Multi-Task Multi-Agent Reinforcement Learning With Interaction and Task Representations

Chao Li, Shaokang Dong, Shangdong Yang, Yujing Hu, Tianyu Ding, Wenbin Li, and Yang Gao, *Senior Member, IEEE*

***Abstract*— Multi-task multi-agent reinforcement learning (MT-MARL) is capable of leveraging useful knowledge across multiple related tasks to improve performance on any single task. While recent studies have tentatively achieved this by learning independent policies on a shared representation space, we pinpoint that further advancements can be realized by explicitly characterizing agent interactions within these multi-agent tasks and identifying task relations for selective reuse. To this end, this article proposes Representing Interactions and Tasks (RIT), a novel MT-MARL algorithm that characterizes both intra-task agent interactions and inter-task task relations. Specifically, for characterizing agent interactions, RIT presents the interactive value decomposition to explicitly take the dependency among agents into policy learning. Theoretical analysis demonstrates that the learned utility value of each agent approximates its Shapley value, thus representing agent interactions. Moreover, we learn task representations based on per-agent local trajectories, which assess task similarities and accordingly identify task relations. As a result, RIT facilitates the effective transfer of interaction knowledge across similar multi-agent tasks. Structurally, RIT develops universal policy structure for scalable multi-task policy learning. We evaluate RIT against multiple state-of-the-art baselines in various cooperative tasks, and its significant performance under both multi-task and zero-shot settings demonstrates its effectiveness.**

***Index Terms*— Knowledge transfer, multi-agent reinforcement learning (MARL), multi-task learning, task representation.**

I. Introduction

MULTIAGENT reinforcement learning (MARL) has witnessed great advances in many cooperative tasks such as autonomous vehicles [1], navigation control [2], swarm robotics control [3], guaranteed display ads allocation [4], and packet routing [5]. However, much of this success is confined to the single task setting, where specialized agents are trained from scratch for each new task. This is prohibitively expensive when deploying agents to address real-world problems. In contrast, humans are adept at multi-tasking by efficiently learning and reusing shared coordination knowledge to collaboratively tackle a set of complex tasks [6]. Informed by this insight, multi-task multi-agent reinforcement learning (MT-MARL) [7] is proposed to provide multiple autonomous agents with an universal joint policy for cooperatively solving multiple multi-agent tasks. This learning paradigm is expected to leverage useful knowledge across multiple related tasks to improve performance over any single task, given the assumption that there exists some common knowledge across these tasks.

Here, two questions are naturally raised: how to characterize the common knowledge for efficient knowledge transfer across these tasks and how to identify task relations for addressing the negative interference [8]. This work presents an inductive bias for solving these two questions: it is necessary to characterize both interactions (i.e., the interactions among multiple agents) and tasks (i.e., the relationships between tasks) to learn an universal policy for efficiently solving multiple cooperative tasks. However, it remains unclear how to achieve this in the multi-agent field, where tasks may vary in their participating agents, observation spaces, state spaces, and other elements [9], [10], [11].

Recent studies [9], [10], [11], [12] have tentatively achieved knowledge transfer across different multi-agent tasks by learning independent policies on a shared representation space. In detail, these algorithms first utilize flexible structures (e.g., graph neural network [13], attention mechanism [14]) to align the observation spaces of different multi-agent tasks, and then learn multi-agent policies on this universal space, aiming for knowledge transfer. Although their promise in some certain contexts, these approaches typically overlook two critical problems that limit the generalization performance of their learned policies. The first problem is that these methods usually assume independent policies for agents, which fails to explicitly characterize agent interactions and therefore lacks efficient extraction of common interaction knowledge across tasks. The second one refers to the negative interference problem [8], which means that some knowledge beneficial to one task may be irrelevant (or even detrimental) to other tasks. Although existing algorithms aim to address this by constructing a shared representation space for multiple tasks and accordingly identifying task relations, they depend on the value functions as sole supervision signals, rendering them inefficient when task rewards are scarce. Thus, extra task information is necessitated to efficiently identify task relations and inform which knowledge should be shared.

Received 29 September 2023; revised 4 August 2024; accepted 1 October 2024. This work was supported in part by the National Science and Technology Major Project under Grant 2021ZD0113303; in part by the National Natural Science Foundation of China under Grant 62192783, Grant 62106100, and Grant 62206133; in part by Jiangsu Natural Science Foundation under Grant BK20221441; and in part by the Young Elite Scientists Sponsorship Program by CAST under Grant 2023QNRC001. *(Corresponding author: Yang Gao.)*

Chao Li, Shaokang Dong, Wenbin Li, and Yang Gao are with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China (e-mail: chaoli1996@smail.nju.edu.cn; shaokangdong@gmail.com; liwenbin@nju.edu.cn; gaoy@nju.edu.cn).

Shangdong Yang is with the School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China (e-mail: sdyang@njupt.edu.cn).

Yujing Hu is with the NetEase Fuxi AI Lab, NetEase Inc., Hangzhou 310052, China (e-mail: huyujing@corp.netease.com).

Tianyu Ding is with the Applied Sciences Group, Microsoft Corporation, Redmond, WA 98052 USA (e-mail: tianyuding@microsoft.com).

Digital Object Identifier 10.1109/TNNLS.2024.3475216

To overcome the above limitations, in this article, we propose Representing Interactions and Tasks (RIT), a novel MT-MARL algorithm that characterizes both agent interactions (common knowledge extraction) and task representations (task relations identification). For explicitly characterizing agent interactions, RIT transforms the coordinated action selections of all agents within each cooperative task into a sequence generation problem. In this process, all agents sequentially choose actions and each agent selects actions considering both its local history and actions selected by its preceding agents. As a result, the joint policy is factorized into dependent local policies, explicitly considering the dependency among agents and characterizing agent interactions. Furthermore, these dependent local policies serve as basis for interaction knowledge transfer across tasks.

Building upon this insight, RIT presents the interactive value decomposition to formalize this process, where each agent's utility function is conditioned on both its local history and the actions selected by its preceding agents. We provide theoretical confirm that the learned utility value of each agent constitutes an approximation to its Shapley value that measures its average marginal contribution across all permutations [15], [16], thus explicitly characterizing agent interactions. Structurally, we instantiate the utility function using a transformer-like structure to handle the varying dimensions of joint actions selected by preceding agents, observations, and states of different tasks. This enables universal and scalable multi-task policy learning.

For characterizing task relations, RIT learns task representations based on per-agent local trajectories in multiple tasks, and employs contrastive learning to update them. This allows for efficient measurement of task similarities at the level of agents' trajectories. Afterward, task representations are used as extra inputs to agents' policies, enabling positive interaction knowledge transfer across similar multi-agent tasks.

We compare our approach against multiple state-of-the-art baselines on Multi-agent Particle Environment (MPE) [17] and StarCraft Multi-Agent Challenge (SMAC) [18] benchmarks. The empirical results demonstrate that RIT achieves significant performance under both multi-task and zero-shot settings. Additionally, ablation studies confirm the benefits of our proposed interactive value decomposition and the utilization of learned task representations in improving generalization performance.

The primary contributions are summarized as follows.

1) We pinpoint that developing efficient MT-MARL algorithms across multiple cooperative tasks needs to explicitly characterize both agent interactions and task relations.
2) We accordingly propose a novel MT-MARL algorithm, named RIT, which introduces the interactive value decomposition to acknowledge inter-agent interactions and learns task representations to identify task relations.
3) Extensive experiments demonstrate that RIT consistently outperforms multiple state-of-the-art baselines under both multi-task and zero-shot settings. Ablation studies further verify the effectiveness of its components.

II. Related Work

In this section, we provide a comprehensive introduction to MARL algorithms about knowledge transfer. These algorithms are broadly divided into two categories: intra-task knowledge transfer and inter-task knowledge transfer. For the latter one, we particularly focus on achieving knowledge transfer among multiple tasks, as done by MT-MARL. Additionally, we offer a brief overview of MARL algorithms that adhere to the centralized training with decentralized execution (CTDE) paradigm.

A. Intra-Task Knowledge Transfer

Intra-task knowledge transfer primarily focuses on the single-task setting and communicates reusable knowledge among agents to improve their learning efficiency. The most common approach to achieve this is through parameter sharing, where all agents [19] or partial agents [20] share the same policy and update it using experiences from relevant agents. Additionally, SEAC [21] employs importance sampling to alleviate the internal distribution shift between transitions of different independent agents, enabling all agents to share their experiences. Other works aim to distill guiding value functions, advice, or policies to promote agents with poor performance [22], [23], [24]. However, these algorithms concerning intra-task knowledge transfer may face challenges in multi-agent tasks where complex coordinated behaviors are hard to emerge and valuable experiences are scarce. Obviously, we are more concerned with the efficient and effective transfer of common knowledge across different multi-agent tasks.

B. Inter-Task Knowledge Transfer

Inter-task knowledge transfer reuses relevant knowledge from source tasks to promote policy learning in target tasks [25]. Here, we particularly focus on MARL works that achieve knowledge transfer among multiple multi-agent tasks. Recent works [9], [10], [11], [12] explore the shared semantic structures existed in the observation spaces of different multi-agent tasks, and therefore advocate for learning a shared representation space across all tasks, based on some flexible structure (e.g., graph neural network [13] and attention mechanism [14]). Specifically, DyMA-CL [9] decomposes the observation of each agent into different semantic components and employs graph neural network to shape a shared representation space for curriculum-based multi-agent tasks, enabling direct experience reuse and policy initialization across them. EPL [10] introduces an evolutionary-based curriculum to scale up the number of agents and uses the attention mechanism to derive a population-invariant policy architecture. UPDeT [11] instantiates each agent's utility function with a transformer to align the observation spaces of different tasks, and proposes the policy decoupling mechanism to deal with their different action spaces via domain knowledge. Although their promise in certain tasks, these algorithms suffer from two drawbacks that severely limit their performance: 1) they often assume independent policies for agents, which fails to explicitly characterize agent interactions and thus lacks efficient extractions of common interaction knowledge across tasks and 2) they solely depend on the value functions to update the shared representation space, which may lead to inefficiency in identifying task relations when task rewards are scarce. Recently, REFIL [12] tries to characterize local interactions by randomly partitioning all entities within each task into two disjoint subgroups, and encourages each agent to coordinate in arbitrary imagined sub-groups, which intuitively makes the learned policies robust. However, the inconsistency between random

sub-group partitions and the underlying ground-truth coordinated behaviors introduces inaccuracies to the optimization of agents' policies, degrading the capability of characterizing agent interactions. Additionally, the drawback 2) still exists for REFIL.

C. CTDE-Based MARL Algorithms

There are two major categories of MARL algorithms under the CTDE paradigm. One is multi-agent policy gradient algorithms [26], [27], [28], [29], [30], [31] (e.g., MADDPG, COMA, MAPPO, and HAPPO), which learns a centralized critic conditioned on both sates and all agents' joint actions to update their decentralized policies. This centralized critic alleviates the non-stationarity problem [32] and instructs agents' joint policy to converge to the global optimum. The other is value decomposition algorithms [33], [34], [35], [36], [37] (e.g., VDN, QMIX, QTRAN, and QPLEX), which learns a factored global action value function to optimize all agents' utility functions (which correspond to their decentralized policies). This factorization needs to adhere to the individual-global-max (IGM) principle [35], which ensures the consistency regarding optimal joint actions and tractable decentralized selections of them.

It is evident that efficient knowledge transfer across multiple tasks benefits agent's capability of solving single complex task in comparison to single-task learning. Thus, this work aims for positive interaction knowledge transfer among similar tasks by explicitly characterizing agent interactions and identifying task relations. Additionally, we develop our algorithm following the CTDE paradigm to alleviate the non-stationarity problem.

III. Preliminary

In this section, we formalize task settings considered in this work and introduce value decomposition algorithms. Additionally, we provide a brief introduction to the Shapley value.

A. Factored Dec-POMDP

To construct multiple similar cooperative multi-agent tasks, we first pre-define an entity set $\mathcal{V}$ (including both agents and non-agents) and an agent set $\mathcal{K} \subset \mathcal{V}$. Subsequently, we arbitrarily sample an agent subset $\mathcal{N} \subset \mathcal{K}$ and a non-agent subset $\mathcal{U} \subset (\mathcal{V}/\mathcal{K})$ to create a specific multi-agent task T. By sampling various combinations of agent and non-agent subsets, we generate a number of multi-agent tasks that adhere to the same task distribution. Finally, we divide them into training tasks and testing tasks, respectively, to evaluate the algorithms.

For each specific cooperative multi-agent task, we consider a factored decentralized partially observable Markov decision process (Factored Dec-POMDP), which can be described as a tuple $\langle \mathcal{E}, \mathcal{N}, S, \boldsymbol{A}, \mathcal{P}, \mathcal{R}, \boldsymbol{Z}, \boldsymbol{O}, \gamma \rangle$. Here, $\mathcal{E} = \mathcal{N} \cup \mathcal{U} = \{e^1, e^2, \ldots, e^m\}$ represents the entity set, which includes both agent set $\mathcal{N}$ and non-agent set $\mathcal{U}$ (e.g., landmarks, rule-based or uncontrollable agents). $\mathcal{N} = \{1, 2, \ldots, n\}$ is the agent set, and $\boldsymbol{A} = \{A^1 \times A^2 \times \cdots \times A^n\}$ denotes the joint action space of all agents. $S = \{S^{e^1} \times S^{e^2} \times \cdots \times S^{e^m}\}$ is the factored state space, where S^{e^j} is the local state space of entity $e^j \in \mathcal{E}$.

At each time step t, each entity $e^j \in \mathcal{E}$ receives its local observation $o_t^{e^j} \in Z^{e^j} \in \boldsymbol{Z}$ according to its local observation function $O^{e^j}(o_t^{e^j}|s_t) \in \boldsymbol{O}$. Specifically, for each agent $i \in \mathcal{N}$, we define its local observation o_t^i as: $o_t^i = \{s^{e^j}|\mu_t(i, e^j) = 1, \forall e^j \in \mathcal{E}\}$, where $\mu_t(i, e^j)$ denotes a binary observable mask that indicates whether agent i can observe entity e^j at current time step t and $\mu_t(i, i)$ always equals to 1. After all agents select the joint action $\boldsymbol{a_t}$, the environment transits to the next state s_{t+1} according to the transition function $\mathcal{P}(s_{t+1}|s_t, \boldsymbol{a_t})$, and provides all agents with the same reward r_t according to the reward function $\mathcal{R}(s_t, \boldsymbol{a_t})$, where $\boldsymbol{a_t} = \{a_t^1, a_t^2, \ldots, a_t^n\}$ is the joint action of all agents. $\gamma \in [0, 1)$ is a discount factor. To deal with the partial observability challenge, each agent i conditions its policy π^i on its own action-observation history $\tau_t^i = \{o_1^i, a_1^i, o_2^i, a_2^i, \ldots, o_t^i\}$. The goal of all agents is to learn the optimal joint policy $\boldsymbol{\pi}^* = \{\pi^{1,*}, \pi^{2,*}, \ldots, \pi^{n,*}\}$ that maximizes the cumulative discounted rewards $\mathbb{E}_{\boldsymbol{\pi}, \mathcal{P}}[\sum_{t=0}^{\infty} \gamma^t r_t]$.

Assumption: We assume that all agents have the same local action spaces and all entities have the same local state spaces. Furthermore, we relax the decentralized execution constraint, and assume that agents in each task sequentially select actions adhering to a specific decision order, where each agent is able to observe the actions selected by its preceding agents.

B. Value Decomposition Algorithms

This work learns policies for agents based on their utility functions, similar to value decomposition algorithms. Specifically, the value decomposition algorithms factorize the global action value function $Q^{\text{total}}(\boldsymbol{\tau}, \boldsymbol{a})$ into per-agent utility function $Q^i(\tau^i, a^i)$ that is conditioned on each agent i's action-observation history τ^i and action a^i. This factorization needs to satisfy the IGM principle

$$\arg\max_{\boldsymbol{a}} Q^{\text{total}}(\boldsymbol{\tau}, \boldsymbol{a}) = \begin{pmatrix} \arg\max_{a^1} Q^1(\tau^1, a^1) \\ \arg\max_{a^2} Q^2(\tau^2, a^2) \\ \cdots \\ \arg\max_{a^n} Q^n(\tau^n, a^n) \end{pmatrix}$$

where $\boldsymbol{\tau}$ represents the joint action-observation histories of all agents and $\boldsymbol{a} = (a^1, a^2, \ldots, a^n)$ is the joint action. Under the IGM principle, the greedy selection over the joint action space can be computed in a tractable way by locally maximizing the per-agent utility function. Specifically, QMIX adheres to this principle by proposing the monotonicity condition

$$\frac{\partial Q^{\text{total}}(\boldsymbol{\tau}, \boldsymbol{a}, s)}{\partial Q^i(\tau^i, a^i)} \geq 0, \ \forall i \in \mathcal{N}.$$

QMIX introduces a mixing network f to calculate the joint action value function: $Q^{\text{total}}(\boldsymbol{\tau}, \boldsymbol{a}, s) = f(Q^1, Q^2, \ldots, Q^n, s)$, where the state s serves as additional inputs. Specifically, the parameters of f are generated by several hyper-networks that take the state s as inputs, and the weights of f are restricted to be positive to satisfy the monotonicity condition. As a result, all agents' utility functions can be updated using the temporal difference (TD) loss of the global action value function.

C. Shapley Value

Shapley value is a popular solution concept that distributes payoff for agents in a grand coalition. For a given characteristic function game $G = (\mathcal{N}, v)$, $\mathcal{N} = \{1, 2, \ldots, n\}$ denotes a finite agent set and $v : 2^{|\mathcal{N}|} \to \mathbb{R}$ is a characteristic function that maps each coalition $C \subseteq \mathcal{N}$ to its value $v(C)$. Let $\prod_{\mathcal{N}}$ denote the set of all permutations over $\mathcal{N}$. Given a permutation $l \in \prod_{\mathcal{N}}$, we denote by $\mathcal{S}_l(i)$ the set of all predecessors

of an agent i in l. The marginal contribution of agent i with respect to the permutation l is denoted by $\Delta_l^i(G) = v(\mathcal{S}_l(i) \cup \{i\}) - v(\mathcal{S}_l(i))$. Then, the Shapley value of agent i is given by [15] and [16]

$$\phi^i(G) = \frac{1}{|\mathcal{N}|!} \sum_{l \in \prod_{\mathcal{N}}} \Delta_l^i(G).$$

Obviously, each agent's Shapley value measures its average marginal contribution for participating in the coalitions formed by its predecessors across all permutations of $\mathcal{N}$, thus serving as **a precise characterization of agent interactions**. However, accurately calculating the Shapley value is intractable in practice. In Section V, we theoretically confirm that each agent's utility function learned by our algorithm can be regarded as an approximation to its Shapley value, validating the effectiveness of our algorithm in characterizing agent interaction knowledge.

IV. Methodology

In this section, we present a comprehensive introduction to our algorithm, RIT. We first explicitly characterize agent interactions by proposing the interactive value decomposition. Then, we use a transformer-like structure to enable universal and scalable multi-task policy learning. Afterward, we learn task representations to identify task relations. Finally, we give a summary of our algorithm.

A. Interactive Value Decomposition

RIT explicitly characterizes the interactions between agents by focusing on their coordinated action selections. Specifically, RIT transforms the coordinated action selections of all agents in each cooperative multi-agent task as a sequence generation problem. In this process, agents select their actions adhering to a specific decision order, and each agent selects actions based on both its local action-observation history and the actions selected by its preceding agents. This action selection process introduces dependency among agents and factorizes the joint policy into dependent local policies. For each agent, its policy is conditioned on both its local history and the actions selected by its preceding agents, which evaluates whether to select its coordinated actions for collaboration with predecessors. As a result, these dependent local policies explicitly represent agent interactions, and can serve as basis for interaction knowledge transfer across tasks. Therefore, we use this conditional policy factorization as the basis of RIT.

Building upon this insight, we propose the interactive value decomposition to follow this conditional policy factorization, which is consistent with the assumption of sequential action selection in Section III-A. In detail, for a n-agent cooperative task, we begin by assigning agents with an arbitrary decision order $\mathcal{I} = \{I^1, I^2, \ldots, I^n\}$ through random permutation at the beginning of each episode, where I^k denotes the k-th agent that selects actions. For each agent $I^k \in \mathcal{I}$, we define its utility function (corresponding to its local policy) as follows:

$$Q^{I^k} = Q^{I^k}\left(\tau^{I^k}, a^{<I^k}, a^{I^k}\right) \tag{1}$$

where $a^{<I^k}$ denotes the joint action selected by agents indexed lower than agent I^k. τ^{I^k} and a^{I^k}, respectively, represent current agent I^k's action-observation history and action. This utility function intuitively measures the profit of agent I^k selecting the local action a^{I^k} to collaborate with its preceding agents, after observing their selected actions. With it, each agent I^k learns a conditional policy $\pi^{I^k}(a^{I^k}|\tau^{I^k}, a^{<I^k})$. Based on the chain rule factorization of joint probability, we establish the relationship between the joint policy and agents' local policies

$$\boldsymbol{\pi}(\boldsymbol{a}|\boldsymbol{\tau}) = \prod_{k=1}^{n} \pi^{I^k}\left(a^{I^k}|\tau^{I^k}, a^{<I^k}\right) \tag{2}$$

where $\boldsymbol{\pi}(\boldsymbol{a}|\boldsymbol{\tau})$ represents the joint policy of all agents. Based on this conditional policy factorization formalism, RIT explicitly considers the dependency among agents, and accordingly characterizes agent interactions using their dependent policies.

1) Complexity Analysis: RIT assumes a sequential decision paradigm where all agents sequentially select actions. For each agent, it still selects actions in its local action space. And all agents still search for the optimal joint policy over their joint action space. Further, different from HAPPO [29] adhering to the sequential update paradigm, RIT simultaneously trains all agents' policies without extra training complexity. These three are the same as works that adhere to decentralized execution.

The increased complexity of RIT mainly comes from two aspects: 1) the selection regarding agents' joint actions and 2) the dimensions of per-agent policy inputs. For 1), existing works with decentralized execution enable simultaneous action selections of all agents, and form the joint action with $\mathcal{O}(1)$ time complexity. In contrast, RIT outputs the joint action with $\mathcal{O}(n)$ time complexity, where all agents need to select actions sequentially and n denotes the number of agents.

For 2), RIT learns dependent local policies that take current agent's action-observation history, action and actions selected by its preceding agents as inputs. Thus, the input space scales linearly with the number of agents and actions, and deep neural networks can generalize well across such space.

2) Universal Policy Structure: One potential issue is that it remains prohibitively expensive to respectively approximate the utility functions for all agents due to their distinct inputs. Furthermore, different multi-agent tasks often feature varying participating agents, different observation spaces, state spaces, and other elements. These differences further necessitate that the learned multi-task policy be universal to any agents within the decision order and any multi-agent tasks.

As is stated above, we recast all agents' coordinated action selection into a sequence generation problem. The same also applies to the calculations of their utility functions. Therefore, the utility function calculations for different agents and tasks can be regarded as sequences with different lengths. Motivated by the fact that the transformer [14] is able to handle sequences with arbitrary lengths, we employ a transformer-like structure to instantiate the utility functions of all agents.

Specifically, for each agent I^k, we employ the model shown in Fig. 1(a) to calculate the utility function Q^{I^k}. The model is comprised by two components: (a1) the encoder module that extracts shared representations among multiple tasks, and (a2) the decoder module that outputs the utility functions. In detail, for each task, the encoder module takes the state s as inputs and extracts fixed-length representations $\hat{e}^{I^k}$ by a multi-head attention module. In Section III-A, we make assumptions that all entities have the same local state spaces, and the global states are the concatenation of all entities' local states. And for each agent, its local observations are comprised by local

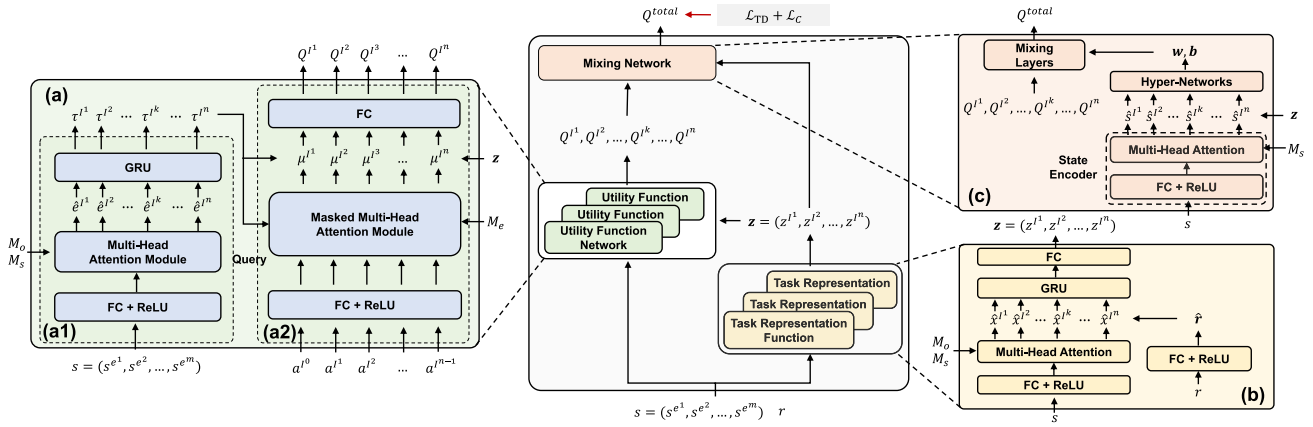


Fig. 1. Architecture of RIT. (a) Utility function network. It is comprised by two components: (a1) encoder module and (a2) decoder module. (b) Task representation function shared by all agents. (c) Mixing network. Note that we include agents' selected action at the last time step into their local states, and thereby their action-observation histories can be encoded using a GRU based on the state inputs and additional masks.

states of entities within its observable field. Therefore, the model that is capable of extracting fixed-length representations can adapt to tasks with different observation and state spaces. Additionally, we introduce a partially observable mask $\mathcal{M}_o$ to ensure that all agents' representations $\hat{e}^{I^k}$ conform to the partially observable property, and a scenario mask $\mathcal{M}_s$ to indicate padded agents in different tasks. After extracting these partially observable representations, we use a GRU to encode each agent I^k's local action-observation history τ^{I^k}. For achieving convenient batch data processing, we employ padded agents in this work. More details can be found in Appendix A-A, A-B, and A-E.

Within the decoder module, we begin with an indicator a^{I^0} and the first agent I^1 selects its action a^{I^1} based on its utility function $Q^{I^1}(\tau^{I^1}, a^{I^1})$. Then, the second agent I^2 receives the action a^{I^1} and calculates its utility function $Q^{I^2}(\tau^{I^2}, a^{I^1}, a^{I^2})$, as well as selecting its action a^{I^2}. The remaining agents follow this process in an auto-regressive formalism to calculate their utility functions and select actions. To deal with distinct action inputs in agents' utility functions, we utilize a masked multi-head attention module, where each agent's action-observation history τ^{I^k} outputted by the encoder module serves as the query. Furthermore, we employ a sequence mask $\mathcal{M}_e$ to ensure that each agent can only observe the actions selected by its preceding agents during training, similar to the transformer [14]. More details are provided in Appendix A-C and A-F.

B. Task Representation Learning

One remaining problem is the negative interference, where indiscriminately transferring knowledge among tasks can degrade the overall performance when there are conflicts between them. This problem can be alleviated by efficiently identifying task relations [39]. However, constructing a shared representation space by the multi-head attention module may exhibit inefficiency in capturing task relations when task rewards are scarce. To address this limitation, complementary task-related information should be explored to achieve positive interaction knowledge transfer among similar multi-agent tasks.

We achieve this by learning task representations to acquire task-related information, and thus characterize task relations. Instead of measuring similarities between tasks from the perspective of collective multi-agent systems (e.g., state transition function and reward function), we propose to measure the local similarities between tasks by extracting task representations that focus on the trajectories of agents in multiple tasks, using contrastive learning. The intuition behind this is that there are usually significant differences between different tasks from the perspective of collective systems. In contrast, the same policy may generate similar local state distributions in very different multi-agent tasks. In such scenarios, partial knowledge transfer is permitted and local similarities between tasks are promising.

Specifically, we learn per-agent task representations based on the augmented action-observation-reward history. This augmented history $\hat{\tau}^{I^k}$ of each agent I^k contains its current local observation $\tau_t^{I^k}$, action $a_{t-1}^{I^k}$, and additional rewards r_{t-1} at last time step $t-1$, which enables decentralized identification of current task from the view of each agent. Subsequently, we introduce a task representation function shared by all agents $f^\omega : \hat{\tau}^{I^k} \to z^{I^k}$ to infer task representations for them. This function takes each agent's augmented history $\hat{\tau}^{T,I^k}$ in task T as inputs, and outputs its local representation z^{T,I^k} of task T, sampled from the task distribution $\mathcal{T}$. Given multiple agents' trajectories from different tasks, we randomly sample pairs of task representation (z^1 and z^2) and shape a binary label y that indicates whether the two representations belong to the same task. Then, we define the contrastive loss as follows:

$$\mathcal{L}_C = \mathbb{E}_{(z^1,z^2,y)\sim\mathcal{D}} \operatorname{CrossEntropy}\big(g\big(z^1, z^2\big); y\big) \tag{3}$$

where we consider two task representations of agents from the same tasks as positive pairs ($y = 1$) and the rest as negative pairs ($y = 0$). We introduce a discriminator g to distinguish between positive and negative pairs as an auxiliary objective to optimize the task representation function f^ω. In this case, mapping two agent trajectories from different tasks to the same representations will increase the loss, therefore, be avoided.

We instantiate the task representation function by the model illustrated in Fig. 1(b). We again use a multi-head attention module to extract fixed-length partially observable representations $\hat{x}^{I^k}$ for each agent I^k, based on the state inputs s and auxiliary masks $\mathcal{M}_o, \mathcal{M}_s$. Subsequently, we concatenate these representations with $\hat{\boldsymbol{r}}$ of rewards r from the previous time step and feed this concatenation into a GRU to encode each agent's augmented history. Finally, we output task representations for all agents and optimize this representation function with TD loss and the aforementioned contrastive loss.

After acquiring task representations, we feed them into utility functions as additional inputs. These task representations can identify task relations and inform which interaction knowledge can be shared across tasks, enabling positive interaction knowledge transfer among similar tasks.

C. Overall Learning Objective

After calculating all agents' utility functions, we estimate the global action value function Q^{total} as a weighted summation of them, which is defined as follows:

$$Q^{\text{total}}(\boldsymbol{\tau}, \boldsymbol{a}, s, \boldsymbol{z}) = \sum_{I^k \in \mathcal{I}} k^{I^k}(s, \boldsymbol{z}) Q^{I^k}\left(\tau^{I^k}, a^{<I^k}, a^{I^k}\right) + b(s, \boldsymbol{z}) \tag{4}$$

where $k^{I^k} \geq 0$ and b are generated by hyper-networks whose inputs are states and all agents' task representations.

As shown in Fig. 1(c), we specialize a mixing network to achieve this calculation. The mixing network is comprised by a state encoder and several hyper-networks. Specifically, we employ the state encoder to map the state s into representations $(\hat{s}^{I^1}, \hat{s}^{I^2}, \ldots, \hat{s}^{I^n})$, the number of which is consistent with the number of agents within different tasks. This consistency is achieved by a multi-head attention module and the scenario mask. Then, these representations together with all agents' task representations $\boldsymbol{z} = (z^{I^1}, z^{I^2}, \ldots, z^{I^n})$ are used as inputs of hyper-networks to generate weights and biases of the mixing layers. Finally, we calculate the global action value function, and update all components by minimizing the total loss below

$$\mathcal{L} = \mathcal{L}_{\text{TD}}(\theta, \sigma, \omega) + \mathcal{L}_C(\omega, \phi). \tag{5}$$

In (5), θ denotes the parameters of the utility function network. ω and ϕ, respectively, represent parameters of the task representation function f^ω and the auxiliary discriminator g. σ are parameters of the mixing network (including the state encoder and hyper-networks). $\mathcal{L}_{\text{TD}}$ is the TD loss of the global action value function, which is defined as follows:

$$\mathcal{L}_{\text{TD}}(\theta, \sigma, \omega) = \left(r + \gamma \max_{\boldsymbol{a}'} \bar{Q}^{\text{total}}(\boldsymbol{\tau}', \boldsymbol{a}', s', \boldsymbol{z}') - Q^{\text{total}}(\boldsymbol{\tau}, \boldsymbol{a}, s, \boldsymbol{z})\right)^2 \tag{6}$$

where $\bar{Q}^{\text{total}}$ denotes a periodically updated target network for stable training. We use the TD loss to optimize all components of RIT except the auxiliary discriminator. And the contrastive loss $\mathcal{L}_C$ defined in (3) is used to update both the auxiliary discriminator and the task representation function.

As presented in Algorithm 1, the training procedure of RIT is as follows. For multi-task setting, we sample task from the set of training tasks and randomly permute the decision order at the beginning of each episode. At each time step, each agent infers the task representation and calculates its utility function, as well as selecting action in an auto-regressive formalization. When training occurs, RIT samples batches of episodic data from the replay buffer to calculate the loss defined in (5), and updates all components of it. This continues until some terminal conditions are reached (e.g., maximum episodes).

Additionally, we also investigate the zero-shot generalization performance of RIT on unseen testing tasks. For zero-shot setting, we sample tasks from the set of testing tasks and load the multi-task policies trained by RIT for agents. Then, we roll out several episodes to evaluate RIT's zero-shot performance.

Algorithm 1 RIT Training Procedure

```
Initialize necessary hyperparameters
Initialize parameters θ, σ, ω, φ of all components
if Multi-task setting then
    for Each episode do
        Sample task from the training task set
        Randomly permute the decision order 𝓘
        foreach agent I^k ∈ 𝓘 do
            Infer the task representation z^{I^k}
            Calculate the utility function Q^{I^k}
            Select its local action a^{I^k}
    Store the episode into the replay buffer 𝒟
    if train then
        Sample batches of episodes D̄ ~ 𝒟
        Calculate loss ℒ based on (5)
        Update all components with gradient descent
if Zero-shot setting then
    Sample task from the testing task set
    Load trained policy and roll out several episodes
```

V. Theoretical Analysis

In this section, we theoretically confirm that per-agent utility function learned by RIT can be regarded as an approximation to its Shapley value. We begin by a n-agent cooperative game. First, we formalize the definition of Shapley value in a Dec-POMDP, and describe the relationship between the Shapley values of all agents and the global reward.

Definition 1 (Shapley Value in a Dec-POMDP): Fix a Dec-POMDP $\langle \mathcal{E}, \mathcal{N}, S, \boldsymbol{A}, \mathcal{P}, \mathcal{R}, \boldsymbol{Z}, \boldsymbol{O}, \gamma \rangle$, and let s_t denote the environmental state at each time step t. After all agents select the joint action $\boldsymbol{a_t} = (a_t^1, a_t^2, \ldots, a_t^n)$, the environment transits to the next state s_{t+1} and returns the shared global reward r_t.

At each time step t, let $\prod_{\mathcal{N}}$ denote the set of all permutations over $\mathcal{N}$. Given a specific permutation $l \in \prod_{\mathcal{N}}$, we denote by $\mathcal{S}_l(i)$ the set of all predecessors of an agent i in l. The marginal contribution of agent i with respect to the permutation l is denoted by

$$\Delta_l^i(s_t, \boldsymbol{a_t^l}) = Q^{\boldsymbol{\pi}^{\mathcal{S}_l(i)\cup\{i\}}}\left(s_t, \boldsymbol{a_t}^{\mathcal{S}_l(i)\cup\{i\}}\right) - Q^{\boldsymbol{\pi}^{\mathcal{S}_l(i)}}\left(s_t, \boldsymbol{a_t}^{\mathcal{S}_l(i)}\right) \tag{7}$$

Then, the Shapley value of agent i is given by

$$\phi^i(s_t, \boldsymbol{a_t}) = \frac{1}{|\mathcal{N}|!} \sum_{l \in \prod_{\mathcal{N}}} \Delta_l^i(s_t, \boldsymbol{a_t^l}).$$

Specifically, in a Dec-POMDP, for specific state s_t, agents' joint action $\boldsymbol{a_t}$ and agents' joint policy $\boldsymbol{\pi}$ at any time step t, there exists a specific characteristic function game as defined in Section III-C. Their major difference is that the value of any coalition $C \subseteq \mathcal{N}$ is defined as $Q^{\boldsymbol{\pi}^C}(s, \boldsymbol{a}^C)$, where $\boldsymbol{a}^C = (a^i)_{i \in C}$ denotes the joint action of members in the coalition C, and $Q^{\boldsymbol{\pi}^C}$ represents the global reward earned by the coalition C. Obviously, $Q^{\boldsymbol{\pi}^C}$ measures the value of the coalition C.

Note that each agent's Shapley value is the average marginal contribution over all permutations of $\mathcal{N}$. Therefore, in (7),

for a specific permutation l, agent i's marginal contribution is conditioned on the state s and **ordered joint action**. Under this definition, *the permutation over agents is characterized by the order over their actions*. To estimate the marginal contribution defined in (7), we directly model a function to approximate it as done in [38], such that

$$\hat{\Delta}_l^i\left(s_t, \boldsymbol{a}_t^{\mathcal{S}_l(i)\cup\{i\}}\right) \rightarrow \Delta_l^i\left(s_t, \boldsymbol{a}_t^l\right). \tag{8}$$

Here, the approximating function $\hat{\Delta}_l^i(s_t, \boldsymbol{a}_t^{\mathcal{S}_l(i)\cup\{i\}})$ is conditioned only on joint actions of $\mathcal{S}_l(i)\cup\{i\}$ due to irrelevance to others' actions. Then, for calculating agent i's Shapely value, we need to enumerate all permutations of $\mathcal{N}$, and respectively approximate agent i's marginal contribution in them. Followed by this insight, agent i's Shapely value can be rewritten as:

$$\phi^i(s_t, \boldsymbol{a}_t) = \mathbb{E}_{l\sim P(l|\prod_{\mathcal{N}})}\Delta_l^i\left(s_t, \boldsymbol{a}_t^l\right) \tag{9}$$

where $P(l|\prod_{\mathcal{N}})$ is the permutation sampling probability.

To enable (9) to be tractable, we can employ a sample-based approach to approximate each agent i's Shapley value. Also, by substituting $\hat{\Delta}_l^i(s_t, \boldsymbol{a}_t^{\mathcal{S}_l(i)\cup\{i\}})$ for $\Delta_l^i(s_t, \boldsymbol{a}_t^l)$, we can approximate each agent i's Shapley value as follows:

$$\phi^i(s_t, \boldsymbol{a}_t) \approx \frac{1}{M}\sum_{l=1}^{M}\hat{\Delta}_l^i\left(s_t, \boldsymbol{a}_t^{\mathcal{S}_l(i)\cup\{i\}}\right),\ \forall l \sim \mathbb{E}_{l\sim P(l|\prod_{\mathcal{N}})} \tag{10}$$

where M denotes the number of sampling permutations.

Before presenting our proof, based on the definition of the Shapley value in a Dec-POMDP, we also extend the efficiency property of the Shapley value to the Dec-POMDP as follows:

Lemma 1 (Efficiency): All agents' Shapley values satisfy the efficiency property, i.e., $\sum_{i\in\mathcal{N}}\phi^i(s_t, \boldsymbol{a}_t) = r_t$.

Based on above interpretations, we present the proposition about each agent's utility function learned by RIT as follows.

Proposition 1: Each agent i's utility function $Q^{I^k=i}$ learned by RIT is an approximation to its Shapley value ϕ^i.

Proof: At first, RIT randomly permutes agents' decision order at the beginning of each episode, and agents follow this order throughout the episode. This operation finishes $M = 1$ sample of random permutations from the entire agent permutation set.

Subsequently, each agent i's utility function $Q^{I^k=i}$ learned by RIT follows a similar formation as $\hat{\Delta}_l^i$ in (10), which is conditioned on both its local action-observation history $\tau_t^{I^k}$ (an approximation to the state s) and the selected actions $a^{<I^k}$ of its preceding agents, besides its local action a^{I^k}. Thus, we regard the utility function as the approximating function $\hat{\Delta}_l^i$. Furthermore, the transform-like policy structure corresponds to an universal function that is capable of approximating each agent's marginal contribution $\hat{\Delta}_l^i$ within different permutations l, which enables scalable learning.

Finally, in practice, the characteristic functions of different coalitions are inaccessible. As done in (4) and (6), the linear summation of all agents' utility functions (corresponding to the approximating functions) are restricted to equal to the global reward (i.e., the value of the grand coalition), which makes agents' utility functions satisfy the efficiency property.

As a result, each agent's utility function learned by RIT can be regarded as an approximation to each agent's Shapley value, and thus the proof is completed. □

VI. Experiments

In this section, we design experiments to answer the following four questions: 1) can RIT benefit the multi-task performance? (see Section VI-C); 2) can RIT exhibit superior zero-shot generalization performance? (see Section VI-D); 3) if so, which component of it contributes the most to its performance gain? (see Section VI-E); and 4) can the task representations identify the local similarities between different tasks at the level of agents' trajectories? (see Section VI-F).

To address the first two questions, we use MPE and SMAC as the testbeds, and evaluate RIT under both multi-task and zero-shot settings. For question 3), we conduct ablation studies to validate the effectiveness of each component in RIT. Additionally, we visualize the learned task representations of different tasks via t-SNE to answer question 4).

A. Environments

MPE and SMAC are two commonly used benchmarks in the single-task MARL community. Building upon them, we customize some task sets for conducting multi-task learning and zero-shot generalization. We detail them for more clarity.

1) Multi-Agent Particle Environment: In MPE, we customize three task sets including cooperative navigation series, predator and prey series, and predator and prey series pro. Below is a brief description of each.

1) *Cooperative Navigation Series:* In this task set, n agents aim to occupy n unmovable landmarks while avoiding collisions with each other. Agents are globally rewarded according on the proximity of the closest agent to each landmark (sum of the minimum distance) and the total number of collisions (-1 for each collision). We customize the tasks for n ranges from 2 to 6, with training tasks including (2, 2), (3, 3), (4, 4), which denote the respective numbers of agents and landmarks. The remaining tasks, (5, 5), (6, 6), serve as testing tasks.
2) *Predator and Prey Series:* This task set is comprised by m predators and n preys competing, and two landmarks serving as obstacles. Compared to predators, preys move and accelerate faster, aiding their escape. To ensure full cooperation, we only control the predators while the preys follow a hard-coded heuristic: move toward the furthest sampled position from the nearest predator. If any one of the predators has collided with the preys, a shared team reward of +10 is emitted otherwise 0. We denote the tasks by (m, n), and the training tasks consist of (2, 1), (3, 1), (3, 2). The testing tasks include (4, 2), (5, 2).
3) *Predator and Prey Series Pro:* According to predator and prey series, we introduce a more challenging extension of it where the team reward of +10 is emitted only when any two predators simultaneously collide with the same prey otherwise 0. For advancing this hunting, we present extra rewards that penalize the predators when they are far from the closest preys. Except these changes, all other settings remain identical to predator and prey series.

2) StraCraft Multi-Agent Challenge: In SMAC, we use the task sets introduced by REFIL [12], which focus on the multi-task setting. We further divide the original maps in each task set into both training and testing maps for multi-task and zero-shot settings. Below, we briefly introduce these task sets.

TABLE I

PAYOFF MATRIX GAMES OF THE MULTI MATRIX GAME TASK SET. WE DENOTE THE COLLECTIVE REWARDS AT THE FIRST AND SECOND TIME STEPS BY A/B (E.G., −12/−12, −6/−12), AND AGENTS WIN BY SELECTING THE OPTIMAL JOINT ACTION THAT RESULTS IN A +100 REWARD

A^1 \ A^2	a^1	a^2	a^3
a^1	-12/-12	-12/-12	-12/**100**
a^2	-12/-12	-12/-12	-12/-12
a^3	-12/-12	-12/-12	-12/-12

A^1 \ A^2	a^1	a^2	a^3
a^1	-6/-12	-6/-12	-6/-12
a^2	-6/-12	-6/-12	-6/-12
a^3	-6/-12	-6/-12	-6/**100**

A^1 \ A^2	a^1	a^2	a^3
a^1	0/-12	0/**100**	0/-12
a^2	0/-12	0/-12	0/-12
a^3	0/-12	0/-12	0/-12

A^1 \ A^2	a^1	a^2	a^3
a^1	6/**100**	6/-12	6/-12
a^2	6/-12	6/-12	6/-12
a^3	6/-12	6/-12	6/-12

1) *3–8m:* Two symmetric teams of m marines fight against each other in this task set where m ranges from 3 to 8, which results in six tasks (maps) in total.
2) *3–8sz:* This task set pits symmetric teams against each other, with each team comprising three–eight agents, resulting in a total of 39 tasks (e.g., $3s$, $1s2z$, $2s1z$, $3z$, $4s$, $3s1z$), where s and z, respectively, denote the stalker and zealot.
3) *3–8MMM:* This task set provides two symmetric teams with medivacs (range from 0 to 2) and marines/marauders (range from 3 to 6), resulting in 66 tasks in total.
4) *3–8csz:* Similarly, 3–8csz assigns colossi (range from 0 to 2) and stalkers/zealots (range from 3 to 6) to each one of two symmetric teams, yielding 66 tasks (e.g., $3s$, $1c3s$, $2c3s$, $1s2z$, $1c1s2z$, $2c1s2z$, $2s1z$, $1c2s1z$, $2c2s1z$).

Specifically, we consider the first 80 percentile of all maps in each task set as training tasks and the rest as testing tasks. In addition, we present a didactic matrix game as follows.

1) *Multi Matrix Game:* This task set consists of four two-step cooperative matrix games for two agents. As shown in Table I, agents win by selecting the optimal joint actions to receive a +100 reward in each game. However, these games possess identical state spaces, requiring agents to infer which game they are in through interactions at the first time step. This task set is exclusively used under the multi-task setting to demonstrate the necessity of extra task-related information for effective multi-task learning.

B. Baselines

We compare RIT with multiple state-of-the-art MT-MARL algorithms. Below, we give a brief introduction to them.

1) *UPDeT-MIX and UPDeT-VDN:* Both of them instantiate the utility function of each agent with a transformer (only the encoder) to learn an universal policy, the only difference between them being their mixing functions [11].
2) *REFIL:* REFIL aims to extract shared local coordination patterns among agents from multiple multi-agent tasks by random sub-group partitioning, and demonstrates superior performance under the multi-task setting [12].
3) *QMIX-ATTN:* We equip QMIX [34] with the multi-head attention mechanism to handle different multi-agent tasks.
4) *VDN-ATTN:* We incorporate a multi-head attention module into VDN [33] to learn universal policies.
5) *ROMA:* Vanilla ROMA [40] aims to learn emergent roles in the single-task setting. In this work, we extend it with the multi-head attention mechanism to evaluate whether the roles enable superior knowledge transfer across tasks.

TABLE II

HYPERPARAMETERS OF RIT ACROSS ALL TASK SETS

Hyper-parameters	Multi Matrix Game	MPE	SMAC
epsilon_start ϵ_{max}	1.0	1.0	1.0
epsilon_finish ϵ_{min}	0.05	0.05	0.05
epsilon_anneal_time $T_{epsilon}$	50000	50000	500000
n_rollout_threads	1	8	8
buffer_size N_{max}	5000	5000	5000
batch_size N_{batch}	32	64	32
env_representation_dim	32	20	32
T_{max}	50000	5050000	10000000
lr	0.0005	0.001	0.001
mixing_embed_dim	32	32	32
hypernet_embed_dim	64	64	64
optimizer	RMSprop	Adam	Adam
attention_dim	N/A	64	64
n_heads	N/A	1	2
target_update_interval	200	200	200

6) *RIT-VDN (Ours):* We remove the mixing network in RIT, and calculate the global action value function by directly summing up all agents' utility functions. This variant is considered a separate baseline algorithm.
7) *MAPPO:* MAPPO [28] extends PPO [41] into the multi-agent scenario, and exhibits significant single-task performance. To evaluate its performance under multi-task and zero-shot settings, we adapt this algorithm for each task set by pre-defining the maximum possible inputs.
8) *MAT:* This algorithm follows the same sequential decision paradigm and employs the transformer to instantiate per-agent actor and critic, alongside a monotonic performance improvement guarantee [31]. We also pre-define the maximum possible inputs for it to evaluate its multi-task and zero-shot performance in all task sets.

C. Multi-Task Evaluation

We begin by evaluating all algorithms under the multi-task setting. Specifically, for MPE and SMAC benchmarks, we set eight parallel threads, and respectively equip them with randomly sampled training tasks at the beginning of each episode. For multi matrix game, we use 1 thread. All comparative results under this setting is carried out with five different random seeds. We set the maximum training steps T_{max} to 50 000, 5 050 000, and 10 000 000, respectively, for the multi matrix game task set and the task sets of MPE and SMAC benchmarks. During training, every T_{eval} steps, we conduct evaluations by rolling out n episodes and calculating the average episodic rewards (or win rates). More detailed settings about the hyperparameters of RIT on these task sets can be found in Table II.

Fig. 2 shows the multi-task performance of all algorithms on four task sets: multi matrix game, cooperative navigation series, predator and prey series, and predator and prey series pro. In the multi matrix game, one can observe that only RIT and RIT-VDN successfully solve all matrix games, achieving a

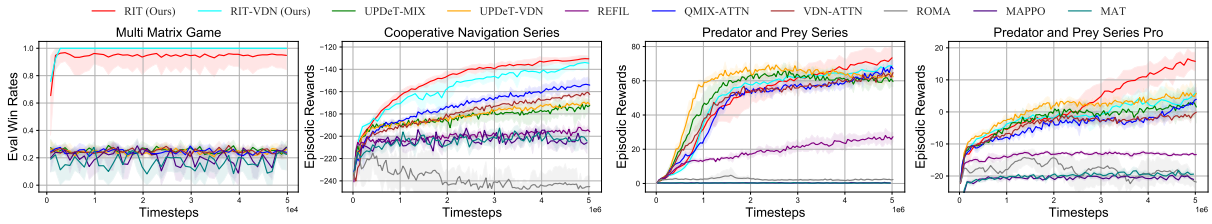


Fig. 2. Multi-task performance of all algorithms on four task sets including multi matrix game, cooperative navigation series, predator and prey series, and predator and prey series pro. All results are averaged across five different random seeds.

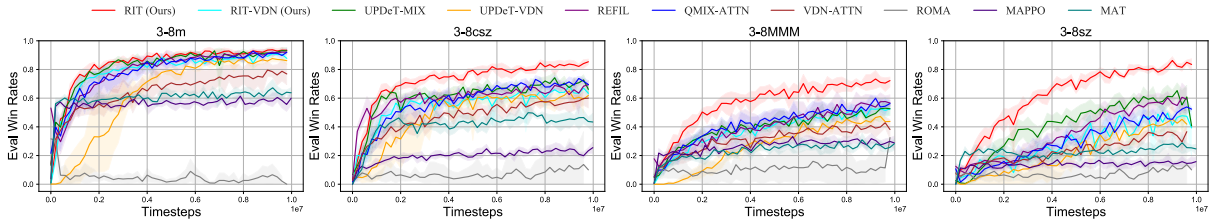


Fig. 3. Multi-task performance of all algorithms on four task sets including 3–8m, 3–8sz, 3–8MMM, and 3–8csz. The solid line denotes the median test win rates across five different random seeds and the shaded areas represent the standard errors.

1.0 win rate. And RIT-VDN exhibits more stable performance in comparison to RIT in this simple scenario. In contrast, other baselines fail to learn efficient multi-task policies and achieve low win rates of about 0.25, indicating that they solely solve single matrix games. This demonstrates that extra task-related information is necessitated to learn efficient multi-task policies and RIT achieves this by learning task representations.

In three task sets of the MPE benchmark, it is evident that RIT significantly outperforms other baselines in the cooperative navigation series (where RIT-VDN performs comparably) and delivers superior episodic rewards in the other two. On the contrary, other algorithms such as UPDeT-MIX, UPDeT-VDN, QMIX-ATTN, and VDN-ATTN suffer from slow convergence rates and suboptimal performance due to their inefficiency in explicitly characterizing agent interactions and task relations. In addition, REFIL suffers from poor performance, indicating that the discrepancy between its random sub-group partitioning and the ground-truth coordinated behaviors introduces inaccuracies to the policy optimization, degrading its performance.

MAPPO and MAT also get stuck in suboptimal outcomes in the cooperative navigation series, and fail to learn effective policies that yield positive episodic rewards in the other two. Beyond their inefficiency in achieving multi-task learning, this failure can be attributed to the sample inefficiency inherent in the on-policy paradigm. ROMA fails to extract efficient roles in a multi-task setting and thus performs poorly.

We also evaluate these algorithms on four task sets designed in the SMAC benchmark. As presented in Fig. 3, it is obvious that RIT demonstrates significant performance on all four task sets. Several baselines, including UPDeT-MIX, QMIX-ATTN, and REFIL also achieve pretty performance. These algorithms update agents' utility functions by using a mixing network to implicitly characterize agent interactions, and devise universal policies within a shared representation space. This approach enables preliminary knowledge transfer across multiple tasks, as validated by their performance.

In contrast, RIT-VDN, UPDeT-VDN, and VDN-ATTN suffer from low win rates on these task sets. This may be attributed to the limited representational capacity regarding the global action value function caused by the additivity condition. We hypothesis that the mixing network is capable of distributing agents' utility functions, and accordingly implicitly characterize agent interactions. Similar to the results in the MPE task sets, MAPPO and MAT again exhibit sub-optimal performance due to their inefficiencies in achieving multi-task policy learning and reusing off-policy data. Additionally, ROMA fails to learn effective policies to defeat enemies, indicating that useful roles are difficult to learn in a multi-task setting.

D. Zero-Shot Evaluation

In order to further evaluate the generalization capability of our algorithm to unseen tasks, we conduct experiments under the zero-shot setting on both MPE and SMAC benchmarks. In detail, we run eight parallel threads and sample from the testing tasks for each at the beginning of each episode. Then, we load the trained policies (five models corresponding to five random seeds from the multi-task evaluation), and roll out k episodes to evaluate the zero-shot performance of these trained policies.

Specifically, we set k to 64 for the MPE task sets (cooperative navigation series, predator and prey series, and predator and prey series pro), and 160 for all task sets 3–8m, 3–8csz, 3–8MMM, and 3–8sz) in the SMAC benchmark. For the task sets in the MPE benchmark, after rolling out k episodes for all five trained policies, we calculate the average episodic reward over these k episodes and further calculate the mean and standard error of average episodic rewards of the five trained policies. As for the task sets in the SMAC benchmark, we calculate the average eval win rates, as well as the mean and standard error of the average eval win rates of all trained policies.

Table III presents the zero-shot performance of all algorithms on three MPE task sets. Obviously, RIT achieves the highest episodic rewards on the unseen testing tasks of the cooperative navigation series, while other baselines perform poorly. This result demonstrates that RIT successfully characterizes agent interactions and transfers them to similar tasks, thus resulting in superior generalization performance than other algorithms.

However, in the remaining two task sets (predator and prey series, predator and prey series pro), all algorithms receive

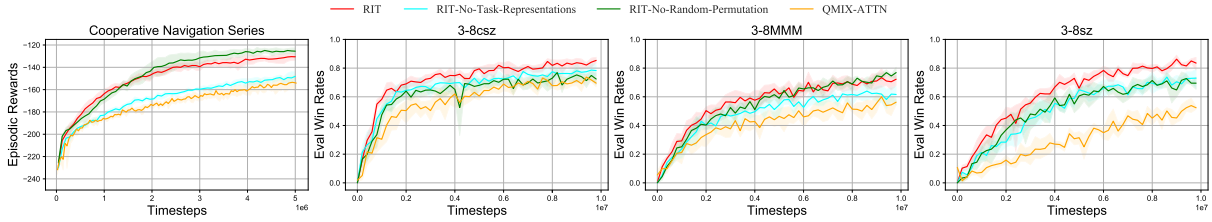


Fig. 4. Ablation studies under the multi-task setting.

TABLE III
ZERO-SHOT PERFORMANCE OF ALL ALGORITHMS IN THE MPE TASK SETS

Algorithm	Cooperative Navigation Series	Predator and Prey Series	Predator and Prey Series Pro
RIT (Ours)	**-342.4±14.1**	6.5±1.5	-52.2±5.4
RIT-VDN (Ours)	-376.1±17.9	5.6±1.3	**-49.9±3.4**
REFIL	-493.0±18.8	4.4±1.3	-54.0±3.9
UPDeT-MIX	-454.8±22.6	**6.9±1.8**	-55.7±2.6
UPDeT-VDN	-459.8±20.4	6.6±0.7	-53.9±4.5
QMIX-ATTN	-418.5±20.2	5.5±0.9	-55.8±4.3
VDN-ATTN	-415.1±18.6	3.8±1.4	-52.1±3.2
ROMA	-1081.6±303.8	5.3±1.2	-57.9±3.7
MAPPO	-478.0±7.7	1.1±0.6	-95.1±3.3
MAT	-524.0±29.0	1.2±0.3	-98.6±5.6

TABLE IV
ZERO-SHOT PERFORMANCE OF ALL ALGORITHMS IN THE SMAC TASK SETS

Algorithm	3-8csz	3-8MMM	3-8sz	3-8m
RIT (Ours)	**0.70±0.07**	**0.54±0.07**	**0.59±0.08**	**0.84±0.06**
RIT-VDN (Ours)	0.50±0.07	0.25±0.03	0.26±0.07	0.80±0.07
REFIL	0.61±0.03	0.35±0.08	0.34±0.12	0.82±0.05
UPDeT-MIX	0.61±0.07	0.27±0.05	0.39±0.09	0.81±0.08
UPDeT-VDN	0.49±0.04	0.20±0.01	0.18±0.04	0.68±0.04
QMIX-ATTN	0.61±0.03	0.27±0.01	0.23±0.03	0.83±0.05
VDN-ATTN	0.46±0.09	0.16±0.06	0.13±0.02	0.72±0.10
ROMA	0.11±0.17	0.04±0.08	0.03±0.07	0.00±0.00
MAPPO	0.27±0.04	0.34±0.05	0.15±0.04	0.63±0.05
MAT	0.45±0.07	0.27±0.04	0.20±0.04	0.66±0.02

low episodic rewards and fail to solve the unseen testing tasks. We hypothesis that the rule-based preys introduce stochasticity into these testing tasks. In such situation, there are significant distinctions between the testing tasks and training tasks, which prevents the learned multi-task policies from adapting well.

Table IV shows the zero-shot performance of all algorithms on the testing maps of four SMAC task sets. One can observe that RIT significantly outperforms other baselines across all four task sets, demonstrating its efficiency in both characterizing and transferring agent interactions across similar maps, thus solving the unseen maps. The baselines, including REFIL, UPDeT-MIX, and QMIX-ATTN, achieve notable performance on 3–8m and 3–8csz. This aligns with their performance in the multi-task evaluation, where these algorithms achieves preliminary interaction knowledge transfer among similar maps and thus possess moderate generalization capabilities. For the rest of algorithms, they fail to achieve efficient knowledge transfer and get stuck in low win rates within the multi-task evaluation, thus also exhibiting poor zero-shot performance.

E. Ablation Studies

To understand the superior performance of RIT, we compare it against two additional baselines on four task sets including cooperative navigation series, 3–8MMM, 3–8csz, and 3–8sz. These baselines consist of: 1) RIT-No-Random-Permutation, where we remove the operator of randomly permuting agents' decision order at the beginning of each episode to assess its effect and 2) RIT-No-Task-Representations, which excludes the usage of task representations and solely employs the interactive value decomposition paradigm. This ablation allows us to evaluate the effectiveness of learned task representations.

The results of ablation studies under the multi-task and zero-shot settings are shown in Fig. 4 and Table V, respectively. It is obvious that RIT-No-Task-Representations exhibits significant performance degradation compared to RIT on all four task sets, highlighting the importance of learning task representations to identify task relations for positive knowledge transfer across similar tasks. What is more, RIT-No-Task-Representations still outperforms QMIX-ATTN, which is attributed to its efficiency in explicitly characterizing agent interactions using the interactive value decomposition paradigm.

Moreover, the performance of RIT-No-Random-Permutation is comparable to that of RIT and converges better eventually in the cooperative navigation series. We hypothesis that the operator of random permuting agent decision order theoretically samples different agent permutations but has limited impact on simpler multi-agent tasks. However, for more complex tasks such as 3–8csz, 3–8MMM, and 3–8sz, excluding this operator leads to a significant degradation in both convergence rate and asymptotic performance. This demonstrates that this operator plays an important role in assisting agents' utility functions in characterizing agent interactions.

To further assess the effect caused by this random permuting operator, we incorporate the interactive value decomposition paradigm into both QMIX and VDN, resulting in two novel algorithms named QMIX-Int and VDN-Int. Additionally, we devise two baselines, QMIX-Int-NR (No-Random-Permutation) and VDN-Int-NR (No-Random-Permutation), by excluding the random permuting operator from them. We evaluate them on eight maps of the SMAC benchmark, under the single-task setting. As shown in Fig. 5, with the random permuting operator, both QMIX-Int and VDN-Int exhibit improvements in convergence rate and asymptotic performance on almost all maps, compared to QMIX-Int-NR and VDN-Int-NR. This further validates the effectiveness of the random permuting operator.

F. Visualization of Task Representations

We use a t-SNE plot shown in Fig. 6 to visualize the learned task representations of all task in the cooperative navigation series task set. Specifically, we again load the trained multi-task policies for agents and roll out 16 episodes for each task (80 episodes in total) to collect agents' trajectories in all tasks. Then, we use the trained task representation function to output all agents' representations regarding these tasks.

We can observe that the learned representations of tasks (2, 2), (3, 3), (4, 4) diverge from one another, a consequence

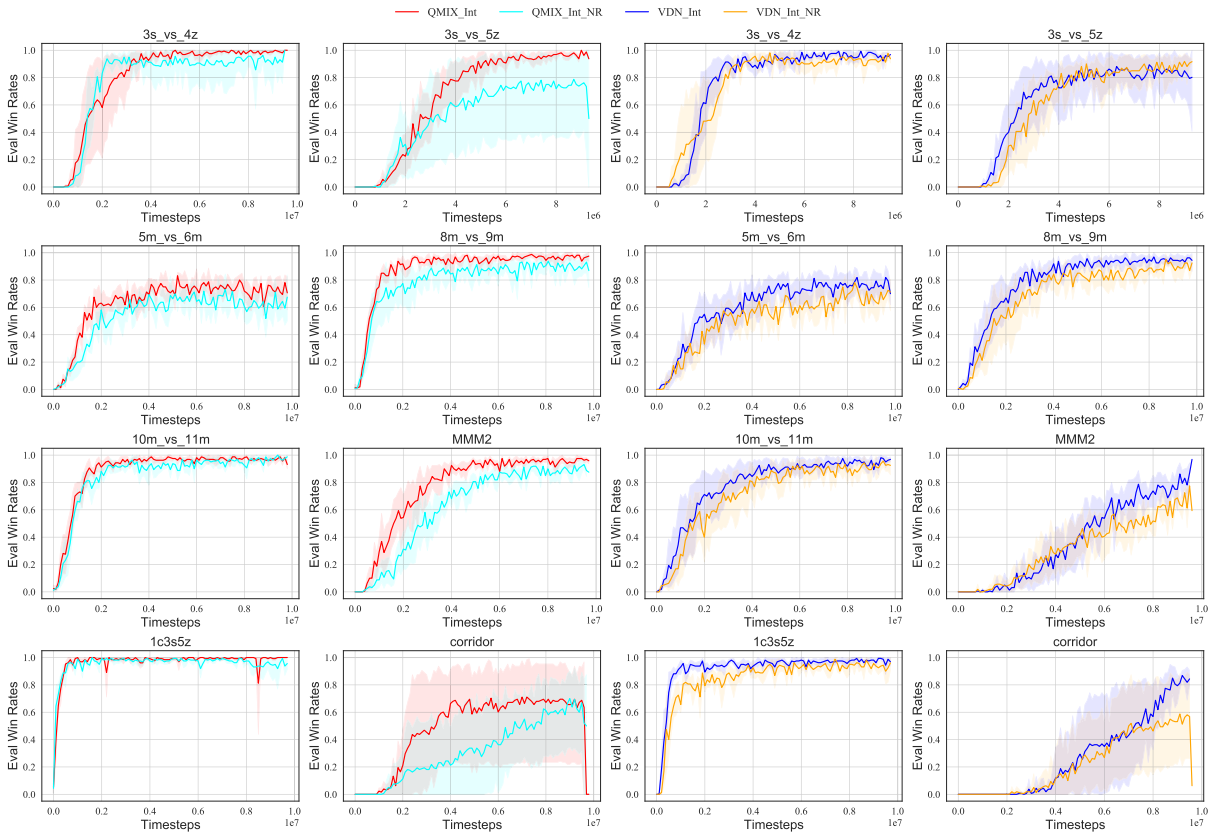


Fig. 5. Ablation study of the random permuting operator under the single-task setting.

TABLE V
ABLATION STUDIES UNDER THE ZERO-SHOT SETTING

Algorithm	Cooperative Navigation Series	3-8csz	3-8MMM	3-8sz
RIT	-342.4±14.1	0.70±0.07	0.54±0.07	0.59±0.08
RIT-No-Random-Permutation	-339.9±26.9	0.60±0.10	0.51±0.07	0.43±0.07
RIT-No-Task-Representations	-388.4±16.6	0.66±0.08	0.38±0.05	0.55±0.09
QMIX-ATTN	-418.5±20.2	0.61±0.03	0.27±0.01	0.23±0.03

of the unique trajectories of agents in solving these tasks. Notably, the representations of tasks (5, 5), (6, 6) share some common spaces with the ones of task (4, 4). This is because agents with the trained policies in the training task (4, 4) may generate similar trajectories in the unseen testing tasks (5, 5), (6, 6), thus emitting similar task representations. This demonstrates that the learned task representations are capable of characterizing task relations at the agent trajectories level, thereby resulting in knowledge transfer across partial scenarios of different tasks.

G. Server Settings

We run all experiments with five different random seeds by default and plot the mean/std in all figures. The experiments in the MPE task sets are performed on a server with two 12-core Intel[1] Xeon[1] CPU E5-2650 v4 CPUs, 256GB RAM, and 8 NVIDIA GTX1080Ti GPUs. The experiments in the multi matrix game and SMAC task sets are performed on a server with two 22-core Intel Xeon Gold 6238 CPUs, 512GB RAM, and 8 NVIDIA GTX2080Ti GPUs.

[1]Registered trademark.

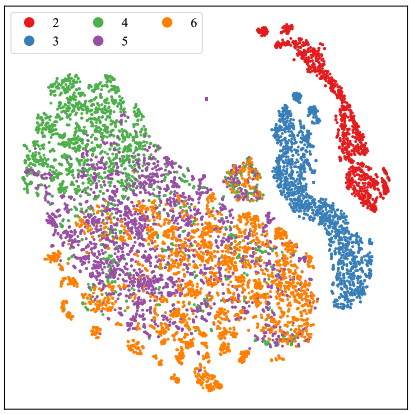


Fig. 6. t-SNE diagram of the task representations of all tasks in the cooperative navigation series. Labels 2, 3, 4, 5, and 6 refer to the tasks (2, 2), (3, 3), (4, 4), (5, 5), and (6, 6).

VII. CONCLUSION

In this article, we present RIT, a novel MT-MARL algorithm capable of characterizing both intra-task agent interactions and inter-task task relations. We highlight that efficient interaction knowledge transfer among multiple multi-agent tasks necessitates these two characterizations simultaneously. Extensive

experiments under multi-task and zero-shot settings confirm this viewpoint and validate the effectiveness of RIT.

A. Limitation and Future Works

A significant limitation of RIT is that each agent must be capable of observing the actions selected by preceding agents, which violates the decentralized execution. This limitation can be alleviated by distilling fully decentralized policies from the agents' policies learned by RIT. Another area of interest is extending RIT to large-scale multi-agent tasks and identifying task relations from a graphical task modeling perspective, which we leave as future work.

APPENDIX A
DETAILS OF THE ALGORITHM

To handle the varying dimensions of state inputs of different multi-agent tasks, we use the multi-head attention mechanism in the encoder module of the utility function, task representation function, and state encoder to extract population-invariant representations. In addition, we present a mask $\mathcal{M}_{o,t}$ to make these representations satisfy the partially observable property.

Similarly, we also use the multi-head attention mechanism in the decoder module of the utility function to learn fixed-length representations of the actions selected by per-agent preceding agents. To guarantee that each agent can only observe the actions selected by its preceding agents, we introduce a sequence mask $\mathcal{M}_e$ to customize eligible action representations for each agent, respectively, during the centralized training stage, and accordingly accelerate the learning process.

Given that multiple multi-agent tasks may involve different participating agents, state spaces, observation spaces, and other elements, the experiences from these tasks also have different dimensions, necessitating multiple replay buffers to store them. This approach can be prohibitively expensive. To address this problem, we align the dimensions of experiences from all tasks (including both training and testing tasks) by padding zero vectors based on the maximum number of entities (both agents and non-agents). Additionally, we introduce a scenario mask, $\mathcal{M}_s$, to indicate which entities are padded in different tasks. Consequently, we can create a universal replay buffer for all tasks, facilitating flexible batch data processing. Furthermore, the pre-definition regarding the maximum number of entities does not introduce any limitations, and our algorithm can adapt to other tasks not included in training and testing tasks.

In this section, based on the assumption that the maximum number of entities is set to m (where n for agents and $(m-n)$ for non-agents) for all training and testing tasks, we provide a detailed description of the partially observable mask $\mathcal{M}_{o,t}$, the sequence mask $\mathcal{M}_e$, and the scenario mask $\mathcal{M}_s$. Afterward, we demonstrate their practical usage in our algorithm.

A. Partially Observable Mask

In the formation of Factored Dec-POMDP (see Section III-A), we define the local observation o_t^i of each agent i as $o_t^i = \{s_t^{e^j} | \mu_t(i, e^j) = 1, \forall e^j \in \mathcal{E}\}$, where $\mu_t(i, e^j)$ denotes a binary observable mask indicating whether agent i can observe entity e^j at the current time step t and $\mu_t(i, i)$ always equals 1. Based on $\mu_t(i, e^j)$, we introduce the partially observable mask $\mathcal{M}_{o,t}$, described as an $n \times m$ matrix as follows:

$$\begin{bmatrix} \mu_t(1,e^1) & \mu_t(1,e^2) & \mu_t(1,e^3) & \cdots & \mu_t(1,e^m) \\ \mu_t(2,e^1) & \mu_t(2,e^2) & \mu_t(2,e^3) & \cdots & \mu_t(2,e^m) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \mu_t(i,e^1) & \mu_t(i,e^2) & \mu_t(i,e^3) & \cdots & \mu_t(i,e^m) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \mu_t(n,e^1) & \mu_t(n,e^2) & \mu_t(n,e^3) & \cdots & \mu_t(n,e^m) \end{bmatrix}$$

where each component $\mu_t(i, e^j)$ indicates that whether agent i can observe entity e^j at the current time step t. Based on this mask, we set the scores of unobservable entities to -inf for each agent in the multi-head attention mechanism. Then, the weights outputted by the softmax operation layer approximate 0 and thus the derived representations of each agent satisfy the partially observable property. In the experiments, we include the partially observable masks into transitions and store them into the replay buffer for more flexible data processing.

B. Scenario Mask

Furthermore, we introduce a scenario mask, $\mathcal{M}_s$, to indicate which entities exist in each task and which ones are padded. Based on this mask, we can exclude the information of these padded entities from all vectors (e.g., hidden states outputted by the GRU, values outputted by the utility function, representations in the state encoder, partially observable representations in the utility function and task representation function, and the representations of actions). Specifically, the scenario mask $\mathcal{M}_s$ is defined as a 1-D array as follows:

$$\left[\delta\left(e^1\right), \delta\left(e^2\right), \delta\left(e^3\right), \ldots, \delta(e^m)\right]$$

where each element $\delta(e^j)$ lies in $\{0, 1\}$ and indicates whether entity e^j is padded. Based on this mask, we set scores of these padded entities to -inf in all multi-head attention mechanisms and further set the corresponding components of them in other representations (such as utility values and hidden states outputted by the GRU) to 0. Additionally, the scenario mask remains fixed during each episode and is reset only when we reinitialize tasks. We also include the scenario masks into the experiences and store them in the replay buffer.

C. Sequence Mask

As done in the decoder of the vanilla Transformer [14], we also introduce a sequence mask, $\mathcal{M}_e$, into the decoder module of the utility function to ensure that each agent only observes the actions selected by its preceding agents during training. In detail, the sequence mask is defined as a $n \times n$ lower triangular matrix where all nonzero elements are equal 1

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 1 & 0 & \cdots & 0 \\ 1 & 1 & 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & 1 & \cdots & 1 \end{bmatrix}.$$

Based on this sequence mask, for each agent, we retain the score of action representations of its predecessors in the multi-head attention mechanism of the decoder module (of the utility

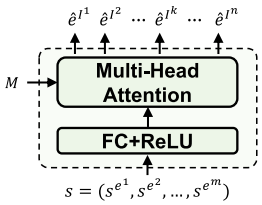


Fig. 7. Illustration of the multi-head attention mechanism.

function network), and set the others to -inf. Accordingly, each agent's final action representations, as outputted by the multi-head attention mechanism, only contain actions selected by its predecessors, satisfying the sequential decision requirement.

D. Transformation in the Attention Mechanism

We give a comprehensive description about the transformation in the attention mechanism for more clarity. As shown in Fig. 7, given the state $s = (s^{e^1}, s^{e^2}, \ldots, s^{e^m})$ whose length is equal to the maximum number of entities m, we first reshape it to the vector $\boldsymbol{S}$ whose shape is (m, d), where d represents the dimension of each entity's local state. In Section III-A, we assume the same local state spaces for all entities. Thus, we use a shared hidden layer (dubbed as the function F) to encode all entities' local states

$$F(\boldsymbol{S}; \boldsymbol{W}, \boldsymbol{b}) = \boldsymbol{S}\boldsymbol{W} + \boldsymbol{b}, \boldsymbol{S} \in \mathbb{R}^{m\times d}, \boldsymbol{W} \in \mathbb{R}^{d\times h}, \boldsymbol{b} \in \mathbb{R}^{h} \tag{11}$$

where h is the units of the hidden layer.

Based on the function F, we derive the representation $\boldsymbol{X} = F(\boldsymbol{S}; \boldsymbol{W}, \boldsymbol{b})$ whose shape is (m, h) and the rows of it represent the entities. Afterward, we specify the operation that defines an attention head. Given the addition inputs $J \subseteq \mathbb{Z}^{[1,m]}$, a set of indices that select which rows of $\boldsymbol{X}$ are used to compute queries such that $\boldsymbol{X}_J \in \mathbb{R}^{|J|\times h}$, and $\mathcal{M}$, a mask (e.g., the partially observable mask $\mathcal{M}_{o,t}$, the scenario mask $\mathcal{M}_s$, and the sequence mask $\mathcal{M}_e$) specifying the information of which entities each query entity can include into its own contexts. We refer to the attention mechanism as the operator ATTEN and show the transformations in it as follows:

$$\begin{aligned}&\text{ATTEN}\left(J, \boldsymbol{X}, \mathcal{M}; \boldsymbol{W}^Q, \boldsymbol{W}^K, \boldsymbol{W}^V\right)\\ &\quad= \text{softmax}\left(\text{mask}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{h^a}}, \mathcal{M}\right)\right)\boldsymbol{V} \in \mathbb{R}^{|J|\times h^a}\end{aligned} \tag{12}$$

where

$$\begin{aligned}&\boldsymbol{Q} = \boldsymbol{X}_J\boldsymbol{W}^Q, \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}^K, \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}^V, \mathcal{M} \in \{0,1\}^{|J|\times m},\\ &\boldsymbol{W}^Q \in \mathbb{R}^{h,h^a}, \boldsymbol{W}^K \in \mathbb{R}^{h,h^a}, \boldsymbol{W}^V \in \mathbb{R}^{h,h^a}\end{aligned} \tag{13}$$

where h^a represents the hidden units of current attention head.

In (12), the operator mask$(\boldsymbol{A}, \mathcal{M})$ takes two equal sized matrices (the score matrix whose shape is $(|J| \times m)$ and the mask matrix whose shape is also $(|J|\times m)$) and fills the entries of $\boldsymbol{A}$ with -inf in the indices where $\mathcal{M}$ is equal to 0. After the softmax, the weights of these entries become zero, thus preventing the query entities from attending to specific entities. $\boldsymbol{W}^Q, \boldsymbol{W}^K, \boldsymbol{W}^V$ are all learnable parameters of the query, key, and value layers.

After receiving the vector outputted by each attention head, we further define the multi-head attention mechanism MHA as the parallel computation of n^a attention heads as follows:

$$\begin{aligned}&\text{MHA}(J, \boldsymbol{X}, \mathcal{M})\\ &= \text{concat}\Big(\text{ATTEN}\Big(J, \boldsymbol{X}, \mathcal{M}; \boldsymbol{W}_b^Q, \boldsymbol{W}_b^K, \boldsymbol{W}_b^V\Big),\\ &\qquad b \in \left(1, 2, \ldots, n^a\right)\Big)\end{aligned} \tag{14}$$

where the final outputted vector is the concatenation of outputs of n^a attention heads.

Here, we use the fixed number of entities m and the ones of agents n only for clarity. The attention mechanism can be applicable to multiple multi-agent tasks with different m and n. Given the assumption that all entities have the same local state spaces, the shared hidden layer F and the attention head ATTEN are universal to different multi-agent tasks to encode the local states of entities in them. Thus, RIT is also universal to multiple tasks featured by different number of entities.

E. Transformation in the Encoder Module and Task Representation Function

In the encoder module, to ensure that the representations $(\hat{e}^{I^1}, \hat{e}^{I^2}, \ldots, \hat{e}^{I^n})$ of all agents satisfy the partially observable property, we replace the mask $\mathcal{M}$ with the partially observable mask $\mathcal{M}_{o,t}$, and define the index inputs J as the agent set $\mathcal{N}$. With the agent set $\mathcal{N}$, the computed queries $Q = \boldsymbol{X}_{\mathcal{N}}\boldsymbol{W}^Q \in \mathbb{R}^{n\times h^a}$, which only contain the agents. Then, we calculate the score matrix with $(\boldsymbol{Q}\boldsymbol{K}^\top/(h^a)^{1/2}) \in \mathbb{R}^{n,m}$, and further set the entries in it to -inf where $\mathcal{M}_{o,t}$ equals to 0. As a result, for each row (query agent), the weights of corresponding invisible entities outputted by the softmax equal 0 such that each query agent I^k only incorporates information of the entities that lie in its observable field into its local representations $\hat{e}^{I^k}$, satisfying the partially observable setting. In this process, only one attention layer is permitted to prevent the information of unseen entities being propagated through multiple rounds.

Additionally, we introduce the scenario mask $\mathcal{M}_s$ to remove the information of padded entities. In detail, the original $\mathcal{M}_s$ is an 1-D array whose length is m, which indicates which entities are padded. Similarly, we use the agent set $\mathcal{N}$ as the index inputs J, and select the components $\mathcal{M}_{s,\mathcal{N}}$ whose length is equal to n. Subsequently, we expand its dimension to match the size of certain vectors (e.g., hidden states outputted by the GRU, utility values). Finally, we set the entries in these vectors to 0 where the expanded $\mathcal{M}_{s,\mathcal{N}}$ is equal to zero.

In the task representation function, these two masks are used following the same way to extract partially observable representations $(\hat{x}^{I^1}, \hat{x}^{I^2}, \ldots, \hat{x}^{I^n})$.

F. Transformation in the Decoder Module

In the decoder module of the utility function, where each agent I^k takes the actions $a^{<I^k}$ selected by its predecessors as inputs. During training, we have access to the actions of all agents. We begin with an indicator a^{I^0} and initialize the action inputs of all agents as $\boldsymbol{a} = (a^{I^0}, a^{I^1}, a^{I^2}, \ldots, a^{I^{n-1}})$, whose length equals to the number of agents n. Based on the assumption that all agents share the same local action spaces, we can reshape the action inputs to the vector $\boldsymbol{A}$ with a shape of (n, d^a), where d^a represents the dimension of each agent's action. Following the operations in the attention mechanism stated above, we replace $\boldsymbol{S}$ with the action vector $\boldsymbol{A}$ and set the index inputs J to the agent set $\mathcal{N}$. We also replace the mask

$\mathcal{M}$ with the sequence mask $\mathcal{M}_e$. The remaining operations are the same as the original ones of the attention mechanism.

During centralized training, we have access to the selected actions of all agents at each time step, allowing us to directly shape the action inputs $\boldsymbol{a}$ and the vector $\boldsymbol{A}$. With the sequence mask $\mathcal{M}_e$, for each agent I^k, we retain the scores of actions selected by its predecessors and set others to -inf. Accordingly, in Fig. 1(a2), the learned action representations μ^{I^k} of agent I^k solely contain the information about the actions selected by its predecessors, satisfying the sequential decision paradigm.

During decentralized execution, all agents sequentially select actions following a pre-defined decision order. We initialize $\boldsymbol{A}$ as a zero vector with a shape of (n, d^a) and insert a^{I^0} into the first row of it. This initialization is then fed into the decoder module to calculate the utility function $Q^{I^1}(\tau^{I^1}, a^{I^1})$ for the first agent I^1. After agent I^1 selects its action a^{I^1}, we insert a^{I^1} into the second row of $\boldsymbol{A}$ and calculate Q^{I^2} for the second agent I^2 based on this updated vector. The action a^{I^2} selected by agent I^2 is inserted into the third row of $\boldsymbol{A}$. This process continues until the last agent I^n selects its action a^{I^n} based on the actions chosen by its predecessors.

G. Transformation in the Mixing Network

To efficiently coordinate the action selections of all agents, we use the mixing network [34] to calculate the global action value function Q^{total}. This mixing network taking all agents' utility functions as inputs requires that the parameters of the mixing layers depend on the number of agents present, while also incorporating state info. As shown in Fig. 1(c), we use the multi-head attention mechanism to extract representations $\hat{\boldsymbol{s}} = (\hat{s}^{I^1}, \hat{s}^{I^2}, \ldots, \hat{s}^{I^n})$ for the present agents. The shape of them is (n, h), where h is the hidden unit of the attention mechanism. We achieve this by setting the index inputs J to the agent set $\mathcal{N}$ and also removing the information of padded entities for each query agent using the scenario mask $\mathcal{M}_s$.

After extracting the representations $\hat{\boldsymbol{s}}$, we concatenate them with the task representations $\boldsymbol{z}$ of all agents. The shape of this concatenation is $(n, h + h^z)$, where h^z denotes the dimension of the task representations. Using this concatenation $[\hat{\boldsymbol{s}}, \boldsymbol{z}]$, we then use four hypernetworks to generate the weights $\boldsymbol{W}_1 \in \mathbb{R}^{+(n \times h^m)}$, $\boldsymbol{w}_2 \in \mathbb{R}^{+(h^m)}$, and the biases $\boldsymbol{b}_1 \in \mathbb{R}^{h^m}$, $b_2 \in \mathbb{R}^1$ of two-layer mixing layers, where h^m denotes the hidden unit of the mixing layers. For the weight $\boldsymbol{w}_2 \in \mathbb{R}^{+(h^m)}$ and the bias $\boldsymbol{b}_1 \in \mathbb{R}^{h^m}$, the shape of original outputs of the corresponding hypernetworks is (n, h^m). We average them across the n sized dimensions, changing their shapes from (n, h^m) to (h^m). For the bias b_2, the shape of original outputs of its hypernetwork is $(n, 1)$. We also average them to make their shape equal 1.

In addition, we employ the absolute value function to ensure positive weights, satisfying the monotonicity condition. This procedure allows for a dynamic generation of mixing networks whose input size varies with the number of agents. Based on all agents' utility functions $\boldsymbol{q} = (Q^{I^1}, Q^{I^2}, \ldots, Q^{I^n})$, the global action value function $Q^{\text{total}}(\boldsymbol{\tau}, \boldsymbol{a}, \boldsymbol{z}, s)$ is calculated below

$$Q^{\text{total}}(\boldsymbol{\tau}, \boldsymbol{a}, \boldsymbol{z}, s) = \text{ELU}\left(\left(\boldsymbol{q}^\top \boldsymbol{W}_1\right) + \boldsymbol{b}_1^\top\right)\boldsymbol{w}_2 + b_2. \tag{15}$$

REFERENCES

[1] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 427–438, Feb. 2012.

[2] Y. Jin, S. Wei, J. Yuan, and X. Zhang, "Hierarchical and stable multiagent reinforcement learning for cooperative navigation control," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 1, pp. 90–103, Jan. 2023.

[3] Y. Huang, S. Wu, Z. Mu, X. Long, S. Chu, and G. Zhao, "A multi-agent reinforcement learning method for swarm robots in space collaborative exploration," in *Proc. 6th Int. Conf. Control, Autom. Robot. (ICCAR)*, Apr. 2020, pp. 139–144.

[4] L. Wang et al., "Hierarchical multiagent reinforcement learning for allocating guaranteed display ads," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 10, pp. 5361–5373, Oct. 2022.

[5] L. Chen, B. Hu, Z.-H. Guan, L. Zhao, and X. Shen, "Multiagent meta-reinforcement learning for adaptive multipath routing optimization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 10, pp. 5374–5386, Oct. 2022.

[6] K. R. Allen, K. A. Smith, and J. B. Tenenbaum, "Rapid trial-and-error learning with simulation supports flexible tool use and physical reasoning," *Proc. Nat. Acad. Sci. USA*, vol. 117, no. 47, pp. 29302–29310, Nov. 2020.

[7] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," in *Proc. 34th Int. Conf. Mach. Learn.*, Aug. 2017, pp. 2681–2690.

[8] Y. Teh et al., "Distral: Robust multitask reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 4499–4509.

[9] W. Wang et al., "From few to more: Large-scale dynamic multiagent curriculum learning," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 5, pp. 7293–7300.

[10] Q. Long, Z. Zhou, A. Gupta, F. Fang, Y. Wu, and X. Wang, "Evolutionary population curriculum for scaling multi-agent reinforcement learning," 2020, *arXiv:2003.10423*.

[11] S. Hu, F. Zhu, X. Chang, and X. Liang, "UPDeT: Universal multi-agent reinforcement learning via policy decoupling with transformers," 2021, *arXiv:2101.08001*.

[12] S. Iqbal, C. A. S. D. Witt, B. Peng, W. Böhmer, S. Whiteson, and F. Sha, "Randomized entity-wise factorization for multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 4596–4606.

[13] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Mar. 2020.

[14] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[15] H. W. Kuhn and A. W. Tucker, Eds., "A value for n-person games," in *Contributions to the Theory of Games*, vol. 2. Princeton, NJ, USA: Princeton Univ. Press, 1953, pp. 307–318.

[16] G. Chalkiadakis, E. Elkind, and M. Wooldridge, "Computational aspects of cooperative game theory," *Synthesis Lectures Artif. Intell. Mach. Learn.*, vol. 5, no. 6, pp. 1–168, 2011.

[17] I. Mordatch and P. Abbeel, "Emergence of grounded compositional language in multi-agent populations," in *Proc. AAAI Conf. Artif. Intell.*, 2018, vol. 32, no. 1, pp. 1–8.

[18] M. Samvelyan et al., "The starcraft multi-agent challenge," in *Proc. 18th Int. Conf. Auto. Agents MultiAgent Syst.*, 2019, pp. 2186–2188.

[19] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *Autonomous Agents and Multiagent Systems*. Cham, Switzerland: Springer, 2017, pp. 66–83.

[20] F. Christianos, G. Papoudakis, M. A. Rahman, and S. V. Albrecht, "Scaling multi-agent reinforcement learning with selective parameter sharing," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 1989–1998.

[21] F. Christianos, L. Schäfer, and S. Albrecht, "Shared experience actor-critic for multi-agent reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 10707–10717.

[22] S. Wadhwania, D.-K. Kim, S. Omidshafiei, and J. P. How, "Policy distillation and value matching in multiagent reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 8193–8200.

[23] T. Yang et al., "An efficient transfer learning framework for multiagent reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 1–12.

[24] D. Ye, T. Zhu, Z. Cheng, W. Zhou, and P. S. Yu, "Differential advising in multiagent reinforcement learning," *IEEE Trans. Cybern.*, vol. 52, no. 6, pp. 5508–5521, Jun. 2022.

[25] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *J. Mach. Learn. Res.*, vol. 10, no. 7, pp. 1633–1685, 2009.
[26] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. 31st Conf. Neural Inf. Process. Syst.*, 2017, pp. 6382–6393.
[27] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 2974–2982.
[28] C. Yu et al., "The surprising effectiveness of PPO in cooperative multi-agent games," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, Nov. 2022, pp. 24611–24624.
[29] J. Grudzien Kuba et al., "Trust region policy optimisation in multi-agent reinforcement learning," 2021, *arXiv:2109.11251*.
[30] C. Sun, W. Liu, and L. Dong, "Reinforcement learning with task decomposition for cooperative multiagent systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 5, pp. 2054–2065, May 2021.
[31] M. Wen et al., "Multi-agent reinforcement learning is a sequence modeling problem," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, pp. 16509–16521, 2022.
[32] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proc. 10th Int. Conf. Mach. Learn.*, 1993, pp. 330–337.
[33] P. Sunehag et al., "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proc. 17th Int. Conf. Auto. Agents MultiAgent Syst.*, 2018, pp. 2085–2087.
[34] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "QMix: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4295–4304.
[35] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 5887–5896.
[36] J. Wang, Z. Ren, T. Liu, Y. Yu, and C. Zhang, "QPLEX: Duplex dueling multi-agent Q-learning," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–27.
[37] X. Yao, C. Wen, Y. Wang, and X. Tan, "SMIX(λ): Enhancing centralized value functions for cooperative multiagent reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 1, pp. 52–63, Jan. 2023.
[38] J. Wang, Y. Zhang, T.-K. Kim, and Y. Gu, "Shapley Q-value: A local reward approach to solve global reward games," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 5, pp. 7285–7292.
[39] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 12, pp. 5586–5609, Mar. 2021.
[40] T. Wang, H. Dong, V. R. Lesser, and C. Zhang, "ROMA: Multi-agent reinforcement learning with emergent roles," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2020, pp. 9876–9886.
[41] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

Chao Li received the B.S. degree in computer science and technology and the M.Sc. degree in computer software and theory from Shanxi University, Taiyuan, China, in 2017 and 2020, respectively, and the Ph.D. degree in computer science and technology from Nanjing University, Nanjing, China, in 2024.

His research interests include reinforcement learning and multiagent systems.

Shaokang Dong received the B.S. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2018, and the Ph.D. degree from the Department of Computer Science and Technology, Nanjing University, Nanjing, China, in 2024.

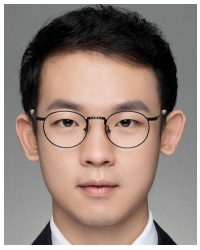
His research interests include reinforcement learning, multiagent learning, AI alignment, and AI agents.

Shangdong Yang received the B.Sc. degree in automation from Southwest Jiaotong University, Chengdu, China, in 2013, and the Ph.D. degree in computer science and technology from Nanjing University, Nanjing, China, in 2020.

He is currently an Assistant Professor with the School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing. His research interests include reinforcement learning and multiagent systems.

Yujing Hu received the bachelor's and Ph.D. degrees from the Department of Computer Science and Technology, Nanjing University, Nanjing, China, in 2010 and 2015, respectively.

He joined Alibaba, Hangzhou, China, in 2016, where he worked on applying reinforcement learning to the domains of search and recommendation and proposed the first reinforcement learning framework for solving the ranking problem of a search engine. He joined NetEase Fuxi AI Lab, Hangzhou, in 2018, where he worked on the application and research of reinforcement learning in computer games. He is currently leading the RL Research Group, NetEase Fuxi, Hangzhou. He has published more than 50 papers in top AI conferences and journals, such as NeurIPS, ICML, ICLR, IJCAI, AAAI, KDD, AAMAS, and IEEE TRANSACTIONS ON CYBERNETICS.



Tianyu Ding received the bachelor's degree in mathematics from Sun Yat-sen University, Guangzhou, China, in 2014, the master's degrees in financial mathematics and computer science from Johns Hopkins University (JHU), Baltimore, MD, USA, in 2016 and 2020, respectively, and the Ph.D. degree in applied mathematics and statistics from JHU in 2021.

He is currently a Principal Researcher with Microsoft Corporation, Redmond, WA, USA. His research interests include improving efficiency in machine learning and artificial intelligence, especially in areas like computer vision and generative models.

Wenbin Li received the Ph.D. degree from the Department of Computer Science and Technology, Nanjing University, Nanjing, China, in 2019.

He is currently an Associate Professor with the School of Intelligence Science and Technology, Nanjing University, Suzhou Campus, Suzhou, China. His research interests include brain-inspired artificial intelligence, advanced machine learning, computer vision, and collaborative optimization of software and hardware.

Yang Gao (Senior Member, IEEE) received the Ph.D. degree from the Department of Computer Science and Technology, Nanjing University, Nanjing, China, in 2000.

He is currently a Professor and the Director of the School of Intelligence Science and Technology, Nanjing University, Suzhou Campus, Suzhou, China. He has published more than 100 papers in top-tier conferences and journals. His research interests include artificial intelligence and machine learning.

Dr. Gao serves as the program chair and area chair for many international conferences.