Deep Multi-Task Multi-Agent Reinforcement Learning with Knowledge Transfer

Yuxiang Mai, Yifan Zang, Qiyue Yin, Wancheng Ni, Kaiqi Huang, *Senior Member, IEEE*

***Abstract*—Despite the potential of Multi-Agent Reinforcement Learning (MARL) in addressing numerous complex tasks, training a single team of MARL agents to handle multiple diverse team tasks remains a challenge. In this paper, we introduce a novel Multi-task method based on Knowledge Transfer in cooperative MARL (MKT-MARL). By learning from task-specific teachers, our approach empowers a single team of agents to attain expert-level performance in multiple tasks. MKT-MARL utilizes a knowledge distillation algorithm specifically designed for the multi-agent architecture, which rapidly learns a team control policy incorporating common coordinated knowledge from the experience of task-specific teachers. Additionally, we enhance this training with teacher annealing, gradually shifting the model's learning from distillation towards environmental rewards. This enhancement helps the multi-task model surpass its single-task teachers. We extensively evaluate our algorithm using two commonly-used benchmarks: StarCraft II micro-management and multi-agent particle environment. The experimental results demonstrate that our algorithm outperforms both the single-task teachers and a jointly-trained team of agents. Extensive ablation experiments illustrate the effectiveness of the supervised knowledge transfer and the teacher annealing strategy.**

***Index Terms*—Multi-task, multi-agent reinforcement learning, cooperation pattern, computer game.**

I. Introduction

With the recent advancements in deep learning, deep Multi-Agent Reinforcement Learning (MARL) has made significant strides in addressing real-world problems that can be effectively represented as multi-agent systems [1]–[3], such as computer games [4], autonomous driving [5] and robotics control [6]. Centralized training with decentralized execution (CTDE) [3], [7], [8] is a widely adopted paradigm in cooperative MARL which can realize efficient agent coordination. Under this paradigm, agents' policies are trained centrally using global information but executed decentrally based on local histories. To enable effective CTDE for MARL algorithms, many approaches [2], [3], [9] propose the factorization of the joint action-value to approximate the Individual-Global-Max (IGM) [3] principle.

Yuxiang Mai, Yifan Zang, Qiyue Yin and Wancheng Ni are affiliated with the Center for Research on Intelligent System and Engineering, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, China. E-mail:maiyuxiang2020@ia.ac.cn, zangyifan2019@ia.ac.cn, qyyin@nlpr.ia.ac.cn, wancheng.ni@ia.ac.cn.

Kaiqi Huang is affiliated with the Center for Research on Intelligent System and Engineering and the National Laboratory of Pattern Recognition, Institute of Automaiton, Chinese Academy of Sciences, Beijing 100190, China, with the University of Chinese Academy of Sciences, Beijing 100049, China, and also with the CAS Center for Excellence in Brain Science and Intelligence Technology, 100190 (kqhuang@nlpr.ia.ac.cn).

While MARL has demonstrated remarkable performance in individual tasks, training a single team of MARL agents to effectively handle multiple diverse tasks remains a challenge. In contrast to single-task MARL, where a team control policy is learned for a specific task, multi-task MARL necessitates agents to acquire a single team control policy capable of performing well across multiple tasks. For example, a team of construction robots learns to collaborate on multiple construction tasks, including material handling, building construction, etc. Multi-task MARL is recognized as a crucial milestone towards achieving Artificial General Intelligence (AGI) [10]. A direct approach involves training a team of MARL agents sequentially for multiple tasks using a conventional single-task learning algorithm. However, empirical evidence suggests that this approach yields poor performance even in single-agent reinforcement learning and can fail on certain tasks. These limitations arise from inherent dissimilarities and potential interference among multiple tasks [11]. Recent research has focused on addressing the challenge of knowledge transfer [12]–[14]. Actor Mimic [12] and policy distillation [13] aim to train a single multi-task agent in single-agent reinforcement learning. They employ task-specific teachers to guide the training process. Through knowledge transfer, it is feasible to integrate multiple control policies into a unified policy, attaining comparable or potentially enhanced performance on individual tasks. [14] extend policy distillation to multi-agent setting, using the extended experience replay to enable agents coordination. However, it was designed based on fully decentralized under partial observability, which is different from the CTDE paradigm due to the different information structure.

In this paper, we propose a novel Multi-task method based on Knowledge Transfer in cooperative MARL under the CTDE paradigm (MKT-MARL). By learning from task-specific teachers, our approach empowers a single team of agents to attain expert-level performance in multiple tasks. The coordination patterns of the MARL agents differ in each task, and it may be beneficial for the multi-task model to construct the associations of the different coordination patterns. In MKT-MARL, we construct the common coordinated feature of patterns for the multi-task agents and adapt it to different tasks. MKT-MARL utilizes a knowledge distillation algorithm specifically designed for the multi-agent architecture, which rapidly learns a team control policy incorporating common coordinated knowledge from the experience of task-specific teachers. We enhance this training with teacher annealing, gradually shifting the model's learning from distillation towards environmental rewards. Specifically, we use an adaptive

credit assignment network to facilitate the multi-task model in exceeding the performance of its single-task teachers. This method guarantees that the multi-task student agents receive a substantial training signal during the early phase of training, while avoiding the constraints of fully imitating the teacher.

We evaluate our algorithm with two commonly used benchmarks in starcraft multi-agent challenge [1] and multi-agent particle environment [15]. The experimental results show that our MKT-MARL can beat the single-task teachers in almost all tasks and exceeds the jointly-trained team of agents trained under state-of-the-art algorithms.

The main contributions can be summarized as follows.

1) To the best of our knowledge, this is the first multi-task MARL work under the CTDE paradigm.
2) We propose a multi-task agent network architecture for MARL, accommodating diverse task input lengths, enabling cross-task learning, extracting common cooperative patterns, and supporting multi-task execution within a single network.
3) To achieve stable and efficient agent learning across multiple tasks, we develop a novel teacher-annealing approach to combine task-specific teachers and environmental feedback to train the agents, allowing agents to obtain rich training signals in the early training phase while not being limited by the teachers' performance in the later training phase.
4) We compare the proposed method with several MARL baselines in two commonly used benchmarks. Extensive experimental results demonstrate the superiority of our method in terms of efficiency and final performance.

II. Related Work

A. Multi-task Deep Reinforcement Learning

Multi-task DRL has recently received much attention [12], [16]–[18]. Policy distillation [13] and Actor-Mimic [12] employ model compression techniques guided by task-specific teachers to train a single agent policy for multiple tasks. [16] propose to use separate action mapping layers for different tasks with a common parameter sharing feature extractor. [18] propose using offline learning from task-specific teachers' experience and online TD-learning [19] to train a multi-task agent for continuous control tasks. [17] propose to distill multiple policies into a single policy. [20] present a network with shared multi-task fully-connected layers and task-specific convolutional layers to perform different tasks. Although many multi-task models have emerged for single-agent reinforcement learning, work related to multi-task multi-agent reinforcement learning is relatively lacking.

B. Transfer Learning in MARL

Recent works [21]–[23] have made progress in transfer learning in cooperative MARL. Most of these works focus on designing adaptive network structures with no limitation on observation and action configuration requirements so that the agents can directly reload the previous knowledge. [21] and [22] propose using graph neural networks to represent the multiple agents so that they can reuse the well-trained agents for a new task. [16] utilize an attention module in both policy and critic to be general in varying input entities. UPDeT [23] first uses a transformer to realize a flexible network structure, which generates individual agent policies that can handle dynamic features. Although multi-agent transfer learning draws on previous experience to improve the agents' performance, its goal is mainly to enhance the performance of an individual task rather than to obtain a good policy on multiple tasks.

C. Credit Assignment Methods in MARL

The credit assignment problem [24] in cooperative MARL studies how a globally shared reward is assigned to each action of the agents. Credit assignment methods in MARL can be categorized into two main classes: policy-based credit assignment methods and value-based credit assignment methods. For value-based credit assignment methods, the value of each action is generally evaluated implicitly, and previous approaches have predominantly focused on learning how to feed the environmental reward signal directly to the action value function of each agent [2], [3], [9], [25]. VDN [2] directly sums each action value to calculate the global Q value. QMIX [3] implements a nonlinear combination of action values by introducing a global state network for the calculation of global Q values, which is an implementation of the IGM [3] condition. QTRAN [9] aims to overcome these limitations by employing a provably more comprehensive factorization but leaves the method with constraints that are difficult to solve computationally in this way. QPLEX [25] proposes the use of dueling networks to avoid monotonicity constraints, expanding the function clusters of the mixing network. Different from the value-based credit assignment methods, policy-based approaches [26]–[28] typically evaluate individual actions explicitly against a certain baseline, which provides strategies that can be shown to be at least locally optimal in terms of agent contributions [26]. COMA [29] utilizes a centralized critic to estimate the counterfactual action values to serve as a baseline. SQDDPG [27] combines the Shapley values [30] from game theory to calculate the advantage of each action. LICA [28] uses an implicit credit assignment similar to value-based methods. In this paper, we mainly focus on the value-based credit assignment methods. Although the previous methods demonstrate remarkable performance in many complex cooperative tasks, they can only execute in single tasks and cannot be used directly in multi-task MARL.

III. Background

Our algorithm MKT-MARL can be easily embedded into existing value-based and policy-based methods, of which we choose the representative method QMIX as the basis. In this section, we introduce reinforcement learning, multi-agent reinforcement learning, and the representative method QMIX.

A. Reinforcement Learning

Reinforcement learning [31] is a classical algorithm for solving sequential decision problems, where agents interact

with the environment over time. Reinforcement learning acquires knowledge directly from feedback received through interactions with the environment, bypassing the need for learning from training samples. In recent decades, there have been numerous advancements in techniques and architectures aimed at enhancing the performance and efficiency of reinforcement learning across diverse tasks, including video games.

Specifically, the sequential decision problem in reinforcement learning is usually modeled by MDP [32], a mathematical formulation extensively employed in the study of optimization problems. A tuple $\langle S, A, P, R, \gamma \rangle$ can be used to represent MDP. At each time step t, the agent observes a state $s_t \in S$ and chooses an action $a_t \in A$ based on its individual policy $\pi(a|s)$. Subsequently, the environment transitions to the next state s_{t+1} through the transition function P and gives an extrinsic reward $r_t \in R$ to the agent. The discount return, denoted as $R_t = \sum_{c=0}^{\infty} \gamma^c r_{t+c}$, represents the cumulative sum of rewards. The discount factor $\gamma \in [0, 1)$ is a parameter that determines the extent to which future cumulative rewards are taken into account relative to immediate rewards. The objective of the reinforcement learning algorithm is to help the agent discover the optimal policy π^* that maximizes the discounted reward. In reinforcement learning algorithms, Q-values are widely used to estimate the discounted rewards of state-action pairs. Additionally, the value of a state-action pair and the value of a state can be defined as:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{s_{t+1:\infty}, a_{t+1:\infty}}\left[R_t | s_t, a_t\right],$$

$$V^{\pi}(s_t) = \mathbb{E}_{a \sim \pi(s_t)}\left[Q^{\pi}(s_t, a_t)\right].$$

Q-learning [33] is a classical algorithm in reinforcement learning that uses a Q-table to represent the estimated state-action values, and approximates the true state-action values by continuously updating the Q-table in the following way:

$$Q(s_t, a_t) = (1 - \alpha) Q(s_t, a_t) + \alpha(r_t + \gamma \max_{\alpha} Q(s_{t+1}, a)),$$

where α is the learning rate. The optimal policy π^* can be obtained according to the optimal $Q^*(s, a)$:

$$\pi^*(s) = \arg\max_{a} Q^*(s, a).$$

Deep Q-learning [34] uses the CNN [35] network to extend Q-learning by replacing the Q-table estimation of state-action pairs with neural networks, allowing the characterization of state-action pairs to be enhanced for use in complex video games. The Bellman equation including the network parameters is updated as presented in

$$Q(s_t, a_t) = r_t + \gamma \max_{a} Q(\delta_t, a; \phi^-),$$

where δ_t represents the pre-processed state s_t and ϕ denotes the parameters of the neural network.

B. Multi-agent Reinforcement Learning

MARL [31] extends reinforcement learning to address sequential decision problems involving multiple agents. The fully cooperative multi-agent task is formulated as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [36]. In this framework, a Dec-POMDP is represented by a tuple $\langle S, A, U, Z, O, P, r, n, \gamma \rangle$, where S represents the global states of the environment and A denotes the finite set of n agents. At each time step t, agent $i \in A$ selects an action u_i from its action set U_i, resulting in a joint action $\boldsymbol{u} \in \boldsymbol{U} \equiv U^n$. The transition function P determines the next state $s' \in S$ given the current state s and joint action $\boldsymbol{u}$. All agents receive the same shared reward $r(s, \boldsymbol{u})$ based on the current state and joint action, with a discount factor $\gamma \in [0, 1)$ influencing the importance of future rewards. We consider a partially observable setting that each agent receives only a local observation $z_i \in Z$ based on the observation function $O(s, \boldsymbol{u}) : S \times \boldsymbol{U} \to Z$ with observation space Z. Each agent learns an individual policy $\pi_i(u_i|\tau_i; \phi_i)$ with parameters ϕ_i from its own action-observation trajectory τ_i. The environment receives a joint action from the agents' joint policy. γ is the discounted factor.

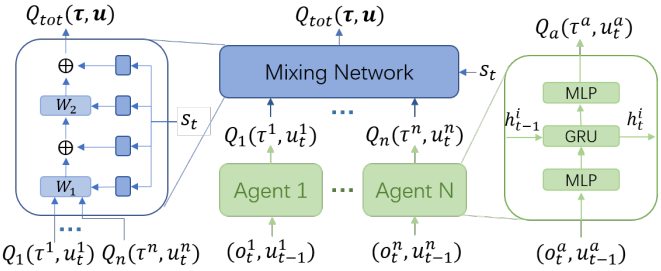

Fig. 1. The network structure of QMIX from origin paper [3]. Our proposed algorithm MKT-MARL is implemented based on the QMIX structure. In the distributed execution stage, each agent network (green) makes decisions based on its own local observations. In the centralized training stage, the outputs of the agents are combined by the mixing network (blue) to estimate the joint action values Q_{tot}, and then the loss between the actual joint action values and Q_{tot} is used to train the mixing network and the agent network.

The widely accepted approach for Dec-POMDP problems is the Centralized Training and Decentralized Execution (CTDE) paradigm. The CTDE paradigm utilizes a mixing network parameterized by θ to compute the joint action value Q_{tot} by aggregating individual Q-values:

$$Q_{tot}(\tau, \boldsymbol{u}, s; \theta) = f\left(\left[Q_i(\tau_i, u_i)\right]_{i=1}^{n}, s; \theta\right).$$

Afterwards, the entire network is updated using TD-learning:

$$\mathcal{L}(\theta, [\theta_i]_{i=1}^{n}) = \sum_{b=1}^{\mathcal{B}} \left[\left(y_b^{tot} - Q_{tot}(\tau, \boldsymbol{u}, s; \theta)\right)^2\right],$$

where $\mathcal{B}$ represents the batch size of transitions, $y^{tot} = r + \gamma \max_{\boldsymbol{u}'} Q_{tot}(\tau', \boldsymbol{u}', s'; \theta^-)$, and θ^- is the parameter of the target network.

QMIX is a classic algorithm in MARL, with a dominant performance in many scenarios including Starcraft II and MPE. Our proposed algorithm builds on QMIX, which is depicted in Figure 1. QMIX effectively decomposes the joint state-action value into individual utility functions for each agent using a mixing network, satisfying the IGM [3] condition in a monotonic and nonlinear manner.

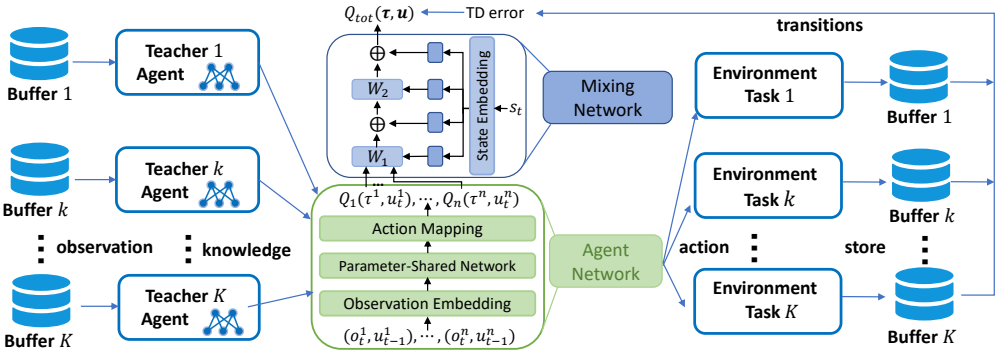

Fig. 2. Diagram of the proposed method. The multi-task agent consists of two parts, the agent network and the mixing network. Agent network generates actions, interacts with each environment, and stores the data of the interaction in each replay buffer. On the one hand, the data in the replay buffer is used to train the mixing network and the agent network using TD learning. On the other hand, the action knowledge is computed by the pre-trained teacher network input observation and distilled to the agent network.

IV. Method

Multi-task MARL goes a step further from MARL, with the goal of enabling multiple agents in a single team to perform different multiple tasks. In this section, we introduce the MKT-MARL algorithm, a multi-task multi-agent approach that leverages knowledge transfer. Specifically, we first introduce the framework of our algorithm and then detail the two important modules of our approach and the teacher annealing strategy that connects them.

A. Overview of MKT-MARL

We first give a brief overview of our MKT-MARL as shown in Figure 2. We build our algorithm based on a value-based algorithm called QMIX [3]. Our approach can be flexibly adapted to integrate with various MARL algorithms [2], [9], [25]. QMIX consists of two parameterized modules: the agent networks, which derive the corresponding action value $Q_i(\tau_i, u_i)$ based on each agent's locally observed trajectory τ_i, and a mixing network that derives the joint action value Q_{tot} using the action values $[Q_i(\tau_i, u_i)]_i^n$ chosen by all the agents. The design of the mixing network aims to fulfill the monotonicity assumption, which approximates the IGM condition. In QMIX, the collected transition samples $(\tau_t, \boldsymbol{u}_t, s_t, r_t, \tau_t', \boldsymbol{u}_t', s_t')$ are stored using an experience replay buffer $\mathcal{D}$ similar to DQN. Consider a set of K well-trained teacher agents, each trained specifically for a particular task, denoted as $\mathcal{T}_1, ..., \mathcal{T}_k, ..., \mathcal{T}_K$. Each teacher $\mathcal{T}_k$ consists of a team of agent networks $[\pi_i]_i^n$ that have been trained by QMIX until they converge on their respective tasks, forming a well-coordinated model. Our goal is to transfer the knowledge of these coordinated teacher agents to a unified team of student agents $\mathcal{S}$, enabling them to achieve comparable or enhanced performance across these tasks.

MKT-MARL consists of two components: the supervised knowledge transfer and the adaptive TD-learning, connected by teacher annealing. The supervised knowledge transfer component provides the multi-task agents to obtain guidance from the experience of the well-trained teachers to gain the common coordinated knowledge, which is described in Section IV-B. While the adaptive TD-learning component provides environmental feedback to multi-task agents to revise the learned common coordinated pattern. The teacher annealing strategy gradually allows the multi-task agents to learn from the teacher to learn from feedback from the environment. This approach guarantees early provision of adequate learning signals to the student while avoiding strict reliance on teacher imitation, which is described in detail in Section IV-D.

In addition, since the length of data varies between different tasks, to reduce data redundancy, we use K separate experience buffers rather than a single unified buffer to store transition samples from individual tasks, and each buffer stores transition samples of one task. At each epoch, MKT-MARL samples $\mathcal{B}$ transition from each buffer to form a mini-batch and then use each mini-batch to train the multi-task agents to mitigate catastrophic forgetting. Thus, the multi-task agents are trained with K mini-batches at each epoch.

B. Supervised Knowledge Transfer

In order to transfer the knowledge from teacher policies to a single student network, we employ a parameter-sharing student network to facilitate collaborative pattern mining. Subsequently, we use Kullback-Leibler (KL) divergence to reduce the discrepancy between the distributions of teacher and student policies for knowledge transfer. Parameter sharing is a common configuration for training deep MARL algorithms. Under parameter sharing, all agents utilize a common set of parameters for their networks. In addition, agents additionally receive the identity of each agent as a one-hot vector input. This allows each agent to develop a different behavior. The loss is computed over all agents and used to optimize the shared parameters. Using parameter sharing makes it easier for the agents to obtain coordinated behavior. Since the length of the agents' observations and actions vary across tasks, we extend the parameter sharing for multi-task agents to accommodate different tasks' inputs and outputs.

In the multi-task multi-agent policy training, we use a separate observation embedding layer $\{E_1, ..., E_k, ..., E_K\}$ for the observation embedding. For task k, the embedding layer E_k is used to map each agent's observations $o_{k,i}$ to the same length, and then the embedded observations are used in the

parameter sharing layer to construct the common coordinated feature as follows:

$$\begin{aligned} e_k &= E_k(o_{k,1}, o_{k,2}, ..., o_{k,n}), \\ c &= \mathcal{F}(e_k), \end{aligned} \tag{1}$$

where c is the common coordinated pattern. $\mathcal{F}$ is usually RNN layers or MLP layers. We use MLP in our experiments with the same setting in EPyMARL [37], and build $\mathcal{F}$ to the same number of MLP layers as the single-task network. Afterwards, since the coordination patterns of the agents may be different in tasks, we use a separate action mapping layer $\{EA_1, ..., EA_k, ..., EA_K\}$ to adapt the common coordinated knowledge to each task as follows:

$$(Q_1, Q_2, ..., Q_n) = EA_k(c). \tag{2}$$

The separate action mapping layer is similar to the observation embedding layer, while solving the problem of the varying length of agents' actions in different tasks. In order for the student agents $\mathcal{S}$ to be able to utilize the experience from the teacher agents $\mathcal{T}_k$, we optimize the parameters ϕ of the multi-task agents $\mathcal{S}$ by employing the KL divergence with a temperature parameter t:

$$\mathcal{L}_{KL}(\mathcal{B}_k, [\phi]_{i=1}^n) = \sum_{b=1}^{\mathcal{B}_k} \sum_{i=1}^{n} \operatorname{softmax}(\tfrac{Q_{i,b}^{\mathcal{T}}}{t}) \ln \tfrac{\operatorname{softmax}(\frac{Q_{i,b}^{\mathcal{T}}}{t})}{\operatorname{softmax}(Q_{i,b}^{\mathcal{S}})}. \tag{3}$$

In MKT-MARL, we use the multi-task multi-agent policy to interact with each environment and store the collected transition data in the corresponding buffer. At each epoch, we sample the transition data from each buffer with batch size $\mathcal{B}$. For task k, we can calculate the action value vector $Q^{\mathcal{T}}$ using the teacher agents $\mathcal{T}_k$ and the action value vector $Q^{\mathcal{S}}$ of the student agents S though the collected observation action pairs $(o_i, \boldsymbol{u})$ for each agent. The task identifier is employed to accurately switch to the appropriate output during both training and evaluation phases. The teacher agents $\mathcal{T}_k$, with their well-coordinated pattern and precise Q-value estimation capabilities, can effectively guide the weight updates of the agent network using Eq. 3.

C. Adaptive TD-learning

The supervised knowledge transfer uses teachers' experience to accelerate the learning process of the multi-task agents, enabling them to rapidly acquire a proficient team agent policy. However, the lack of feedback from the actual environment and the inadequate acquisition of new knowledge by the multi-task agents may lead to over-fitting. To overcome this challenge, we introduce adaptive TD-learning, a method that can adaptively learn the parameters of both the mixing network and the agent network by leveraging feedback from multiple environments with varying input lengths. In order to get the coordination pattern for all the tasks, we propose an adaptive credit assignment network. Due to the global state of each task may have a different length, we use a separate state embedding layer $\{SE_1, ..., SE_k, ..., SE_K\}$ to map the states to same length. The joint action value can be calculated as:

$$Q_{tot}(\tau, \boldsymbol{u}, SE_k(s); \theta) = f\left([Q_i]_{i=1}^n, SE_k(s); \theta\right). \tag{4}$$

Algorithm 1 MKT-MARL

Input: A set of K tasks, each with a teacher $\pi_{\mathcal{T}}^k$ and replay buffer $\mathcal{D}_k$, multi-task student policy $\pi_{\mathcal{S}}$, target student network $\pi_{\mathcal{S}}^{tar}$, maximum number of epochs $T_{\max}$, target network update frequency c;
Initialize: the weights of $\pi_{\mathcal{S}}$;
for $t = 1$ to $T_{\max}$ **do**
  for $task\ k$ in task set **do**
    $\mathcal{D}_k \leftarrow \mathcal{D}_k \cup \textsc{Rollout}(\pi_{\mathcal{S}}, k)$;
    Sample $\mathcal{B}$ transition samples from buffer $\mathcal{D}_k$;
    Calculate the teacher action value $q^{\mathcal{T}}$ by $\mathcal{T}_k$;
    Calculate the student action value $q^{\mathcal{S}}$ by $\pi_{\mathcal{S}}$;
    Calculate $\mathcal{L}_{KL}$ by Eq. 3;
    Calculate $\mathcal{L}_{TD}$ by Eq. 5;
    Calculate annealing coefficient λ with t;
    Update $\pi_{\mathcal{S}}$ by Eq. 6;
  end for
  if t mod c = 0 **then**
    Synchronize the target student network $\pi_{\mathcal{S}}^{tar} := \pi_{\mathcal{S}}$;
  end if
end for
return $\pi_{\mathcal{S}}$

Moreover, each task has a different reward structure, and the rewards scales may vary widely. It is difficult to train the multi-task network directly using the origin reward to obtain a multi-task model that performs well on each task. Therefore, we use the reward standardization technique [37] to normalize the reward values in each mini-batch. The hyper-network structure is the same as QMIX [3]. For task k, the entire network is updated using the TD-learning:

$$\mathcal{L}_{TD}(\theta, [\theta_i]_{i=1}^n) = \sum_{b=1}^{\mathcal{B}} \left[\left(y_b^{tot} - Q_{tot}\left(\tau, \boldsymbol{u}, SE_k(s); \theta\right) \right)^2 \right], \tag{5}$$

where $y^{tot} = \hat{r} + \gamma \max_{\boldsymbol{u}'} Q_{tot}(\tau', \boldsymbol{u}', SE_k(s'); \theta^-)$, and $\hat{r}$ is the standardized reward.

D. Teacher Annealing

In order to allow the agents to learn from both the task-specific teacher's knowledge and environmental feedback, we mix the supervised learning of the task-specific teacher predictions with the adaptive TD-learning of the environmental feedback. Specifically, we use the annealing strategy to obtain the term in the summation of the L_{KL} and L_{TD} as:

$$\mathcal{L} = (1 - \lambda)\mathcal{L}_{KL} + \lambda \mathcal{L}_{TD}, \tag{6}$$

where λ is linearly increased during the training process. In the early stages of training, the multi-task policy is mostly learned from the task-specific teachers. In the final stages of training, the model primarily relies on environmental rewards to facilitate its learning, enabling it to surpass the performance of its single-task teachers.

We summarize MKT-MARL in Algorithm 1. The whole network is updated by both the policy distillation and the adaptive TD-learning connected by the annealing coefficient.

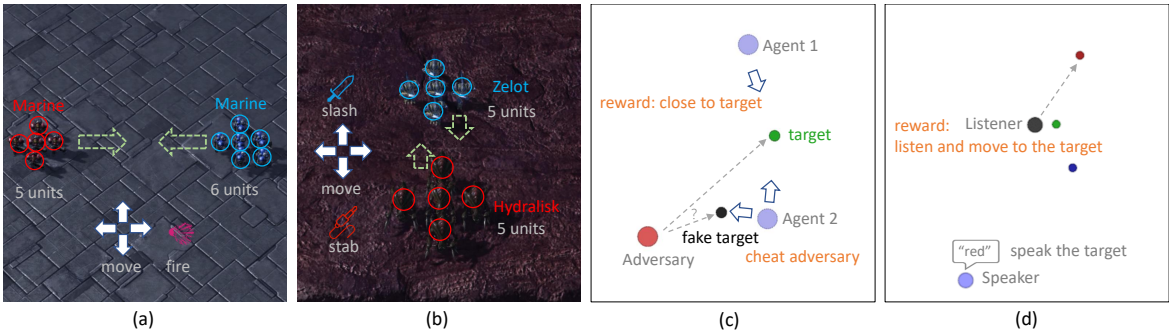

Fig. 3. Screenshots from SMAC and MPE which can be converted to state vectors as agent input. Because of the similar collaboration patterns of the similar number of agents, we choose the environment with the same number of agents as our main experimental environment. The first two figures are from SMAC and the last two figures are from MPE.(a) 5m_vs_6m. (b) 5h_vs_5z. (c) Simple Adversary. (d) Simple Speaker Listener.

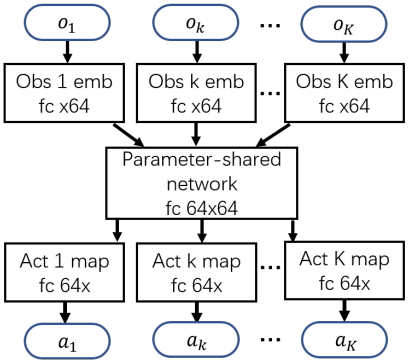

Fig. 4. Deep multi-task agent network architecture for MKT-MARL. The network consists of three parts: a separate observation embedding layer, a parameter-shared layer, and a separate action mapping layer. The inputs of the network are the observations corresponding to each task, and the outputs are the actions corresponding to each task observation.

At each epoch, the multi-task agents collect transitions from all tasks and store them in the corresponding K separated memory buffers. And then, the transition data are sampled from each buffer with a batch size $\mathcal{B}$. Throughout the training process, the supervised knowledge transfer first accounts for a greater proportion of the training of the multi-task agents. Guided by the task-specific teachers, $q^{\mathcal{T}}$ is calculated to be the supervised learning label. And then, the student multi-agent policy calculates action values, and the loss between the teacher labels and student action values back-propagates the gradient to train the student agent network. The multi-task agent network is updated by policy distillation. As the training continues, the annealing coefficient λ gradually becomes larger based on the calculation of the training epoch, and the weight of training signals gradually shifts to the loss of adaptive TD-learning, allowing the multi-task agents to obtain more training signals on the environmental feedback. After each preset fixed epoch, the network parameters of the target student are synchronized with the most recent student network parameters.

V. Experimental Setup

To assess the performance of the proposed MKT-MARL in handling multiple tasks, we evaluate our method using two widely utilized benchmarks: Starcraft Multi-Agent Challenge (SMAC) [1] and Multi-agent Particle Environment (MPE) [15]. This section presents a detailed description of the training environments, the structure of the multi-task agent network, the baseline method we chose for comparison and the training setup we used to train the proposed algorithm.

A. Environments

Starcraft Multi-Agent Challenge (SMAC): SMAC comprises a collection of cooperative multi-agent tasks derived from the real-time strategy game StarCraft II [38] as shown in Figure 3, graphs (a) and (b). These tasks primarily revolve around the micro-management of agents. In SMAC, individual agents are responsible for controlling each unit based on their local observations, while the enemy units are governed by an integrated rule-based AI system within the game. An episode ends either when all units on one side are eliminated or when a predetermined time step limit is reached. The game is won only if all enemies are dead. The objective of optimization is to maximize the rate of winning. We chose tasks with the same number of agents but different unit types as a group in our experiments. The scenarios selected as a group are listed in the same blank row in Table I. Furthermore, the corresponding learning curves are presented in Figure 5.

Multi-agent Particle Environment (MPE): MPE is a multi-agent particle environment characterized by continuous observation and discrete action spaces as shown in Figure 3, graphs (c) and (d). Particle agents can move, communicate, see each other, push each other around, and interact with fixed landmarks. Similar to the SMAC experiments, we select tasks with the same number of agents as a group in our experiments. The tasks we choose are simple speaker listener, simple push and simple adversary. These tasks are listed in the same blank row in Table II, and the corresponding learning curves are displayed in Figure 6.

B. Network Architecture

The details of our multi-task policy model are shown in Figure 4. Our multi-task model can execute in multiple tasks, collect data, and learn from this data. Since different tasks often have different sets of observations and actions, we employ separate input and output layers, referred to as

the embedding layer and the controller layer respectively, for each task. The task identifier is utilized to switch to the appropriate input and output configurations during both training and evaluation phases. The parameter sharing layer is usually RNN layers or MLP layers. We use MLP in our experiments with the same setting in EPyMARL [37].

C. Baseline Method

Our algorithm is compared with QMIX trained on a single task, which serves as an optimal solution (QMIX-Teacher),

TABLE I
THE FINAL AVERAGE WIN RATE ON SMAC.

	MKT-MARL (ours) CTDE	Dec-HDRQN non-CTDE	QMIX-Multi CTDE	MAPPO-Multi CTDE	VDN-Multi CTDE	QMIX-Teacher CTDE
5m_vs_6m	**0.26**	0.16	0.18	0	0.04	0.18
5h_vs_5z	**0.94**	0.81	0.9	0.64	0.49	0.85
2s3z	**0.99**	0.97	0.95	**0.99**	0.72	0.97
corridor	**0.76**	0.17	0.72	0	0	0.75
6h_vs_7z	**0.33**	0.29	0.23	0	0	0.27
8m	**1.0**	0.99	0.98	0.97	0.88	**1.0**
8m_vs_9m	**0.81**	0.69	0.56	0.08	0.16	0.76
3s5z	**0.94**	**0.94**	0.87	**0.94**	0.01	0.9
MMM	**1.0**	0.75	0.68	**1.0**	0.43	**1.0**
MMM2	**0.88**	0.16	0.55	0.18	0	0.77
10m_vs_11m	**0.9**	0.88	0.7	0.35	0.01	0.84
8m	0.99	0.98	0.97	0.89	0.65	**1.0**
8m_vs_9m	**0.8**	0.62	0.58	0.02	0	0.76
1s7z	**0.91**	0.82	0.69	0.59	0	0.82
2s6z	**0.96**	0.82	0.85	0.6	0	0.92
3s5z	**0.91**	0.79	0.74	0.41	0.03	0.9
4s4z	0.92	0.82	0.79	0.75	0	**0.93**
5s3z	**0.95**	0.82	0.69	0.76	0	0.9
6s2z	0.88	0.8	0.59	0.78	0.02	**0.92**
7s1z	**0.95**	0.81	0.77	0.77	0	0.92
8s_vs_8z	**0.95**	0.82	0.46	0	0	0.81
MMM2_8	**0.6**	0.11	0	0	0	0.3
MMM2_8_2	**0.85**	0.28	0.34	0	0	0.83
27h_vs_30z	**0.91**	0.87	0.25	0.76	0	0.85
27m_vs_30m	**0.87**	0.76	0.36	0	0	0.76

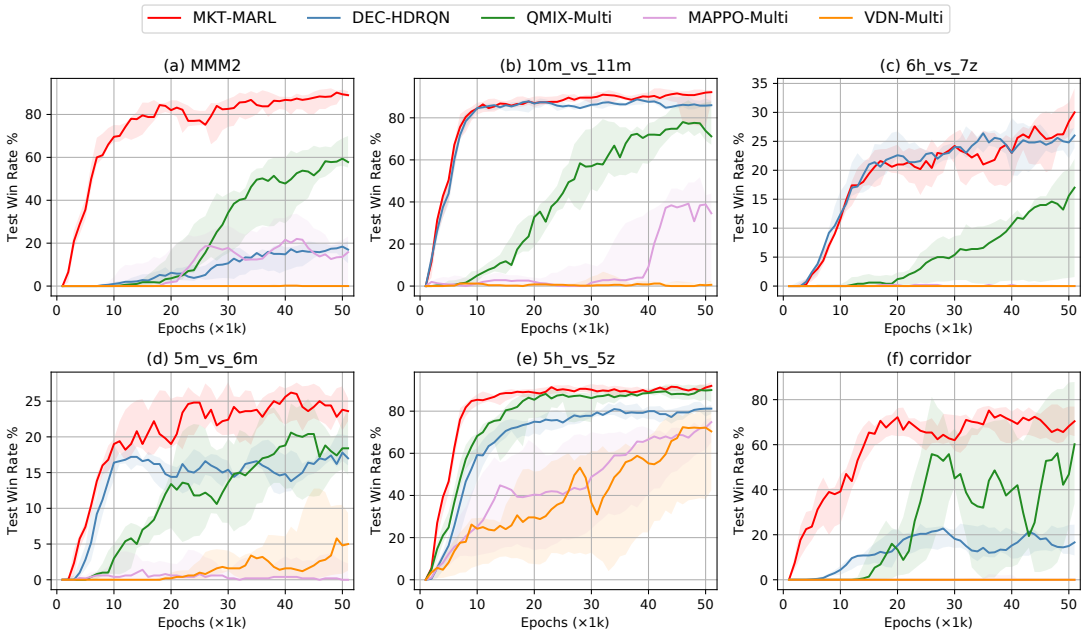

Fig. 5. Learning curves of grouped multi-task training on SMAC. The horizontal axis is the epoch and the vertical axis is the win rate, a common metric used in SMAC to assess the strength of the agents. From the comparison, we can see that our algorithm outperforms other algorithms in terms of efficiency and final performance, and the performance is not limited by the teacher policy. In contrast, Dec-HDRQN with only policy distillation is more influenced by the strength of the teacher policy.

TABLE II
THE FINAL AVERAGE REWARDS ON MPE.

	MKT-MARL (ours) CTDE	Dec-HDRQN non-CTDE	QMIX-Multi CTDE	MAPPO-Multi CTDE	VDN-Multi CTDE	QMIX-Teacher CTDE
speaker listener	-28.2	**-26.6**	-28.02	-40.83	-53.98	-30.14
simple push	-9.54	-9.65	-10.23	**-8.86**	-11.48	-9.64
simple adversary	**18.25**	16.69	12.55	16.85	15.26	16.57

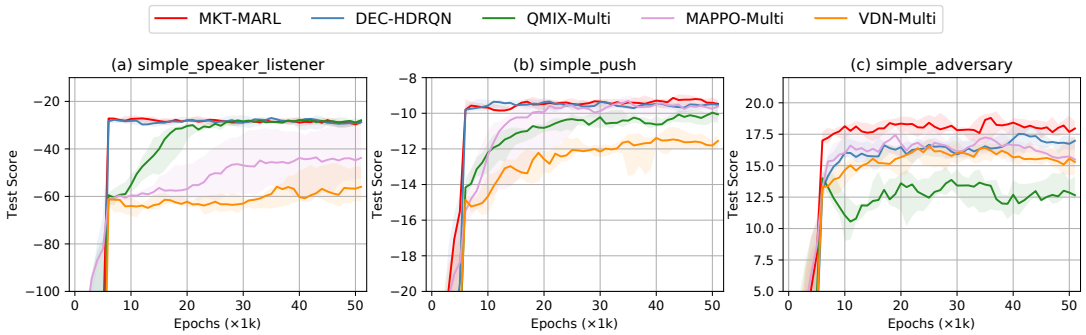

Fig. 6. Learning curves of grouped multi-task training on MPE. The vertical axis indicates the test scores in the evaluation session. The comparison shows that our algorithm is able to beat or flat other algorithms in the simple environments.

TABLE III
HYPERPARAMETERS FOR TRAINING.

action selector	epsilon greedy
epsilon start	1.0
epsilon finish	0.05
evaluation epsilon	0
buffer size	5000
target update interval or tau	200
obs agent id	True
obs last action	False
obs individual obs	False
standardise rewards	True
agent output type	q
double q	True
use rnn	False
mixing embed dim	32
mixing map embed dim	32
hypernet layers	2
hypernet embed	64
learning rate	0.0005
optimizer	Adam

as well as state-of-the-art multi-task MARL algorithms and MARL algorithm extensions designed for multi-task environments. In particular, as mentioned above, we compare our algorithm with Dec-HDRQN [14], which is a non-CTDE method, and extend QMIX (QMIX-Multi), MAPPO [39] (MAPPO-Multi) and VDN [2] (VDN-Multi) in a multi-task setting to train a unified multi-task multi-agent policy. In addition, we use separate observation mapping functions and k buffers for training QMIX-Multi, MAPPO-Multi and VDN-Multi. Otherwise, those methods turn out to fail on some tasks in the multi-task setting.

D. Training Details and Hyperparameters

Each method is trained for 50k epochs, utilizing a batch size of 32 (close to 1.25M time steps per task in MPE and close to 1.5M time steps in SMAC) and evaluated for 100 episodes to report the average rewards. In MKT-MARL, the annealing coefficient λ is linearly increased from 0.1 to 0.95 and fixed to 0.95 in the last 20k epochs. Training MKT-MARL on SMAC using a GeForce RTX 2080 Ti takes approximately 16 hours to complete 50k epochs of end-to-end training. The training hyperparameters are summarized in Table III. The epsilon greedy was chosen as the agent's exploration strategy, which is a commonly used algorithm for balancing exploration and exploitation in reinforcement learning. The Adam optimizer was chosen as the optimization algorithm for the stochastic gradient descent optimization algorithm. Hyperparameters were fine-tuned using a random search method that involved testing various combinations based on estimations and empirical results. The whole training process for the two benchmarks is shown in Figure 5 and Figure 6, respectively. Using five different random seeds, the experiments are conducted, and the results are presented along with a 95% confidence interval. Table I and Table II show the final average performance of the 5 random seeds. The task-specific teachers are fully trained on each task. The update frequency d for the target multi-task agents network and the target mixing network is set to every 200 epochs. The temperature parameter t in Eq. 3 is assigned a value of 1.

VI. EXPERIMENTS

In this section, we begin by analyzing the experimental results from diverse perspectives. Next, we compare the performance improvements attributed to different modules and offer potential insights. Lastly, we experimentally illustrate the effectiveness of our algorithm with a varying number of agents in the group.

A. Comparison to Baselines

Referring to the learning curves in Figure 5 and Figure 6 and the corresponding final average performance in Table I and

Table II, we can see that MKT-MARL outperforms its strong single-task teachers and the jointly-trained team of agents using state-of-the-art algorithms in terms of both efficiency and ultimate performance. Based on these results for the two benchmarks, we can make the following observations:

Beat single-task learning on all tasks. Even though the multi-task agents of MKT-MARL have a close network size to the single-task agents, they can beat the single-task agents on all tasks as seen in Table I and Table II. For example, in a set of multiple tasks at SMAC, MKT-MARL exceeds single-task agents by 0%, 11%, and 6% in the MMM, MMM2, and 10m vs 11m scenarios, respectively. We believe that the common coordinated knowledge learned from different tasks helps the agents to be able to surpass the task-specific teachers. Moreover, MKT-MARL also has obvious advantages in other SMAC tasks, which illustrates the effectiveness of our algorithm. Related works [12], [13], [18], [40] have also observed similar findings, highlighting that multi-task models can outperform single-task teachers. [40] propose that the distillation method promotes increased exploration of states by agents compared to teachers, thereby potentially resulting in improved performance.

Beat other baselines on most tasks. Our algorithm outperforms other baselines in most scenarios, including all SMAC tasks and most MPE tasks. From the performance in SMAC in Table I we can see that our algorithm outperforms all other algorithms. Only on very few tasks DEC-HDRQN or MAPPO or QMIX-Teacher has a flat performance. The poor performance of MAPPO-Multi and VDN-Multi on several tasks at SMAC is intriguing because it highlights their vulnerability in the face of significantly different tasks. Due to the limited number of online samples, MAPPO may have difficulty in learning effective feature representations in this case. VDN-Multi may be that its simple linear mixing network is not able to characterize the feature space well. Those determine whether they can get high scores or not. Furthermore, Figure 5(d) and Figure 5(e) demonstrate that the performance of DEC-HDRQN is constrained by the task-specific teacher policy, resulting in fluctuating performance below that of the teacher policy. This is due to the fact that DEC-HDRQN only considers the policy distillation from teacher policy. While our algorithm does not have this limitation and exceeds the task-specific teacher.

Effectiveness of two module learning. Referring to the learning curves in Figure 5 and Figure 6, the multi-task agents in our MKT-MARL approach exhibit rapid learning, achieving proficient performance across diverse tasks within approximately 10K epochs. During this period, supervised knowledge transfer is given major weight in the loss, allowing student policy to learn quickly from the task-specific teachers. Then, due to the teacher annealing strategy, the weight of loss depends on environmental feedback and the multi-task agent is further improved during teacher annealing. This observation effectively demonstrates the importance and effectiveness of supervised knowledge transfer and adaptive TD-learning.

TABLE IV
THE FINAL AVERAGE PERFORMANCE ON SMAC AND MPE IN THE ABLATION STUDY.

	Avg.Win Rate (SMAC)	Avg. Score (MPE)
QMIX-Teacher	0.79	-7.74
w/o distillation ($\lambda = 1$)	0.71	-8.57
w/o TD learning ($\lambda = 0$)	0.78	-7.15
w/o annealing ($\lambda = 0.5$)	0.81	-7.82
annealing reverse ($-1 * \lambda$)	0.78	-7.62
MKT-MARL(our)	**0.83**	**-6.49**

B. Ablation Study

We conduct a comprehensive experimental analysis of ablation study for two key modules in our algorithm, including the supervised knowledge transfer module (Section IV-B) and the adaptive TD-learning module (Section IV-D), and we analyzed the effectiveness of the teacher annealing strategy. As in the experiments above, we validate their effectiveness and contribution in both the SMAC and MPE benchmarks. Each method is trained for 50k epochs, and the average performance is evaluated over 100 episodes. As shown in the Table IV, the average performance on all maps was calculated for comparison. We control the ablation experiment by varying the annealing coefficient in the teacher annealing. a = 1, the loss contains only TD loss and trains the multi-agent policy only through the environmental feedback. a = 0, the loss contains only the distillation loss and trains the multi-agent policy only through the task-specific teachers. a = 0.5, the loss contains both the TD loss and distillation loss, but both weights are fixed from the beginning to the end. $-1 * \lambda$, gradually increasing the weight of supervised knowledge transfer. From these results, we can see that our algorithm eventually outperforms the task-specific teachers and other schemes.

Effectiveness of the supervised knowledge transfer. To assess the impact of supervised knowledge transfer, we train the multi-task agents using adaptive TD-learning without incorporating the distillation method. This training configuration is referred to as "w/o distillation". Based on the K separate replay buffer, the multi-agents sample the data from all the tasks to update the whole network by Eq. 5 due to the environmental feedback throughout the training process. From the performance in Table IV, we can see the supervised knowledge transfer improves the average performance of SMAC by 16.9% and MPE by 24.2%. After using the supervised knowledge transfer, the performance of the multi-task agents has approached that of the single-task teachers. Without TD-learning, directly using teacher supervision will constrain the model's final performance.

Effectiveness of the adaptive TD-learning. This solution is labeled as "w/o TD learning". From the performance in the tables, we can see that adaptive TD-learning improves the average performance by 6.4% in SMAC and by 9.2% in MPE. In summary, using the adaptive TD-learning alone has poorer performance than using the supervised knowledge transfer alone, probably because the interference of multiple

TABLE V
THE FINAL AVERAGE WIN RATE FOR ADAPTING TO NEW TASKS ON SMAC.

	MKT-MARL (ours) CTDE	Dec-HDRQN non-CTDE	QMIX-Multi CTDE	MAPPO-Multi CTDE	VDN-Multi CTDE
1s7z (source)	**0.91**	0.77	**0.91**	0.51	0
2s6z (source)	**0.96**	0.79	0.77	0.75	0
3s5z (source)	**0.95**	0.81	0.85	0.39	0.01
4s4z (target)	**0.9**	0.71	0.89	0.28	0
5s3z (target)	**0.94**	0.75	0.8	0.65	0
6s2z (target)	**0.91**	0.84	0.82	0.34	0

tasks makes it difficult for multi-task agents to acquire good strategies from scratch. However, the use of the adaptive TD-learning allows multi-task agents that approach the performance of task-specific teachers to outperform the teachers.

Effectiveness of teacher annealing learning. In addition, we investigate the effectiveness of teacher annealing, we trained the multi-agent policy with a fixed annealing factor of 0.5, which makes the agents affected by both teacher policy and environmental feedback from beginning to end, which is labeled as "w/o annealing". We also trained the multi-agent policy to gradually increase its weight for supervised knowledge transfer, which is labeled as "annealing reverse". The experimental results indicate that only when Supervised Knowledge Transfer had a relatively high weight in the early stages and TD-learning had a higher weight in the later stages, the model achieved the best results. The possible reason for this is that the model's capacity is limited [13], and if it is initially exposed to a large amount of poor-quality data, even with good-quality data guiding it later on, it becomes challenging to correct the model effectively.

C. Adapt to New Tasks

We conducted experiments to assess the generalization of our trained model on new tasks, and the results are shown in Table V. We trained the model on three original tasks, 1s7z, 2s6z, and 3s5z (denoted as source tasks), and directly tested it on three new tasks, 4s4z, 5s3z, and 6s2z (denoted as target tasks). From the experimental results, it can be seen that when directly applying the model trained on the original tasks to the target environment, the model performance is degraded and also lower than when the model is trained directly in the target environment. However, when compared to other algorithms, our approach demonstrates stronger generalization and outperforms them. This could be attributed to acquiring more universal patterns of collaboration through feedback from both the teacher and the environment.

D. Multi-Agent in Variable Scenarios

Our approach can be used for a different number of agent tasks by combining adaptive mixing networks, such as VDN and transformer-based mixing networks [41]. We implemented the VDN version of our method named MKT-MARL-VDN and trained it on several different numbers of agents tasks. We chose three scenarios with 3 agents (3m), 5 agents (2s3z), and 8 agents (8m) to train together, and the final performance compared with the task-specific teacher is shown in Table VI. Based on the results, our algorithm demonstrates comparable performance to the task-specific teacher across scenarios with varying numbers of agents and outperforms VDN-Multi in all scenarios. This suggests that our algorithm effectively captures the similarities in collaboration patterns across tasks involving different numbers of agents.

TABLE VI
THE FINAL AVERAGE PERFORMANCE OF SMAC WITH DIFFERENT NUMBERS OF AGENTS.

	3m	2s3z	8m
QMIX-Teacher	1.0	0.97	1.0
VDN-Multi	0.85	0.81	0.48
MKT-MARL-VDN	0.99	0.94	0.74

VII. CONCLUSION AND FUTURE WORK

In this paper, we present MKT-MARL, an algorithm that trains a team of MARL agents to attain expert-level performance across multiple tasks. In MKT-MARL, the agents are trained through two modules of learning, the supervised knowledge transfer module, which distills knowledge from task-specific teachers to multi-task agents, and the adaptive TD-learning module, which allows the model to learn from the environmental feedback. And the two modules are connected by the teacher annealing strategy, which transfers the model learning from teachers to environmental rewards. We conduct extensive experiments with two popular benchmarks in SMAC and MPE. The experimental results show that 1) MKT-MARL outperforms single-task learning in terms of both efficiency and ultimate performance across the majority of tasks; 2) MKT-MARL beats other state-of-the-art baselines on most tasks, with significant advantages on some tasks; 3) both the two modules of learning and the teacher annealing strategy are effective in our algorithm. Note that we assume all the agents are fully cooperative, or MKT-MARL may struggle to attain an effective policy. To the best of our knowledge, the non-fully cooperative multi-task problem is also important, and raises an open question about the learnability of policies. The scope of our discussion does not cover this challenging problem, and we consider it as a subject for future research.

ACKNOWLEDGMENTS

This research was supported by National Key R&D Program of China (2022ZD0116403).

REFERENCES

[1] M. Samvelyan, T. Rashid, C. S. De Witt, G. Farquhar, N. Nardelli, T. G. Rudner, C.-M. Hung, P. H. Torr, J. Foerster, and S. Whiteson, "The starcraft multi-agent challenge," *arXiv preprint arXiv:1902.04043*, 2019.

[2] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls *et al.*, "Value-decomposition networks for cooperative multi-agent learning," *arXiv preprint arXiv:1706.05296*, 2017.

[3] 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 *International Conference on Machine Learning*. PMLR, 2018, pp. 4295–4304.

[4] D. Ye, Z. Liu, M. Sun, B. Shi, P. Zhao, H. Wu, H. Yu, S. Yang, X. Wu, Q. Guo *et al.*, "Mastering complex control in moba games with deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 6672–6679.

[5] Y. Hu, A. Nakhaei, M. Tomizuka, and K. Fujimura, "Interaction-aware decision making with adaptive strategies under merging scenarios," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 151–158.

[6] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine, "Learning to walk via deep reinforcement learning," *arXiv preprint arXiv:1812.11103*, 2018.

[7] J. Foerster, I. A. Assael, N. De Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," *Advances in neural information processing systems*, vol. 29, 2016.

[8] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *International conference on autonomous agents and multiagent systems*. Springer, 2017, pp. 66–83.

[9] 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 *International Conference on Machine Learning*. PMLR, 2019, pp. 5887–5896.

[10] B. Goertzel and C. Pennachin, *Artificial general intelligence*. Springer, 2007, vol. 2.

[11] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, "Gradient surgery for multi-task learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5824–5836, 2020.

[12] E. Parisotto, J. L. Ba, and R. Salakhutdinov, "Actor-mimic: Deep multitask and transfer reinforcement learning," *arXiv preprint arXiv:1511.06342*, 2015.

[13] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell, "Policy distillation," *arXiv preprint arXiv:1511.06295*, 2015.

[14] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," in *International Conference on Machine Learning*. PMLR, 2017, pp. 2681–2690.

[15] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[16] L. T. Liu, U. Dogan, and K. Hofmann, "Decoding multitask dqn in the world of minecraft," in *The 13th European Workshop on Reinforcement Learning (EWRL)*, vol. 2016, 2016.

[17] Y. Teh, V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, and R. Pascanu, "Distral: Robust multitask reinforcement learning," *Advances in neural information processing systems*, vol. 30, 2017.

[18] Z. Xu, K. Wu, Z. Che, J. Tang, and J. Ye, "Knowledge transfer in multi-task deep reinforcement learning for continuous control," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15 146–15 155, 2020.

[19] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine learning*, vol. 3, pp. 9–44, 1988.

[20] H. Yin and S. J. Pan, "Knowledge transfer for deep reinforcement learning with hierarchical experience replay," in *Thirty-First AAAI conference on artificial intelligence*, 2017.

[21] W. Wang, T. Yang, Y. Liu, J. Hao, X. Hao, Y. Hu, Y. Chen, C. Fan, and Y. Gao, "From few to more: Large-scale dynamic multiagent curriculum learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 7293–7300.

[22] A. Agarwal, S. Kumar, and K. Sycara, "Learning transferable cooperative behavior in multi-agent teams," *arXiv preprint arXiv:1906.01202*, 2019.

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

[24] R. S. Sutton, *Temporal credit assignment in reinforcement learning*. University of Massachusetts Amherst, 1984.

[25] J. Wang, Z. Ren, T. Liu, Y. Yu, and C. Zhang, "Qplex: Duplex dueling multi-agent q-learning," *arXiv preprint arXiv:2008.01062*, 2020.

[26] K. E. Kinnear, W. B. Langdon, L. Spector, P. J. Angeline, and U.-M. O'Reilly, *Advances in genetic programming*. MIT press, 1994, vol. 3.

[27] J. Wang, Y. Zhang, T.-K. Kim, and Y. Gu, "Shapley q-value: A local reward approach to solve global reward games," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 7285–7292.

[28] M. Zhou, Z. Liu, P. Sui, Y. Li, and Y. Y. Chung, "Learning implicit credit assignment for cooperative multi-agent reinforcement learning," *arXiv preprint arXiv:2007.02529*, 2020.

[29] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[30] L. S. Shapley, "Stochastic games," *Proceedings of the national academy of sciences*, vol. 39, no. 10, pp. 1095–1100, 1953.

[31] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[32] R. A. Howard, "Dynamic programming and markov processes." 1960.

[33] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[34] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[35] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[36] F. A. Oliehoek and C. Amato, *A concise introduction to decentralized POMDPs*. Springer, 2016.

[37] G. Papoudakis, F. Christianos, L. Schäfer, and S. V. Albrecht, "Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

[38] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser *et al.*, "Starcraft ii: A new challenge for reinforcement learning," *arXiv preprint arXiv:1708.04782*, 2017.

[39] C. Yu, A. Velu, E. Vinitsky *et al.*, "The surprising effectiveness of mappo in cooperative," *Multi Agent Games*, 2021.

[40] W. M. Czarnecki, R. Pascanu, S. Osindero, S. Jayakumar, G. Swirszcz, and M. Jaderberg, "Distilling policy distillation," in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 1331–1340.

[41] T. Zhou, F. Zhang, K. Shao, K. Li, W. Huang, J. Luo, W. Wang, Y. Yang, H. Mao, B. Wang *et al.*, "Cooperative multi-agent transfer learning with level-adaptive credit assignment," *arXiv preprint arXiv:2106.00517*, 2021.