constant across tasks (e.g., positions and velocity), which allows a shared input layer with dimensionality D^e to be used for message generation. Next, given entity-based observations in task k , we employ a Transformer-based message encoder f_{enc} , shared by N agents, to generate messages. Concretely, the message encoder f_{enc} 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 $\mathbf{m}_i^k = f_{enc}(\mathbf{o}_i^k; \theta_{enc}) \in \mathbb{R}^{D^m}$, where D^m is the dimensionality of messages that remains constant across tasks, and θ_{enc} represents the encoder parameters. As a result, the message encoder maps task- and agent-specific observations into a common message space \mathbb{R}^{D^m} , enabling the generalization of communication across agents and tasks. We denote the joint messages in task k as $\mathbf{m}^k = \langle \mathbf{m}_1^k, \dots, \mathbf{m}_N^k \rangle = \langle f_{enc}(\mathbf{o}_1^k; \theta_{enc}), \dots, f_{enc}(\mathbf{o}_N^k; \theta_{enc}) \rangle$. This can be abbreviated as $\mathbf{m}^k = f_{enc}(\mathbf{o}^k; \theta_{enc})$.

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. [9], 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 :

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

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:

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

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 $\tilde{\mathbf{m}}_j^k = \langle \tilde{m}_{1,j}^k, \dots, \tilde{m}_{N,j}^k \rangle = \mathbf{C}_j^k \odot \mathbf{m}^k$, where $\mathbf{C}_j^k = \langle C_{1,j}^k, \dots, C_{N,j}^k \rangle$ 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.

Notably, the number of received messages can vary across tasks, which changes the input size of the policy network. To address this, we design 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} , which can scale to a variable number of senders while capturing inter-agent dependencies. For each task k , the aggregation function produces an aggregated message \bar{m}_j^k upon received messages $\tilde{\mathbf{m}}_j^k$ (i.e., $\bar{m}_j^k = f_{agg}(\tilde{\mathbf{m}}_j^k)$) as follows:

$$\begin{aligned} h_{j,\ell}^k &= \text{GRU}(\tilde{m}_{\ell,j}^k, h_{j,\ell-1}^k), \quad \ell = 1, \dots, N \\ \bar{m}_j^k &= \frac{1}{N} \sum_{\ell} h_{j,\ell}^k \in \mathbb{R}^{D_h}, \end{aligned} \quad (3)$$

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

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 as the query embedding in a standard multi-head attention module within the Transformer-based policy network. Let $o_j^k \in \mathbb{R}^{|\hat{E}^k| \times D^e}$ denote the observation matrix for agent j in task k . For \tilde{H} heads with $d_{\text{head}} = D^h / \tilde{H}$, each head $\tilde{h} \in \{1, \dots, \tilde{H}\}$ computes:

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

where $W_{\mathcal{Q}}$, $W_{\mathcal{K}}$, $W_{\mathcal{V}}$, 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 derived from observations. The outputs from all heads are concatenated to form z_j^k , which enables receiver j to focus on the most relevant entity-based observations during communication. Then, z_j^k is used in the action layer to decide the receiver's actions. Note that the final action layer to generate task-specific actions follows Hu et al. [13] and Tian et al. [14]. 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:

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

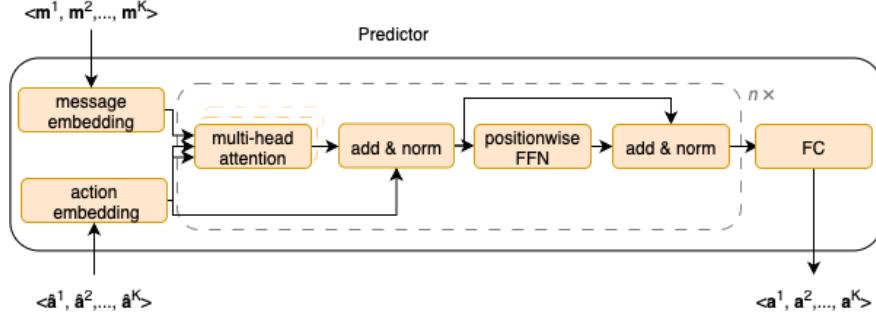


Figure 3: The network structure diagram of the predictor.

$$\begin{aligned} \mathcal{L}(\theta_{enc}, \theta_\pi, \phi) = \\ \frac{1}{K} \sum_k^K \frac{1}{N} \sum_i^N \mathbb{E}_{\mathbf{o}^k, a_i^k} [\log \pi(a_i^k | o_i^k, \bar{\mathbf{m}}_i^k; \theta_\pi) V(\mathbf{o}^k, \bar{\mathbf{m}}^k; \phi)] \end{aligned} \quad (5)$$

where $\bar{\mathbf{m}}^k = \langle \bar{m}_1^k, \dots, \bar{m}_N^k \rangle$ denotes the joint aggregated messages of agents, and $\bar{m}_i^k = f_{agg}(\mathbf{C}_i^k \odot f_{enc}(\mathbf{o}^k; \theta_{enc}))$. 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 perspective of the overall 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 prediction network q_{pred} that correlates the generated messages \mathbf{m}^k with the sender agents' actions \mathbf{a}^k , thereby enhancing coordinated action selection among agents. Concretely, we employ an additional Transformer decoder as the prediction network (Figure 3), which is used only during training and discarded at execution. Within multi-head attention, the prediction network takes received messages as query embeddings. The key and value embeddings are derived from initialized actions denoted as $\tilde{\mathbf{a}}$ (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 actions $\mathbf{a}^k \in A^k$ and messages $\mathbf{m}^k \in M^k$ 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) [39]. By replacing the conditional distribution of actions given messages with a variational distribution $q_{pred}(\mathbf{a}^k | \mathbf{m}^k)$, we have:

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

where $H(\mathbf{a}^k) \equiv -\mathbb{E}_{p(\mathbf{a}^k)} [\log p(\mathbf{a}^k)]$ 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 $\mathbb{E}_{p(\mathbf{a}^k, \mathbf{m}^k)} [\log q_{pred}(\mathbf{a}^k | \mathbf{m}^k)]$ using sampled batch data. Given samples

Algorithm 1 Multi-task Communication Skills

```

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

```

$\{(\mathbf{a}_b^k, \mathbf{m}_b^k, \mathbf{o}_b^k)\}_{k \in \{1, \dots, K\}, b \in \{1, \dots, B\}}$ for K tasks and B batch-size, we maximize the following log-likelihood:

$$\mathcal{L}_{pred}(\theta_{enc}) = \frac{1}{K} \frac{1}{B} \sum_{k=1}^K \sum_{b=1}^B \log q_{pred}\left(\mathbf{a}_b^k \mid f_{enc}(\mathbf{o}_b^k; \theta_{enc})\right) \quad (7)$$

The loss \mathcal{L}_{pred} backpropagates gradients from the predictor to the message encoder, encouraging it to generate messages that reflect the sender agents' behaviors and align agents' action selections.

4.3 The Overall Learning Objective

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

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

where β is a coefficient that balances the influence of the prediction network. \mathcal{L}_{MT} ensures that messages are both informative and beneficial for action selections, and that the policies and critics are learned based on communication. In particular, the messages are optimized not only through gradients backpropagated from the receiver agents' policy and critic networks, but also through the auxiliary prediction network.

We further introduce Algorithm 1 to illustrate how MADRL agents communicate and update their messages, policies, and critics in the multi-task setting. At each time step t of an episode, agents in task k observe $\mathbf{o}_t^k = \langle o_{1,t}^k, \dots, o_{N,t}^k \rangle$, which are used to generate messages \mathbf{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 $\bar{\mathbf{m}}_t^k$ at time step t (lines 7-9), which are then used to produce actions \mathbf{a}_t^k (line 10). The actions 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 a task-specific replay buffer (line 11). During training, mini-batches are sampled from these buffers (line 14). Lines 15–17 are then used to compute the losses for the policy, predictor, and critic in Equation 8. Concretely, the sampled observations are first used to generate messages, which are subsequently passed through the prediction network to produce an action distribution. Then, gradients are backpropagated through the critics, predictors, actors, aggregation function, and message encoders in an end-to-end manner (line 20).

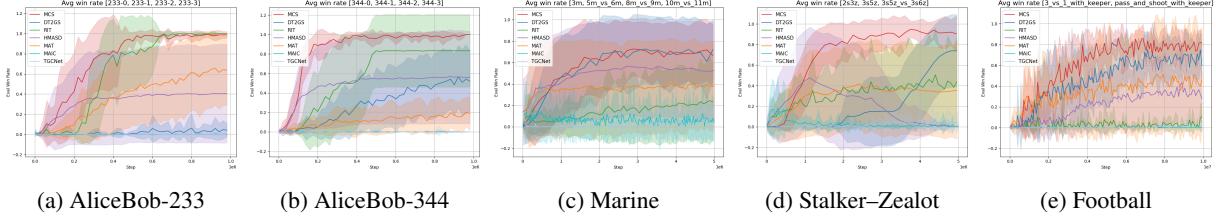


Figure 4: Averaged win-rate across multiple tasks on AliceBob (a–b), SMAC (c–d), and Football (e).

5 Experiments

We evaluate our proposed method in three challenging multi-agent environments: AliceBob [40], SMAC [41], and Google Research Football [18]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. [14], 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. In MCS, K tasks are learned and evaluated simultaneously.
- **Multi-task MADRL methods without communication.** DT2GS [14] and RIT [21] are two SOTA methods in this branch. DT2GS is a policy-based method which uses Transformers as an encoding of entity-based observations and decode actions for multiple tasks. RIT is a value-based method which uses padding techniques for unobservable entities for different tasks. We compare MCS with DT2GS and RIT to evaluate the benefit of communication under multi-task settings. Similar to MCS, K tasks are learned and evaluated simultaneously.
- **Single-task MADRL methods with communication.** TGCNet [9] is the most recent method that employs a multi-key gated attention mechanism. In contrast to MCS, TGCNet does not consider the correlation between messages and actions. MAIC [6] is another communication approach that promotes coordination by sharing messages correlated with other agents’ intended actions. Notably, neither TGCNet nor MAIC considers the multi-task setting. Instead, K tasks are learned and evaluated separately.
- **Single-task MADRL methods without communication.** MAT [42] is the SOTA method on many SMAC tasks, which employs a Transformer to encode observations and generate action sequences for agents. HMASD [40] represents the SOTA in leveraging latent skills for agent coordination. We compare MCS with MAT to evaluate the impact of our proposed Transformer-based communication mechanism, and with HMASD to assess the effectiveness of the prediction network in promoting coordination. Also, K tasks are learned and evaluated separately.

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 [14], we evaluate the averaged performance across K tasks, defined as $\text{Avg} = \frac{1}{K} \sum_k \text{WinRate}(k)$, where $\text{WinRate}(k)$ denotes the win rate on task k . The win rate $\text{WinRate}(k)$ is estimated by running several evaluation episodes for each task at fixed intervals during training, then averaging across episodes and tasks. We further show task-specific performance $\text{WinRate}(k)$ for each method, which demonstrates whether multi-task methods can outperform sing-task methods in each particular task. The hyper-parameters of each methods are either from published papers or fine-tuned for fair comparison. For MCS, we further fine-tune the threshold $\hat{\alpha}$ (Equation 2) and coefficient β (Equation 8), and we also analyze the sensitivity of these parameters on the learning performance. We provide a list of critical hyperparameters and the details of the entity-based representation for each environment in Appendix.

5.1 Evaluation Results

AliceBob Series The AliceBob environment, originally designed by Yang et al. [40], is created to explicitly demonstrate 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 $\{\text{red diamond}, \text{red button}\}$, $\{\text{blue diamond}, \text{blue button}\}$, $\{\text{pink diamond}, \text{pink button}\}$ with 2

²The source code will be released publicly on Github upon acceptance.

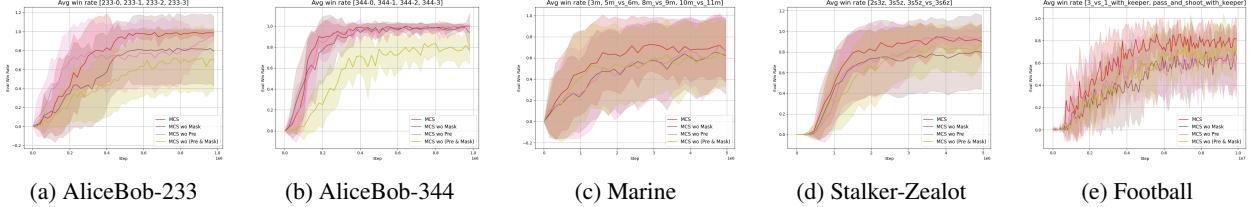


Figure 5: Ablation studies of MCS on AliceBob (a–b), SMAC (c–d), and Football (e).

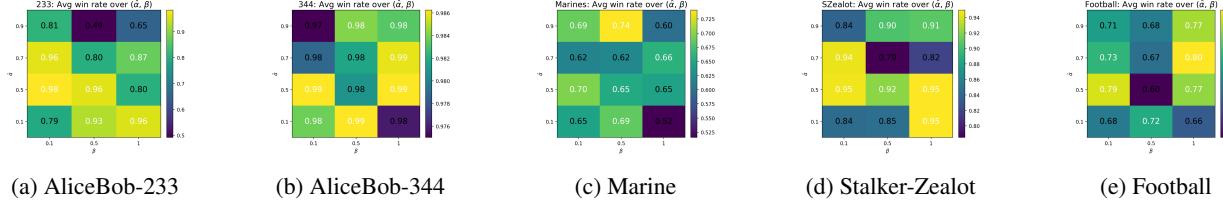
agents $\{Alice, Bob\}$. We denote this configuration as 233-0, indicating 2 agents, 3 diamonds/foods, and 3 buttons/keys with index 0. By varying entity types and associated colors (therefore different IDs), we obtain the AliceBob-233 series: {233-0, 233-1, 233-2, 233-4}. To examine scalability, we 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, 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 them to coordinate effectively to reach the correct and dynamically changing targets. Each agent perceives only the relative distances between its own position and other entities, and its surrounding grids. 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 show the averaged performance across tasks in AliceBob-233 and AliceBob-344 series in Figures 4a–4b, with results averaged over 6 random seeds. MCS shows faster convergence than other methods in AliceBob-233 series and significantly better performance than other methods in AliceBob-344 series. With communication, the Charlie agent in AliceBob-344 series can be used for better reaching the goals in MCS. In contrast, RIT, which applies a unified domain mask across tasks, achieves comparable performance in this environment, suggesting that it can effectively generalize knowledge between tasks. DT2GS, however, does not consider inter-agent coordination and therefore fails in the tasks. Other methods such as TGCNet and MAIC rely on training with single batch, which prevents them from efficiently exploiting successful episodes (where agents must collect all diamonds or food for a win). We also report per-task performance for both multi-task and single-task methods in the Appendix, where MCS consistently achieves much higher win rate across all tasks in AliceBob 344 series.

SMAC Series. The StarCraft Multi-Agent Challenge (SMAC) is a real-time strategy game serving as a benchmark in the MADRL community [41]. In SMAC, agents controlled by the learning algorithm must defeat all enemies, requiring effective cooperation strategies and fine-grained micro-control of movement and attack. Following Tian et al. [14], we construct two series of maps: the Marine series {3m, 5m_vs_6m, 8m_vs_9m, 10m_vs_11m} and the Stalker-Zealot series {2s3z, 3s5z, 3s5z_vs_3s6z}. In these settings, the number of agents varies across tasks, testing the scalability of multi-task methods. Moreover, the observation and action spaces differ across tasks, adding further complexity to learn a unified model.

Figures 4c–4d reports the average performance across the Marine and Stalker-Zealot series, with results averaged over 6 random seeds. On the Marine series, MCS achieves similar performance as DT2GS, and both MCS and DT2GS outperform other baselines, including the strong single-task method HMASD. On the Stalker-Zealot series, MCS significantly outperforms all the other methods. Notably, HMASD suffers a severe performance drop around 1M time steps, indicating convergence to a poor local optimum. RIT converges very quickly but with much lower win rates. DT2GS improves only after about 3M steps, which is much slower than MCS in this series. We further report per-task performance for both multi-task and single-task methods in the Appendix, where MCS consistently achieves the highest win rates across all tasks in the Stalker-Zealot series.

Football Series. Google Research Football (GRF) is a physics-based 3D soccer simulator for reinforcement learning [18], providing a challenging multi-agent benchmark environment with high stochasticity and sparse rewards. In GRF, opponents are controlled by expert agents, which significantly increase the complexity of the state–action space. To highlight the difficulty of the tasks, we train and evaluate using only the scoring reward: agents receive a reward of +1 when scoring a goal, which is sparse. Episodes terminate when a goal is scored, the ball goes out of bounds, or possession changes. To construct entity-based representations, features from the left and right teams are grouped 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 also included. The final entity representation remains comparable in size to the

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

original observations, which does not alter task difficulty. We construct a multi-task Football series using two maps: $\{3_vs_1_with_keeper, pass_and_shoot_with_keeper\}$, which involve different numbers of RL agents attempting to score from the edge of the field.

Figure 4e shows the averaged win rate across tasks in the Football series, with results further averaged over 4 random seeds. MCS achieves significantly a higher averaged win rate than all baseline methods. RIT fails to perform effectively in this high-dimensional, stochastic domain. The results of per-task performance are provided in the Appendix, where MCS obtains the highest win rate in $3_vs_1_with_keeper$ and achieves similar performance as MAT and DT2GS in $pass_and_shoot_with_keeper$.

5.2 Ablation Studies

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 can be seen from Figure 5, removing either communication mask (MCS wo Mask) defined in Equation 2 or the prediction network (MCS wo Pre) defined in Section 4.2 can lead to slower convergence or performance degradation. In all multi-task series, removing both communication mask and prediction network (MCS wo (Pre & Mask)) can lead to significantly lower win rate compared to MCS.

5.3 Hyperparameters Analysis

We also conducted a sensitivity analysis of the threshold $\hat{\alpha}$ and coefficient parameter β . We report the mean of the averaged win rate in the last 10 episodes across tasks under different values of $\hat{\alpha}$ and β in Figure 6. In all tasks, moderate values of $\hat{\alpha}$ (e.g., 0.5 or 0.7) yield the best results, whereas larger values (e.g., 0.9) tend to degrade performance. This indicates that limited communication may negatively affect the performance, while a moderate amount of communication is sufficient to achieve good performance. The effect of β is more subtle and task-dependent. In AliceBob and SMAC Stalker-Zealot series, both low values (e.g., $\beta = 0.1$) and high values (e.g., $\beta = 1$) can lead to good performance. However, in SMAC Marine and Football series, β must be carefully tuned in combination with $\hat{\alpha}$. We also observe that inappropriate combinations of $\hat{\alpha}$ and β , for example, $\hat{\alpha} = 0.9$ and $\beta = 0.5$ in the AliceBob 233 series, can lead to a drastic drop in performance. This typically occurs when communication is severely limited (high $\hat{\alpha}$) or when the message encoding is overly regularized (high β). Overall, the analysis suggests that a robust configuration across most tasks is $\hat{\alpha}=0.5$ and $\beta=0.1$, suggesting moderate amount of communication and slowly regularize the messages. In fact, in the finetuning stage, we adopt a strategy of adjusting $\hat{\alpha}$ around 0.5 to regulate communication, while starting with a small value of β and gradually increasing it.

5.4 Analysis of Message Representations

We further investigate the 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 [43] to project the high-dimensional latent representations into two dimensional points, as shown in Figure 7, where each point corresponds to an agent. In AliceBob, where different tasks differ only in the types 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, even though 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 prediction network that maximizes mutual information between messages and actions to promote coordinated action selection. 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, while the amount of communication required may vary across different environments. 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] 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.
- [2] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *CoRR*, abs/1610.03295, 2016.
- [3] 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*, pages 1–8, 2009.
- [4] Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *Int. J. Robotics Res.*, 32(11):1238–1274, 2013.
- [5] Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. In *Advances in Neural Information Processing Systems 31 (NIPS)*, pages 7265–7275, 2018.
- [6] 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 Twelfth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 9466–9474. AAAI Press, 2022.
- [7] 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.
- [8] 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, editors, *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020.
- [9] 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, editors, *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 23395–23403. AAAI Press, 2025.
- [10] 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)*, pages 1538–1546, 2019.
- [11] 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, editors, *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.

- [12] Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Trans. Knowl. Data Eng.*, 34(12):5586–5609, 2022.
- [13] 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.
- [14] 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, editors, *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.
- [15] 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.
- [16] 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*, pages 4733–4741. ijcai.org, 2024.
- [17] 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, editors, *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019*, pages 2186–2188, 2019.
- [18] 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*, pages 4501–4510. AAAI Press, 2020.
- [19] 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, editors, *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*, pages 2681–2690. PMLR, 2017.
- [20] 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*, pages 4596–4606. PMLR, 2021.
- [21] 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, 2025.
- [22] Lukas Schafer, Filippos Christianos, Amos Storkey, et al. Learning task embeddings for teamwork adaptation in multi-agent reinforcement learning [c/ol]. In *NeurIPS 2023 Workshop on Generalization in Planning*, 2023.
- [23] 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, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 9924–9935, 2019.
- [24] 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.
- [25] Jiayu Chen, Tian Lan, and Vaneet Aggarwal. Variational offline multi-agent skill discovery. *arXiv preprint arXiv:2405.16386*, 2024.
- [26] 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.
- [27] 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, editors, *Advances*

- in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2244–2252, 2016.
- [28] 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.
 - [29] 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.
 - [30] 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, editors, *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pages 3230–3239, 2019.
 - [31] 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, editors, *Advances in Neural Information Processing Systems 33 (NIPS)*, 2020.
 - [32] 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, editors, *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023, London, United Kingdom, 29 May 2023 - 2 June 2023*, pages 439–447. ACM, 2023.
 - [33] 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*, pages 5142–5149, 2020.
 - [34] 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, editors, *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*, pages 15154–15163. AAAI Press, 2024.
 - [35] 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)*, pages 7211–7218, 2020.
 - [36] Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu. Graph convolutional reinforcement learning. In *International Conference on Learning Representations*, 2020.
 - [37] 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)*, pages 964–973, 2021.
 - [38] 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.
 - [39] 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, editors, *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*, pages 5171–5180. PMLR, 2019.
 - [40] 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, editors, *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.
 - [41] 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, editors, *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS ’19, Montreal, QC, Canada, May 13-17, 2019*, pages 2186–2188. International Foundation for Autonomous Agents and Multiagent Systems, 2019.
 - [42] 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, editors, *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.
- [43] Leland McInnes and John Healy. UMAP: uniform manifold approximation and projection for dimension reduction. *CoRR*, abs/1802.03426, 2018.

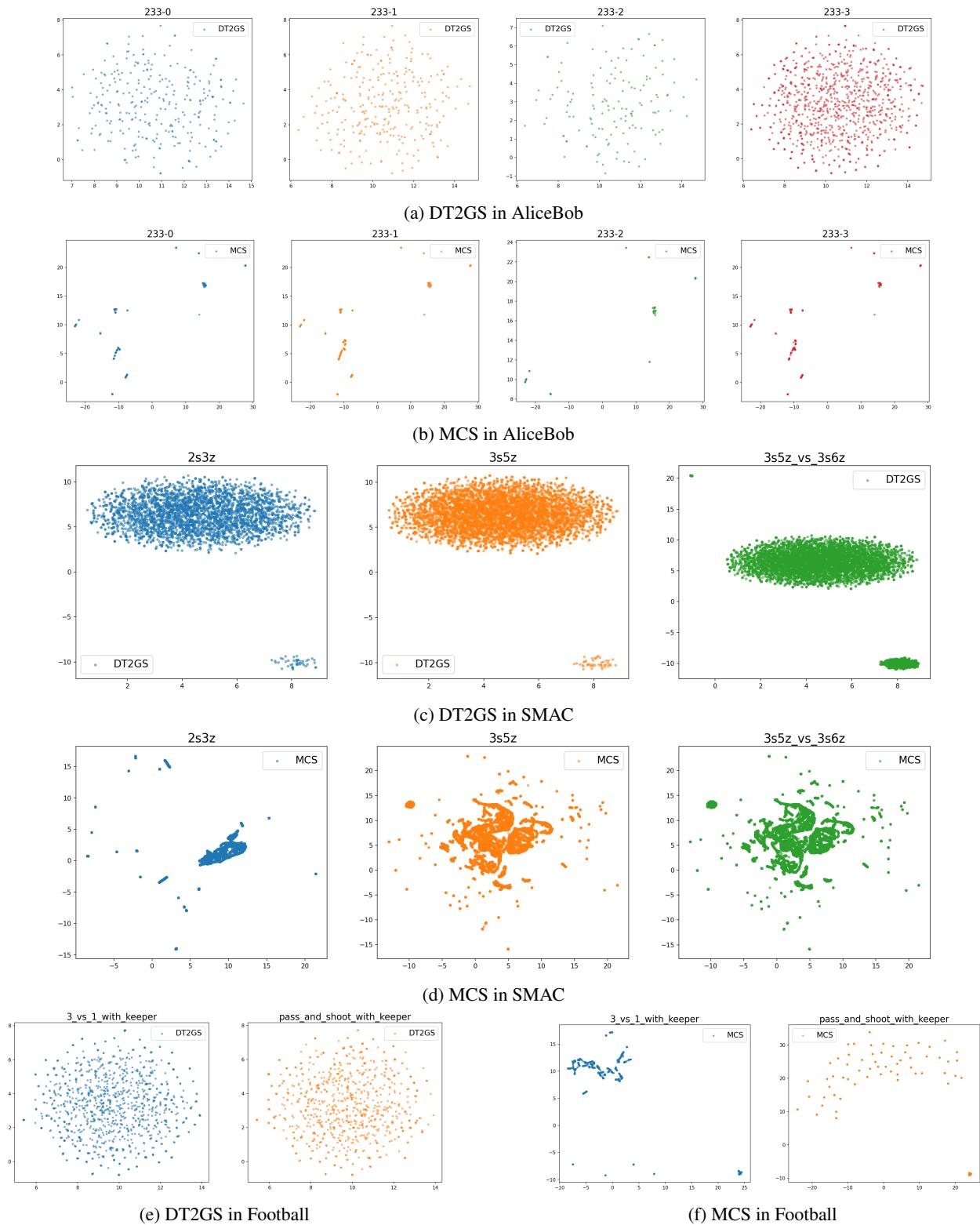


Figure 7: Latent representations of DT2GS and MCS.

A Details of Entity-based Representations

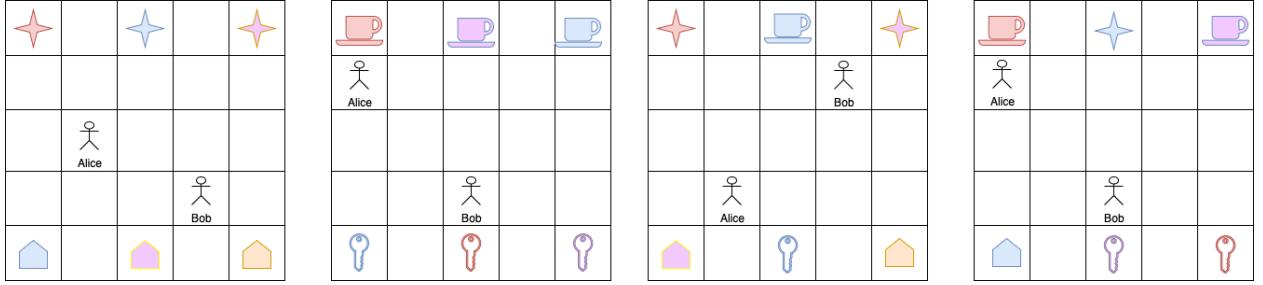


Figure 8: An illustration of the AliceBob-233 series. Agents, diamonds/food, and buttons/keys are randomly placed on the map in each episode.

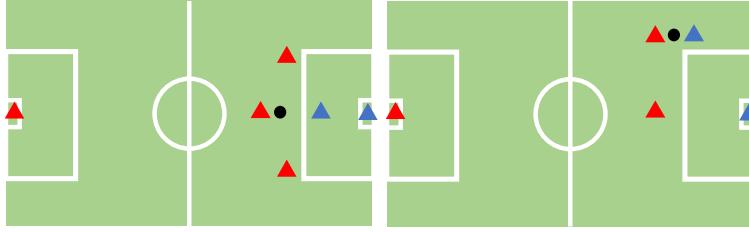


Figure 9: Football series with 3 vs 1 with keeper (Left) and Pass and shot with keeper (Right).

AliceBob series. The AliceBob environment was originally proposed for single-task MADRL without considering entity-based representations. Following the approaches of Hu et al. [13] and Tian et al. [14], 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). We provide an example of the multi-task setting in the AliceBob-233 series in Figure 8, where two agents must coordinate to collect 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 series. We use the same multi-task environment as used by Hu et al. [13] and Tian et al. [14], where entities consist of the agent itself, allied agents, and enemies. 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 considered³.

Football series. We adapt the Google Research Football environment [18] 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}`. 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

³Implementation available at: <https://github.com/Felixvillas/DT2GS>

Table 1: Default hyperparameters of MCS across different environments.

HYPERPARAMETER	ALICEBOB	SMAC	FOOTBALL
<i>lr</i>	5E-4	5E-4	5E-4
<i>critic_lr</i>	5E-4	5E-4	5E-4
<i>opti_eps</i>	1E-5	1E-5	1E-5
PPO EPOCHS	8	8	15
MINI-BATCHES	10	8	2
<i>clip_param</i>	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

Table 2: Hyperparameters ($\hat{\alpha}$ threshold and β) used by MCS in different multi-task series.

Parameters	AliceBob 233	AliceBob 344	SMAC Marine	SMAC Stalker Zealots	Football
$\hat{\alpha}$	0.5	0.7	0.9	0.5	0.5
β	0.1	1.0	0.1	0.5	0.1

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 Critical Hyperparameters and Per-task Performance

The hyperparameters used for MCS in AliceBob, SMAC, and Google Research Football are summarized in Table 1. As shown, the three environments share most hyperparameters, except for the number of PPO epochs and the mini-batch size. For SMAC, we use the same hyperparameters as in DT2GS, which have already been fine-tuned. For AliceBob, we adopt similar values but use a slightly larger mini-batch size to obtain more samples. For the Football environment, we use the same hyperparameters as those employed in the single-task setting. Consequently, these hyperparameters are also applied to the baseline methods (e.g., DT2GS and RIT) to ensure a fair comparison. We also report the communication threshold ($\hat{\alpha}$) and coefficient parameter (β) used by MCS in Table 2, which correspond to the results presented in Figures 4 and 5 of the manuscript.

The experiments were run in parallel on a cluster with 32 CPU cores and an NVIDIA A100 GPU. Table 3 reports the wall-clock training time for each method on each multi-task series; values are averaged over the tasks within a series. On 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). On SMAC (Marine and Stalker Zealot) and Football, runtimes spread more widely. RIT scales poorly (tens to > 100 h), suggesting unfavorable difficulty in complex environments. MCS remains reasonable on simpler tasks but is longer on Marines (23h). 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 AliceBob and Football, 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 a time, which may come at the expense of multi-task learning performance.

We further report the per-task performance of each method across all multi-task series, as shown in Figure 10 and Table 4. As illustrated in Figures 10a–10b, MCS consistently achieves faster convergence and significantly higher win rates across all tasks in the AliceBob environments. In SMAC Marines (Figure 10c), MCS outperforms DT2GS in the $3m$ and $5m_vs_6m$ tasks, while DT2GS surpasses MCS in $8m_vs_9m$ and $10m_vs_11m$, potentially due to the benefits of task decomposition in the two tasks. 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

Table 3: Average wall-clock runtime (**hours**) per method across tasks under each multi-task series.

Method	AliceBob 233 series	AliceBob 344 series	SMAC Marines	SMAC 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

lower overall average performance. In the Stalker Zealots series, MCS consistently outperforms all baselines in terms of learning performance. In Football, 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.

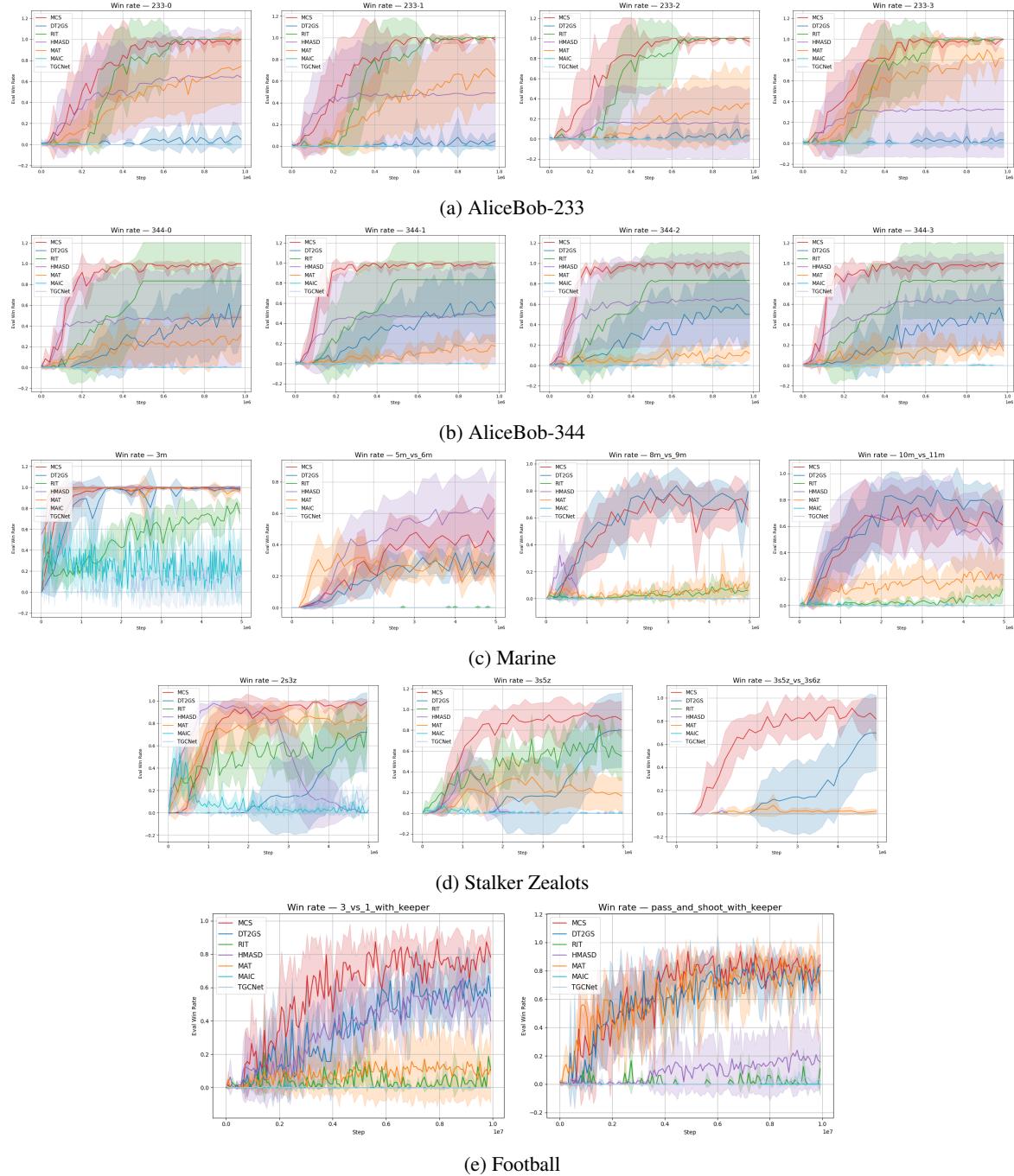


Figure 10: Per-task win rate curves across different environments: AliceBob-233/344, SMAC Marine/Stalker Zealots, and Football.

Table 4: Per-task win rates ($mean(std)$) for all benchmark multi-task series. Columns list individual tasks; the rightmost column is the average across tasks.

AliceBob 233 series					
Method	233-0	233-1	233-2	233-3	Avg
MCS	0.98 (0.02)	0.98 (0.02)	0.99 (0.01)	0.98 (0.01)	0.98 (0.00)
DT2GS	0.05 (0.09)	0.04 (0.08)	0.04 (0.06)	0.02 (0.06)	0.04 (0.01)
RIT	1.00 (0.00)	1.00 (0.01)	1.00 (0.00)	0.99 (0.01)	1.00 (0.00)
HMASD	0.64 (0.50)	0.48 (0.53)	0.16 (0.39)	0.32 (0.50)	0.40 (0.21)
MAT	0.70 (0.35)	0.64 (0.30)	0.32 (0.40)	0.82 (0.13)	0.62 (0.21)
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 series					
Method	344-0	344-1	344-2	344-3	Avg
MCS	0.98 (0.02)	0.99 (0.01)	0.99 (0.01)	0.98 (0.01)	0.99 (0.00)
DT2GS	0.49 (0.35)	0.56 (0.33)	0.55 (0.32)	0.47 (0.28)	0.52 (0.04)
RIT	0.83 (0.41)	0.83 (0.41)	0.83 (0.41)	0.83 (0.41)	0.83 (0.00)
HMASD	0.48 (0.52)	0.49 (0.53)	0.64 (0.50)	0.64 (0.49)	0.56 (0.09)
MAT	0.27 (0.24)	0.15 (0.11)	0.10 (0.06)	0.16 (0.05)	0.17 (0.07)
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)	0.01 (0.01)	0.00 (0.01)	0.00 (0.00)
SMAC Marine series					
Method	3m	5m_vs_6m	8m_vs_9m	10m_vs_11m	Avg
MCS	0.99 (0.00)	0.43 (0.20)	0.67 (0.11)	0.66 (0.08)	0.69 (0.23)
DT2GS	0.99 (0.02)	0.28 (0.03)	0.73 (0.12)	0.77 (0.16)	0.69 (0.29)
RIT	0.74 (0.07)	0.00 (0.00)	0.06 (0.04)	0.08 (0.05)	0.22 (0.35)
HMASD	1.00 (0.00)	0.60 (0.24)	0.00 (0.00)	0.55 (0.31)	0.54 (0.41)
MAT	0.98 (0.02)	0.26 (0.06)	0.09 (0.07)	0.23 (0.14)	0.39 (0.40)
MAIC	0.24 (0.23)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.06 (0.12)
TGCNet	0.13 (0.26)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.03 (0.07)
SMAC Stalker Zealots series					
Method	2s3z	3s5z	3s5z_vs_3s6z	Avg	
MCS	0.97 (0.02)	0.93 (0.15)	0.86 (0.12)	0.92 (0.06)	
DT2GS	0.53 (0.33)	0.63 (0.36)	0.48 (0.31)	0.55 (0.07)	
RIT	0.65 (0.19)	0.63 (0.16)	0.00 (0.00)	0.43 (0.37)	
HMASD	0.07 (0.16)	0.00 (0.00)	0.00 (0.00)	0.02 (0.04)	
MAT	0.83 (0.11)	0.21 (0.18)	0.02 (0.01)	0.36 (0.43)	
MAIC	0.03 (0.06)	0.00 (0.00)	0.00 (0.00)	0.01 (0.02)	
TGCNet	0.11 (0.07)	0.00 (0.00)	0.00 (0.00)	0.04 (0.06)	
Football series					
Method	3_vs_1_with_keeper	pass_and_shoot_with_keeper	Avg		
MCS	0.79 (0.12)	0.78 (0.07)	0.79 (0.00)		
DT2GS	0.58 (0.15)	0.74 (0.08)	0.66 (0.12)		
RIT	0.06 (0.03)	0.04 (0.04)	0.05 (0.02)		
HMASD	0.48 (0.12)	0.17 (0.26)	0.33 (0.22)		
MAT	0.10 (0.19)	0.81 (0.05)	0.45 (0.50)		
MAIC	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)		
TGCNet	0.00 (0.00)	0.03 (0.05)	0.01 (0.02)		