CTDS: Centralized Teacher With Decentralized Student for Multiagent Reinforcement Learning

Jian Zhao, Xunhan Hu, Mingyu Yang, Wengang Zhou, Senior Member, IEEE, Jiangcheng Zhu, and Houqiang Li, Fellow, IEEE

Abstract—Due to the partial observability and communication constraints in many multiagent reinforcement learning (MARL) tasks, centralized training with decentralized execution (CTDE) has become one of the most widely used MARL paradigms. In CTDE, centralized information is dedicated to learning the allocation of the team reward with a mixing network while the learning of individual Q-values is usually based on local observations. The insufficient utility of global observation will degrade performance in challenging environments. To this end, this work proposes a novel Centralized Teacher with a Decentralized Student (CTDS) framework, which consists of a teacher model and a student model. Specifically, the teacher model allocates the team reward by learning individual Q-values conditioned on global observation while the student model utilizes the partial observations to approximate the Q-values estimated by the teacher model. In this way, CTDS balances the full utilization of global observation during training and the feasibility of decentralized execution for online inference. Our CTDS framework is generic, which is ready to be applied upon existing CTDE methods to boost their performance. We conduct experiments on a challenging set of StarCraft II micromanagement tasks to test the effectiveness of our method and the results show that CTDS outperforms the existing value-based MARL methods.

Index Terms—Multiagent system, reinforcement learning.

I. INTRODUCTION

COOPERATIVE multiagent reinforcement learning (MARL) has been demonstrating significant applicability in a variety of domains, such as autonomous cars [1], sensor networks [2], robot swarms [3], [4], and games [5], [6]. To train a MARL system, a group of agents coordinate with each other to learn action-values conditioned on state information and jointly optimize a single-reward signal, i.e., the team reward, accumulated [7], [8].

A naive solution to MARL is to convert a cooperative multiagent problem into a single-agent RL problem by taking the joint state/action space of all the agents as the state/action space of one virtual agent [9], [10], [11], which is usually viewed as the paradigm of centralized training with centralized execution (CTCE). Despite its effectiveness, cooperative MARL with centralized execution encounters a major challenge of scalability, i.e., the joint state-action space grows exponentially as the number of agents increases. Besides, in many real-world settings, due to observation and communication constraints, it becomes impractical to make centralized execution [12]. Another alternative approach is the paradigm of decentralized training with decentralized execution (DTDE), which trains independent learners to optimize for the team reward. However, it is difficult to design effective individual reward functions for different agents when only the team reward is available. In addition, decentralized training neglects the coordination among agents, which is essential for multiagent systems. Compared to decentralized training, training the agents in a centralized fashion allows access to global information and removes interagent communication constraints, which is beneficial to capture the coordination pattern and allocate the team reward.

Considering the necessity of decentralized policies and the benefits of centralized training in MARL, the paradigm of centralized training with decentralized execution (CTDE) [12], [13], [14] has gained increasing attention in the last few years. Fig. 1 illustrates the framework of CTCE, DTDE, and CTDE. In this paradigm, the policy of each agent is trained in a global context
in a centralized way and executed only based on local histories in a decentralized way. There are two nontrivial issues in CTDE: credit assignment and partial observation. Credit assignment refers to allocating the team reward to each agent by leveraging the global state information during the centralized training while partial observation indicates the fact that each agent can only access a proportion of the global state during inference.

To address the above two issues, many CTDE learning approaches [7], [12], [15], [16], [17], [18], [19] have been proposed recently, among which value-based MARL algorithms [7], [12], [17], [18], [19] have shown the state-of-the-art performance on challenging tasks, e.g., unit micromanagement in StarCraft II [20]. In existing value-based MARL algorithms, to adapt to the constraint of agents’ partial observation during the inference stage, they make the same state setting for each agent during training. In this way, for each agent, its individual Q-value is conditioned only on individual local observation and its action. To take advantage of the global state and team reward, a mixing network is designed to aggregate the individual Q-values of all agents so as to approximate the team reward following the Individual-Global-Max (IGM) principle [17]. Following this paradigm, there are many algorithms with different designs on the mixing network. For instance, the value decomposition network (VDN), the first attempt in this field, adopts the summation operation [7]. Differently, QMIX [12] assigns the nonnegative weights to individual Q-values with a nonlinear function of the global state. Later, more and more algorithms employ the variants of the attention mechanism as the aggregation function [17], [18], [19]. The algorithms discussed above all suffer an essential limitation, i.e., the individual Q-value estimation of all agents fails to consider the global state, which makes them difficult to learn tacit coordination. This is due to the information gap between the training phase and the inference phase. In other words, during training, the global state is available to all agents. But in inference, only local and partial observation is accessible to each agent.

To bridge the above gap, in this work, we propose a novel framework, named Centralized Teacher with Decentralized Student (CTDS), with a teacher model and a student model. Specifically, the teacher model estimates the individual Q-value estimation of agents based on the global state while the student model regresses each agent’s Q-value with only local and partial observation. With a global state, the teacher model is likely to learn more tacit coordination among agents, which is transferred to the student model with a knowledge distillation strategy. After the training, the student model is ready to be used for online inference with only local observation. It is notable that our CTDS framework is generic and is ready to be applied to various existing CTDE methods to boost their performance.

We evaluate our CTDS on a grid world environment Combat and a range of unit micromanagement tasks built-in StarCraft II. Our experiments show that CTDS significantly outperforms the recent representative CTDE methods, in terms of both performance and learning speed. Moreover, we also investigate the performance of CTDS under the scenario with different sight ranges, which further illustrates the robustness of our method.

| Scheme | Approach | Methods |
|--------|----------|---------|
| DTDE   | policy-based | IPPO [28] |
|        | value-based | IQL [27] |
| CTDE   | policy-based | MAPPO [29], DOP [30], MADDPG [15] |
|        | value-based | VDN [7], QMIX [12], QTRAN [17], QATTEN [18], QPLEX [19] |
| CTCE   | policy-based | HATRPO [25], HAPPO [25], MAT [26] |
|        | value-based | CommNet [23], BiCNet [24] |

II. RELATED WORK

A. Cooperative MARL Methods

In a cooperative multiagent system, a team of agents coordinates with each other and obtains an overall team reward. Due to its wide application in practice, cooperative MARL has gained substantial research interests [21], [22]. The overview of current mainstream training schemes applied in MADRL works is shown in Table I.

In view of the coordination among agents, early efforts focus on centralized learning and execution [23], [24], [25], [26], in which agents can communicate with each other to access more observation of the environment. Among them, the value-based methods, such as CommNet [23] and BiCNet [24] mainly obtain more information through the communication between agents and then use the single-agent value-based method to generate policies. The policy-based methods, such as HATRPO [25], HAPPO [25], and MAT [26], utilize the multiagent advantage decomposition theorem to transform the joint policy search problem into a sequential decision-making process.

However, in many real-world scenarios, the partial observability and/or communication constraints among agents necessitate decentralized training and execution. An intuitive approach is to adopt the DTDE paradigm, i.e., use the team reward as an individual reward, and each agent is trained according to the method of a single agent. Representative work includes IQL [27], IPPO [28], etc.

In more cases, agents can communicate with each other or obtain global information during training, but they cannot during execution. At this time, a paradigm of CTDE has been widely adopted recently. Among them, the policy-based method, such as MAPPO [29], DOP [30], and MADDPG [15], often adopts the actor–critic structure, the policy network uses individual information as input, and the value network uses global information as input. In this way, the centralized value network can be used to calculate the advantage to assist training, and only need to use the decentralized policy network to take actions during execution.

At present, the most popular topic of MARL is the value-based CTDE algorithm. In the following, we introduce some representative methods. To the best of our knowledge, the VDN [7] is the first attempt to allow for centralized value-function learning with decentralized execution. This work assumes that each agent contributes equally to the team reward, and thus, a central
action-value function is decomposed into a sum of individual $Q$-values conditioned only on local observations. While achieving decentralization, VDN obtains a suboptimal solution by simple summation integration and ignores the available global state information during training.

To distinguish the importance of different agents, QATTEN [18] utilizes a multihed attention structure to weigh the individual $Q$-values based on the global state and the individual features and then linearly integrates these values into the central action value. Unlike VDN and QATTEN, which take linear monotonic value functions, QMIX [12], [31] relaxes this assumption and employs a mixing network by leveraging state information to decide the transformation weights, which allows nonlinearity operations to decompose the central $Q$-value. Later on, QTRAN [17] relaxes the monotonicity restriction in QMIX [32] and proposes a factorization method to express function space induced by IGM consistency. The computational intractability, however, renders the poor performance of QTRAN in complex tasks. QPLEX [19] decomposes the central $Q$-value into the sum of individual value functions and a nonpositive advantage function, which is 0 only when all of the agents choose the optimal actions. By introducing the duplex dueling structure, QPLEX achieves the complete function class that satisfies the IGM principle.

In this article, we study the value-based MARL methods under the CTDE paradigm. As far as we know, all the existing value-based CTDE methods only utilize local observations instead of global observations for learning individual $Q$-values during training in order to achieve decentralized execution. Intuitively, in a cooperative multiagent system, it is beneficial when knowing more about other agents. Inspired by the benefit of global observation, we adopt the concept of knowledge distillation [33], [34].

B. Knowledge Distillation

Knowledge distillation is first proposed for network compression [35], which distills the knowledge (i.e., the output distribution) from a teacher model that is typically large into a smaller student model so that the student model can achieve similar performance as the teacher model. Gradually, the concept of knowledge distillation has been generalized into a framework consisting of a teacher model and a student model, where the student model aims to imitate the teacher model under its guidance. In other words, knowledge distillation suggests training by matching the student’s predictions to the teacher’s predictions [36], [37]. Due to the advantage of fast optimization, network minimization and comparable performance [37], [38], the technique of knowledge distillation has been widely applied in various ways [36], [39], [40], [41]. In [42], knowledge distillation is first leveraged to reinforcement learning to distill the policies into a dramatically smaller and more efficient network. The experiments show that the distilled agents outperform their teachers in most games, which indicates the effectiveness of knowledge distillation in reinforcement learning tasks.

By leveraging knowledge distillation to MARL, we aim to take full advantage of full observation during training with a teacher module. To allow decentralized execution, we design the student module, which approximates individual $Q$-values based on local observations, to distill the $Q$-values estimated by the teacher module. In this way, tacit coordination can be learned among the agents on the student side.

III. PRELIMINARIES

A. Decentralized Partially Observable MDP

A cooperative multiagent problem under the partial observability can be modeled as DEC-POMDP [14], [43], which is defined as a tuple $M = \langle N, S, A, \Pi, \Omega, O, R, \gamma, \gamma >$, where $N = \{g_1, g_2, \ldots, g_N\}$ is a finite set of agents and $S$ describes a set of the global state $s$ of the environment. At each time step, every agent $g_i \in N$ receives an individual partial observation $o_i \in \Omega$ according to the observation probability function $O_i(o_i|s)$. Each agent $g_i$ has an action-observation history $\tau_i \in \tau = (\Omega \times A)^{\tau}$ and its individual policy $\pi_i(a_i|\tau_i)$. According to $\tau_i$, each agent $g_i$ chooses an action $a_i \in A_i$, which forms a joint action $a = (a_1, \ldots, a_N).$ After the execution of all agents, it results in a team reward $r(s, a)$ and a transition to the next global state $s' \sim P(\cdot|s, a)$ by the environment. $\gamma \in [0,1)$ is a discount factor. The formal objective is to find a joint policy $\pi = (\pi_1, \ldots, \pi_n >$ that maximizes a joint state value function $V^\pi(s) = \mathbb{E}_{\sum_{i=1}^N \gamma^i \tau_i|s_0 = s, \pi}$. In addition, the joint action-value function $Q^\pi(s, a) = r(s, a) + \gamma \mathbb{E}_s[V^\pi(s')]$ often appears in algorithm expressions to replace the joint state value function $V^\pi(s)$.

B. Deep Multiagent Q-Learning of CTDE Paradigm

CTDE is a popular paradigm of cooperative MARL [7], [12], [14], [17], [18], [19]. In CTDE, each agent can access the global information during the centralized training, while each agent takes actions only based on local action-observation histories during the decentralized execution. Multiagent Q-learning algorithms represent the individual action-value function of the agent $g_i$ with a neural network parameterized by $\theta_i$, denoted as $Q_i(\tau_i, a_i|\theta_i)$. Define $\Theta = [\theta_1, \ldots, \theta_N]$ as the parameters of all individual action-value functions. These algorithms combine the individual action-value $Q_i(\tau_i, a_i|\theta_i)$ with a mixing network $f$ to fit the joint action-value $Q_{tot}$, i.e.,

$$Q_{tot} = f(Q_1, Q_2, \ldots, Q_n, s|\phi)$$

where $\phi$ is the parameters of mixing network $f$. An important concept of CTDE framework is that the mixing network should satisfy the requirement that the optimal joint action induced from the optimal centralized action-value function is equivalent to the collection of individual optimal actions of agents, which is called IGM, such that the following holds:

$$\arg \max_{\alpha} Q_{tot}(\tau, \alpha) = \left[\arg \max_{a_1} Q_1(\tau_1, a_1), \ldots, \arg \max_{a_n} Q_n(\tau_n, a_n)\right].$$

Given the value decomposition, multiagent Q-learning algorithms use a replay memory $D$ to store the transition tuple
(\tau, a, r, r'), where \( r \) is the team reward for taking action \( a \) at joint action-observation history \( \tau \) with a transition to \( \tau' \). Based on the above discussion, parameters \( \theta \) and \( \phi \) are learned by minimizing the following expected TD error:

\[
L(\theta, \phi) = E_{(\tau, a, r, r') \in D}[(r + \gamma V(\tau'|\theta^-) - Q_{\text{tot}}(\tau; a|\theta, \phi))^2]
\]  

(3)

where \( V(\tau'|\theta^-) = \max_a Q_{\text{tot}}(\tau'|a'|\theta^-) \) is the one-step expected future return of the TD target. \( \theta^- \) and \( \phi^- \) are the parameters of the target network, which will be periodically updated with \( \theta \) and \( \phi \).

IV. METHOD

In this section, we first explain the motivation of our proposed framework and introduce the main components in detail. Besides, we give a theoretical analysis of why our method works.

A. CTDS Framework

In this section, we present a new framework called CTDS, which leverages the idea of knowledge distillation to balance the advantage of global observation and the requirement of decentralized execution.

Recall that in a cooperative multiagent system, a group of agents coordinate with each other to take joint actions based on the global state information and will obtain a team reward correspondingly. To achieve decentralized execution, the existing CTDE framework requires the individual \( Q \)-values in the lower layer only conditioned on the local action-observation histories and incorporates the state information into the mixing network to integrate the individual \( Q \)-values into the overall \( Q \)-value function. Obviously, if an agent could have the full observation of other agents, it would be more likely to learn a better policy under the cooperative scenario. Considering the benefit of centralized observations during training, our CTDS decouples the centralized training and decentralized execution with the combination of the teacher module and the student module, as illustrated in Fig. 2. The teacher module is dedicated to allocating the team reward through centralized training while the student module utilizes partial observations to approximate the individual \( Q \)-value estimated by the teacher model. In the following, we will give a detailed description of the design of the teacher module and student module, and the knowledge distillation mechanism between them, respectively.

1) Teacher Module: The teacher module consists of two parts: One is the \( Q \)-network for estimating individual action-value functions for each agent; another is the mixing network for integrating the individual \( Q \)-values to be the centralized \( Q \)-value. Note that the teacher module only participates in the centralized training.

The teacher module allows each agent to have an infinite sight range to receive the observation of all the agents. For each agent \( g_i \), the \( Q \)-network takes the centralized observation \( o_i^t \), last action \( a_i^{t-1} \), and last hidden state \( h_i^{t-1} \) as inputs, fits the perfect individual action-value \( \hat{Q}_i(\hat{\tau}_i, \cdot) \) through the multilayer perceptron (MLP) and gated recurrent unit (GRU) modules, and outputs the optimal action-value \( \hat{Q}_i(\hat{\tau}_i, a_i^{t}) \). Given the perfect individual action-value \( \hat{Q}_i(\hat{\tau}_i, a_i^{t}) \), the mixing network \( f \) combines these individual \( Q \)-values into the joint action-value function \( \hat{Q}_{\text{tot}} \) by utilizing the global state information. The parameters of the teacher module are updated by the TD loss defined in (3) iteratively.

2) Student Module: To achieve decentralized execution, the agent \( g_i \) in the student module only has access to partial observation \( o_i^t \), different from the full observation \( o_i^t \) in the teacher
Algorithm 1: CTDS Framework.

**Input:** the discount factor \(\gamma\), the number of agents \(N\), the maximum steps \(t_{\text{max}}\) and target network update period \(I\);

**Output:** the individual student network;

1: Initialize replay memory \(D\), set \(t_{\text{step}} = 0\);
2: Initialize the teacher network, target teacher network and student network with random parameters;
3: while \(t_{\text{step}} \leq t_{\text{max}}\) do
4:   For each episode, observe initial centralized observation \(o_i^{[0]}\) and initial decentralized observation \(o_i^{[0]}\);
5:   set \(t = 0\);
6:   while Not terminal do
7:      With probability \(\epsilon\) select a random action \(a_i^t\), otherwise \(a_i = \arg\max Q_i^{\hat{\tau}}(\hat{\tau}_i^t, a_i)\) for each agent \(i\);
8:      Take action \(a_i^t\) and receive reward \(r_i\), centralized observation \(o_i^{[t+1]}\) and decentralized observation \(o_i^{[t+1]}\);
9:      \(t \leftarrow t + 1\);
10: end while
11: \(t_{\text{step}} \leftarrow t_{\text{step}} + t\);
12: Insert the current episode data into buffer \(D\);
13: Sample a random episode minibatch of transitions \((\hat{\tau}, \tau, a, r, \hat{\tau}^t)\) from \(D\);
14: Set joint action-value
15: \(Q_{\text{tot}} = f([\hat{Q}_i(\hat{\tau}_i, a_i)]_{i=1}^N)\);
16: Set target joint action-value
17: \(Q_{\text{tot}}^- = f^-([\hat{Q}_i(\hat{\tau}_i, a_i)]_{i=1}^N)\),
where \(\hat{a}_i = \arg\max Q_i^{\hat{\tau}}(\hat{\tau}_i, a_i)\);
18: Update the teacher network parameters with TD error
19: \(L_{\text{TD}} = (r + \gamma Q_{\text{tot}}^- - Q_{\text{tot}})^2\);
20: Update the student network parameters with distillation loss
21: \(L_{\text{dis}} = (\hat{Q}_i(\hat{\tau}_i, a_i) - Q_i(\tau_i, a_i))^2\);
22: Update target network teacher parameters with period \(I\);
23: end while

---

module. Compared with the teacher module, the student module only contains the \(Q\)-network, which has the same network architecture as the one in the teacher module but the parameters of \(Q\)-networks are not shared. Similarly, the \(Q\)-network takes the partial observation \(o_i^t\), last action \(a_i^{t-1}\), and last hidden state \(h_i^{t-1}\) as inputs, and generates the individual action-value \(Q_i(\tau_i, \cdot)\) through the MLP and GRU modules. It can be seen that the student module does not involve any global information and has a relatively simple network structure.

3) Knowledge Distillation Mechanism: As introduced above, the teacher module learns to allocate the team reward reasonably by taking full advantage of the global information while the student module focuses on learning a reasonable individual local \(Q\)-value to realize decentralized execution. To bridge the teacher module and the student module, we employ the knowledge distillation mechanism, i.e., distilling the perfect action-value \(Q_i\) estimated by the teacher model to guide the student module to approximate the local \(Q\)-value. Specifically, we adopt a mean-squared error loss (MSE) to minimize the difference between the \(Q\)-values estimated by these two modules. The advantage of MSE is that it constrains the complete set of action-value in the student model [42]. Formally, the MSE loss is calculated as follows:

\[
L_{\text{MSE}} = \sum_{i=1}^N \sum_{a_i \in A} (\hat{Q}_i(\hat{\tau}_i, a_i) - Q_i(\tau_i, a_i))^2.
\]

4) Training and Execution: In the training phase, the teacher module generates the corresponding actions of each agent based on the centralized observations to interact with the environment. Formally, the interactive actions are the results of the \(\epsilon\)-greedy from the \(Q_i\), which means choosing random actions with probability \(\epsilon\), otherwise choosing actions with the maximal \(Q_i\). The parameters of the teacher module and the student module are updated at the same time through iterative interactions. The teacher module is optimized under the constraint of TD loss while the student module learns the parameters with MSE loss. The execution phase only relies on the decentralized student module, which utilizes the local action-observation records. To be concrete, each agent chooses a greedy action \(a_i^t\) at each timestamp with respect to the individual \(Q\)-value \(Q_i\) estimated by the student module. Therefore, the CTDS framework still meets centralized training and decentralized execution. For better understanding, we provide a detailed procedure of our framework below (see Algorithm 1).

B. Theoretical Analysis

In this section, we theoretically derive why the student module performs better than the execution module of CTDE with the equivalent input information.

As mentioned above, the teacher module learns from the centralized history trajectory \(\hat{\tau}_i\), which contains the decentralized information \(\tau_i\) (accessible to the student module) and the complement information \(\tau_i^\ast\) (we assume that \(\tau_i\) and \(\tau_i^\ast\) are independent). We also use implicit information to refer to the complement information \(\tau_i^\ast\) in the following discussion. In order to explain the relationship between input trajectories and output individual \(Q\)-values more clearly, we simplify the student module and teacher module into the following formulation:

\[
\begin{align*}
(\text{Teacher network}) & \quad f_T : (\tau_i^\ast) \rightarrow \hat{Q}_i \\
(\text{Student network}) & \quad f_S : \tau_i \rightarrow Q_i.
\end{align*}
\]

The student module of CTDS and the execution module of CTDE both employ decentralized information merely. Due to the limitation of partial trajectory, i.e., the lack of \(\tau_i^\ast\), the execution module of CTDE tends to make nonoptimal actions for the current state. The centralized trajectory releases the restriction above; however, it is usually unavailable during execution.
CTDS takes both sides into consideration. For a certain decentralized trajectory $\tau_i$, the teacher module can give a centralized $Q$-value $f_T(\tau_i, \tau^*_i)$ with any possible implicit knowledge $\tau^*_i$. If the expectation of the teacher module’s output on implicit information $\mathbb{E}_{\tau^*_i} f_T(\tau_i, \tau^*_i)$ is achieved, it can be used to instruct student module $f_S$ to reduce environment uncertainty caused by the partial trajectory. Nevertheless, approximating $\mathbb{E}_{\tau^*_i} f_T(\tau_i, \tau^*_i)$ directly needs a large number of centralized trajectory samples, which is computationally expensive.

The following theorem proves that CTDS framework successfully employs $\mathbb{E}_{\tau^*_i} f_T(\tau_i, \tau^*_i)$ to train the student module. The comprehensiveness and stability provided by the instruction from the teacher module are the principal cause of the student module’s performance promotion.

**Theorem 1:** For any decentralized trajectory $\tau_i$, the student module of CTDS $f_S(\tau_i)$ is instructed to approximate the expectation of a centralized $Q$-value function on implicit information $\mathbb{E}_{\tau^*_i} f_T(\tau_i, \tau^*_i)$.

**Proof:** With a fixed teacher network, the optimization target of the student network is

$$\arg \min_{\theta_S} \sum_i \mathbb{E}_{\tau^*_i}[f_S(\tau_i) - f_T(\tau_i, \tau^*_i)]^2 \tag{6}$$

where $\theta_S$ represents the parameters of the student module. According to the definition of expectation

$$\frac{\partial}{\partial \theta_S} \mathbb{E}_{\tau^*_i}[f_S(\tau_i) - f_T(\tau_i, \tau^*_i)]^2 = \frac{\partial}{\partial \theta_S} \int [f_S(\tau_i) - f_T(\tau_i, \tau^*_i)]^2 p(\tau^*_i) d\tau^*_i. \tag{7}$$

Note that only $f_S(\tau_i)$ in the integral correlates with $\theta_S$. We move the derivatives into the integral, i.e.

$$\frac{\partial}{\partial \theta_S} \int [f_S(\tau_i) - f_T(\tau_i, \tau^*_i)]^2 p(\tau^*_i) d\tau^*_i = \int \frac{\partial}{\partial \theta_S} [f_S(\tau_i) - f_T(\tau_i, \tau^*_i)]^2 p(\tau^*_i) d\tau^*_i = \int 2(f_S(\tau_i) - f_T(\tau_i, \tau^*_i)) p(\tau^*_i) \frac{\partial f_S(\tau_i)}{\partial \theta_S} d\tau^*_i. \tag{8}$$

Because the $\frac{\partial f_S(\tau_i)}{\partial \theta_S}$ and constant term is irrelevant to $\tau^*_i$, they can be removed from the integral

$$\int 2(f_S(\tau_i) - f_T(\tau_i, \tau^*_i)) p(\tau^*_i) \frac{\partial f_S(\tau_i)}{\partial \theta_S} d\tau^*_i = 2 \frac{\partial f_S(\tau_i)}{\partial \theta_S} \int (f_S(\tau_i) - f_T(\tau_i, \tau^*_i)) p(\tau^*_i) d\tau^*_i = 2 \frac{\partial f_S(\tau_i)}{\partial \theta_S} \left[ \int f_S(\tau_i) p(\tau^*_i) d\tau^*_i - \int f_T(\tau_i, \tau^*_i) p(\tau^*_i) d\tau^*_i \right] = 2 \frac{\partial f_S(\tau_i)}{\partial \theta_S} [f_S(\tau_i) - \int f_T(\tau_i, \tau^*_i) p(\tau^*_i) d\tau^*_i]. \tag{9}$$

The last equality in (9) holds because $f_S(\tau_i)$ is irrelevant to $\tau^*_i$ and the term after removing $f_S(\tau_i)$ from integral is $\int p(\tau^*_i) d\tau^*_i = 1$. Taking into consideration of the derivative of

$$[f_S(\tau_i) - \int f_T(\tau_i, \tau^*_i) p(\tau^*_i) d\tau^*_i]^2$$

$$2 \frac{\partial f_S(\tau_i)}{\partial \theta_S} \left[ f_S(\tau_i) - \int f_T(\tau_i, \tau^*_i) p(\tau^*_i) d\tau^*_i \right] = \frac{\partial}{\partial \theta_S} [f_S(\tau_i) - \int f_T(\tau_i, \tau^*_i) p(\tau^*_i) d\tau^*_i]^2. \tag{10}$$

Given the definition of expectation

$$\frac{\partial}{\partial \theta_S} \mathbb{E}_{\tau^*_i} [f_S(\tau_i) - f_T(\tau_i, \tau^*_i)]^2 = \frac{\partial}{\partial \theta_S} [f_S(\tau_i) - \mathbb{E}_{\tau^*_i} f_T(\tau_i, \tau^*_i)]^2. \tag{11}$$

To sum up, we have proved that

$$\frac{\partial}{\partial \theta_S} \mathbb{E}_{\tau^*_i} [f_S(\tau_i) - f_T(\tau_i, \tau^*_i)]^2 = \frac{\partial}{\partial \theta_S} [f_S(\tau_i) - \mathbb{E}_{\tau^*_i} f_T(\tau_i, \tau^*_i)]^2. \tag{12}$$

Therefore

$$\arg \min_{\theta_S} \sum_i \mathbb{E}_{\tau^*_i}[f_S(\tau_i) - f_T(\tau_i, \tau^*_i)]^2 = \arg \min_{\theta_S} \sum_i \mathbb{E}_{\tau_i}[f_S(\tau_i) - \mathbb{E}_{\tau^*_i} f_T(\tau_i, \tau^*_i)]^2 \tag{13}$$

which means for any decentralized trajectory $\tau_i$, the student module of CTDS $f_S(\tau_i)$ is instructed to approximate the expectation of a centralized $Q$-value function on implicit information $\mathbb{E}_{\tau^*_i} f_T(\tau_i, \tau^*_i)$.

**V. Experiments**

In this section, we conduct comprehensive experiments to demonstrate the effectiveness of our method. All of the following experiments are to prove that our approach is robust enough to work on a variety of tasks. Our codes are available at: https://github.com/cathyhxh/CTDS.

**A. Environment Introduction**

In this article, we test our method on three environments: 1) Combat (a grid world environment introduced in [44]), 2) StarCraft II micromanagement (SMAC) [20], and 3) Google Research Football (GRF) [45]. These three environments are visualized in Fig. 3. Here, we introduce the details of these three environments.
1) Combat: In a 15 × 15 map, there are two opposing teams containing five agents, respectively. One team is controlled by our model and another one is controlled by a built-in AI. At each time step, agents can move to one of four adjacent grids or attack one of the enemies in sight range or do nothing. The agents are not allowed to attack in the next time step after an attack, which means cool-down. Each agent enjoys three health points at the beginning of an episode. When attacked, an agent’s health points will be reduced by 1. And the agent will die if its health point reaches 0. If the agents on the opposing team are all dead, then our team wins. The observation of an agent consists of its unique ID, team ID, location, health points, cool-down, and the enemies in sight range. The model gets a reward of −1 if the team loses or draws at the end of the game. In addition, it also gets a reward of −0.1 times the total health points of the enemy team, which encourages it to attack enemy bots.

2) SMAC: The StarCraft Multi-Agent Challenge (SMAC) [20] is currently a mainstream cooperative multiagent environment with partial observability. In this article, we use the default environment setting for SMAC with version SC2.4.10. There are various scenarios supported by SMAC and each scenario contains a confrontation between two teams of units, where one is controlled by a trained model while the other is controlled by a built-in strategy. The initial position, number, type of units in each team, and the map terrain vary in different scenarios. The details of each map can be found in [20]. In the SMAC environment, a team wins when all the units in the opposite team are killed. The overall objective of the policy is to maximize the win rate among all scenarios. The environment provides a shaped reward function to assist the training, which includes hit-point damage caused and received by agents, the units killed, and the battle result. This environment is highly demanding for agents to learn a coordinated micromanagement strategy in different maps.

3) GRF: GRF [45] is a challenging MARL environment, where a team of agents has to learn how to pass the ball among themselves and how to overcome their opponent’s defense in order to score goals. Similar to the SMAC environment, the opponent team is also controlled by a built-in strategy. There are 19 different actions to choose from each agent, including standard move actions (in eight directions), and different ways to kick the ball (short and long passes, shooting, and high passes that cannot be easily intercepted along the way). The observation of agents contains the position and moving direction of itself, other agents, and the ball. An episode terminates after limited steps or when one of the teams scores and the winning team will receive a +100 reward while the other gets a −1 reward. In this work, we conduct experiments on the scenario academy_3_vs_1_with_keeper and academy_counterattack_easy. The scenario details are introduced in [45].

B. Environment Setting

In the experiment implementation, the difference between partial observation and perfect observation depends on the sight range setting. Partial observation restricts the agents from receiving information about allied or enemy units that are out of range, i.e., the features of enemies and allies are all unknown when the corresponding unit is beyond the sight range. For example, the sight range of each agent is set to 2, which means each agent is allowed to only access its own features and other units’ features within a distance of 2. For the teacher module, perfect observation ϕ_t is not limited by the sight range, that is to say, each agent can access the full observation of all the agents. Fig. 4 shows the examples of the agent with partial and global observations in the SMAC, respectively. In SMAC, we set the partial sight range as 2 and the perfect sight range as infinity. In Combat, the partial sight range is 2 and the perfect sight range is 4. In GRF, the partial sight range is set as 0.4 and the perfect sight range is set as infinity.

The algorithms we use are based on the Pymarl algorithm library [20]. To reduce randomness, all the reported results are averaged over 5 runs with different seeds. Training and evaluation schedules such as the testing episode number and training hyperparameters are kept the same as default hyperparameters in Pymarl. Following the base models [7], [12], [19], we share the parameters of the agent networks across all agents in the teacher/student module to speed up the learning. Note that the parameters of the teacher module and student module are not shared. All the parameters in the neural networks are adjusted by an RMSprop optimizer with a learning rate of 0.0005, the same as that used by the baselines. For all the compared methods, each task is trained for 2 Million steps separately in SMAC and 500,000 steps in Combat.

C. Performance Evaluation and Discussion

Our CTDS framework can be applied to various existing models that follow the CTDE paradigm. To show the effectiveness of our CTDS framework, we conduct the experiments upon three representative models with different mixing network settings: VDN [7], QMIX [12], and QPLEX [19].

First, we conduct the experiment in grid world environment Combat. Fig. 5 reveals the test win rate curves of CTDS and baselines under the settings of the mixing neural network of QMIX and QPLEX after 500,000 steps of training. As demonstrated in Fig. 5, the CTDE paradigm improves convergence speed at the beginning of training and reaches the converging win rate with much lower variance.

Second, we present the experiment results in StarCraft II. Table II shows the final performance in terms of the test win
TABLE II
PERFORMANCE OF THE TEST WIN RATE PERCENTAGE (INCLUDING MEAN AND STANDARD DEVIATION) OF DIFFERENT BASE MODELS UNDER DIFFERENT SCENARIOS OF StarCraft II

| Scenario          | VDN[7]          | QMIX[12]        | QPLEX[19]       |
|-------------------|-----------------|-----------------|-----------------|
|                   | Baseline | Teacher | Student | Baseline | Teacher | Student | Baseline | Teacher | Student |
| 5m_vs_6m          | 6.9 ± 0.2 | 73.8 ± 0.9 | 55.0 ± 0.5 | 21.2 ± 0.4 | 66.9 ± 0.5 | 48.1 ± 1.0 | 23.8 ± 1.4 | 62.5 ± 0.9 | 56.2 ± 3.7 |
| 9m_vs_11m         | 0.6 ± 0.0 | 9.4 ± 0.6 | 12.5 ± 1.1 | 6.9 ± 0.1 | 9.4 ± 0.2 | 11.2 ± 0.1 | 3.1 ± 0.1 | 6.0 ± 0.0 | 6.0 ± 0.0 |
| 27m_vs_30m        | 0.0 ± 0.0 | 6.2 ± 0.1 | 1.9 ± 0.0 | 8.8 ± 1.0 | 58.8 ± 2.4 | 42.5 ± 1.5 | 0.0 ± 0.0 | 60.9 ± 5.8 | 49.2 ± 8.9 |
| MMM2              | 0.0 ± 0.0 | 15.6 ± 1.2 | 2.5 ± 0.1 | 20.6 ± 1.2 | 80.6 ± 1.7 | 28.1 ± 1.7 | 0.0 ± 0.0 | 26.6 ± 0.8 | 1.6 ± 0.1 |
| 2c_vs_64zg        | 44.4 ± 4.7 | 70.6 ± 3.6 | 37.5 ± 2.6 | 69.4 ± 2.6 | 82.5 ± 0.4 | 80.6 ± 0.7 | 81.9 ± 0.6 | 93.1 ± 0.1 | 87.5 ± 0.4 |
| Average           | 10.4       | 35.1       | 25.9       | 25.4       | 59.6       | 42.1       | 21.8       | 48.7       | 39.0       |

*Baseline* represents the original base model; “teacher” and “student” represent the teacher and student modules of CTDS, respectively. The one with a higher win rate of “student” and “baseline” is highlighted in bold.

![Fig. 5](image_url)

Median episode return of the base models (QMIX and QPLEX) and the corresponding ones with CTDS algorithm in Combat. (a) QMIX. (b) QPLEX.

We computed the win rate percentage criteria of different base models under different scenarios. Here, “Baseline” means the performance of the original base model, which achieves decentralized execution; “Teacher” means the performance of the teacher module, which relies on the perfect observation and thus violates decentralized execution; “Student” means the performance of the student module, which distills the knowledge from the teacher and allows decentralized execution. It can be seen that the teacher always achieves a higher win percentage than the student, which is consistent with our assumption that full observation brings benefits.

Under the settings of the mixing neural network of VDN, QMIX, and QPLEX, after 2 million steps of training, the performance gap between CTDS and the baseline exceeds the average win rate of 15.5%, 16.7%, and 17.2%, respectively. Fig. 6 demonstrates the learning curve of the base model, the teacher module, and the student module in different scenarios. The above results indicate that the CTDS framework can boost the performance of the original algorithms to a considerable margin in most scenarios. It can be observed that the performance of CTDS is closer to the baselines, especially in SMAC scenario *bane_vs_bane*. We attribute it to the fact that a large number of units and self-destruction mechanism of banelings, there exists significant uncertainty in this map. Therefore, our CTDS method performs more similarly to the base models in *bane_vs_bane*.

Third, we compare the CTDS framework with base models in the environment of GRF. Fig. 7 shows the mean score reward of the base model QMIX and the corresponding one with CTDS algorithm in two GRF scenarios: 1) *academy_3_vs_1_with_keeper* and 2) *academy_counterattack_easy*. CTDS could also achieve better performance than base models on GRF, which demonstrates the effectiveness of CTDS.

D. Sight Range Analysis

Under the setting of partial observability, agents can only receive the information within the restricted sight range. All the results reported in Section V-C are under the scenario that the sight range is 2. In this section, we further analyze the generalization of our method with regard to different sight ranges.

We adjust the sight range from 1 to 8 in the environment and investigate the performance of the base model, the teacher module and the student module under each setting, respectively. Due to the page limit, we only represent the results of methods taking QMIX as the base model on the 2c_vs_64zg task of *StarCraft II* after 1 million steps of training. As illustrated in Fig. 8, we can see that the performance of the original QMIX significantly increases with the increasing sight range, which indicates that more information about the other agents brings about a higher win percentage. Besides, the margin gain of the win percentage with one more sight range shows a decreasing trend, which is consistent with our common sense. Different from the base model that relies heavily on the sight range, the performance of our CTDS framework is relatively stable under different sight range that increases at a lower margin as the sight range enlarges. In other words, the performance gap between CTDS and the baseline increases with the decrease in sight range from 8 to 1. Our CTDS framework thereby has remarkable advantages in the case of very limited observations. Overall, the generalization of our framework with the varying sight range indicates the efficacy of the knowledge distillation mechanism. If there is no partial observability issue in the game setting, the input of the teacher and the student model is exactly the same. The training of the teacher model is equivalent to the CTDE training paradigm. The student model has no information gap compared to the teacher so it can completely learn what the teacher model grasps. Therefore, in this case, our CTDS training paradigm is equivalent to the original CTDE paradigm and will not bring any negative effects.
Fig. 6. Test win rates of the base models (i.e., VDN, QMIX, and QPLEX) and the corresponding ones with CTDS algorithm on five different maps of StarCraft II. “Baseline” represents the performance of the original base model; “Teacher” and “Student” represent the performance of teacher and student modules of CTDS, respectively. “Baseline” and “Student” adopt decentralized execution. The solid/dashed line shows the median win rate and the shadow area represents the max-min win rate in five different random seeds. (a) VDN-5m_vs_6m. (b) QMIX-5m_vs_6m. (c) QPLEX-5m_vs_6m. (d) VDN-bane_vs_bane. (e) QMIX-bane_vs_bane. (f) QPLEX-bane_vs_bane. (g) VDN-27m_vs_30m. (h) QMIX-27m_vs_30m. (i) QPLEX-27m_vs_30m. (j) VDN-MMM2. (k) QMIX-MMM2. (l) QPLEX-MMM2. (m) VDN-2c_vs_64zg. (n) QMIX-2c_vs_64zg. (o) QPLEX-2c_vs_64zg.

Fig. 7. Mean score reward of the base model QMIX and the corresponding one with CTDS algorithm in two GRF scenarios: 1) academy_3_vs_1_with_keeper and 2) academy_counterattack_easy.

E. Efficiency Analysis

Compared to the base model, our CTDS framework requires joint train the teacher module and the student module, which will increase the time cost. In this section, we further investigate how much extra time that CTDS introduces. All the efficiency experiments are conducted on a Ubuntu 18.04 sever with 4 Intel(R) Xeon(R) Gold 6252 CPU @ 2.10 GHz and GeForce RTX 2080Ti GPU.

Fig. 8. Performance of the test win rate percentage (including mean and standard deviation) of the different sight ranges. The experiment takes QMIX as the base model on the 2c_vs_64zg task after 1 million steps of training. “Baseline” represents the performance of the original base model, which achieves decentralized execution; “Teacher” and “Student” represent the performance of teacher and student modules of CTDS, respectively.
Table III illustrates the time cost of the three base models and the corresponding CTDS methods under different scenarios. We can see that applying CTDS over the base model brings about around 30% extra time costs on average. The reason is that compared to the teacher module, which has a similar network as the base model, the student module has a much simpler network structure with fewer trainable parameters.

### VI. CONCLUSION

In this article, we propose a novel cooperative MARL framework, named CTDS, which adopts the teacher module to allocate the team reward during centralized training and employs the student module to allow the agents to take actions only conditioned on local observations for decentralized execution. To realize decentralized execution, the student module aims to learn the individual Q-values only based on local observations by distilling the knowledge learned by the teacher module. In this way, CTDS makes full use of global observation to facilitate the learning process while achieving decentralized execution. Empirical results demonstrate that CTDS outperforms the recent state-of-the-art baselines both in terms of absolute performance and learning speed, which indicates the efficacy of our method and illustrates the benefits of utilizing centralized observations. Our immediate future work is to explore a more effective distillation way to accelerate model training. For example, in order to reduce training time, we can train the student module after obtaining a well-trained teacher module instead of jointly train teacher module and student module. As for long-term future work, we hope to continue the thinking of CTDS and improve the representations of the full range observations during training to achieve better performance and pay attention to improving efficiency as well.

### 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. Inform., vol. 9, no. 1, pp. 427–438, Feb. 2013.

[2] C. Zhang and V. Lesser, “Coordinated multi-agent reinforcement learning in networked distributed POMDPs,” in Proc. AAAI Conf. Artif. Intell., 2011, pp. 764–770.

[3] M. Hüttenrauch, A. Sošić, and G. Neumann, “Guided deep reinforcement learning for swarms systems,” 2017, arXiv:1709.06011.

[4] L. Busoniu, R. Babuska, and B. D. Schutter, “A comprehensive survey of multiagent reinforcement learning,” IEEE Trans. Syst., Man, Cybern., Part C: Appl. Rev., vol. 38, no. 2, pp. 156–172, Mar. 2008.

[5] S. Chen, M. Zhu, D. Ye, W. Zhang, Q. Fu, and W. Yang, “Which heroes to pick? Learning to draft in MOBA games with neural networks and tree search,” IEEE Trans. Games, vol. 10, no. 4, pp. 410–421, Dec. 2021.

[6] C. Hong, I. Jeong, L. F. Vecchiotti, D. Har, and J.-H. Kim, “AI world cup: Robot-soccer-based competitions,” Proc. IEEE Trans. Games, vol. 10, no. 4, pp. 330–341, Dec. 2021.

[7] P. Sunehag et al., “Value-decomposition networks for cooperative multi-agent learning,” 2017, arXiv:1706.05296.

[8] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, “A survey and critique of multiagent deep reinforcement learning,” Auton. Agents Multi-Agent Syst., vol. 33, no. 6, pp. 750–797, 2019.

[9] C. Openai et al., “Dota 2 with large scale deep reinforcement learning,” 2019, arXiv:1912.06680.

[10] D. Ye et al., “Mastering complex control in MOBA games with deep reinforcement learning,” in Proc. AAAI Conf. Artif. Intell., 2020, pp. 6672–6679.

[11] J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson, “Learning to communicate with deep multi-agent reinforcement learning,” in Proc. Ann. Conf. Neural Inf. Process. Syst., 2016, vol. 29, pp. 2137–2145.

[12] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, “QMX: Monotonic value function factorisation for deep multi-agent reinforcement learning,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 4295–4304.

[13] L. Kruemer and B. Banerjee, “Multi-agent reinforcement learning as a rehearsal for decentralized planning,” Neurocomputing, vol. 190, pp. 82–94, 2016.

[14] F. A. Oliehoek, M. T. Spaan, and N. Vlassis, “Optimal and approximate Q-value functions for decentralized POMDPs,” J. Artif. Intell. Res., vol. 32, pp. 289–333, 2008.

[15] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, “Multiagent actor-critic for mixed cooperative-competitive environments,” in Proc. Adv. Neural Inf. Process. Syst., 2017, vol. 30, pp. 6379–6390.

[16] J. N. 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.

[17] D. Ye et al., “Mastering complex control in MOBA games with deep reinforcement learning,” in Proc. AAAI Conf. Artif. Intell., 2020, pp. 6672–6679.

[18] W. Li, X. Wang, B. Jin, D. Luo, and H. Zha, “Structured cooperative multiagent deep reinforcement learning,” in Proc. Int. Conf. Learn. Representations, 2021.

[19] M. Samvelyan et al., “The StarCraft multi-agent challenge,” 2019, arXiv:2002.03939.

[20] J. Wang, Z. Ren, T. Liu, Y. Yu, and C. Zhang, “QPLEX: Duplex dueling multi-agent Q-learning,” in Proc. Int. Conf. Learn. Representations, 2021.

[21] A. OroojlooyJadid and D. Hajinezhad, “A review of cooperative multi-agent deep reinforcement learning,” Appl. Intell., pp. 1–46, 2022.

[22] W. Li, X. Wang, B. Jin, D. Liao, and H. Zha, “Structured cooperative reinforcement learning with time-varying composite action space,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 1, pp. 8618–8634, Nov. 2021.

[23] S. Sukhbaatar, A. Szlam, and R. Fergus, “Learning multiagent communication with backpropagation,” in Proc. Ann. Conf. Neural Inf. Process. Syst., 2016, pp. 2252–2260.

[24] P. Peng et al., “Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play StarCraft combat games,” 2017, arXiv:1703.10069.

[25] J. G. Kuba et al., “Trust region policy optimisation in multi-agent reinforcement learning,” 2021, arXiv:2109.11251.
[26] M. Wen et al., “Multi-agent reinforcement learning is a sequence modeling problem,” 2022, arXiv:2205.14953.

[27] M. Tan, “Multi-agent reinforcement learning: Independent versus cooperative agents,” in Proc. Int. Conf. Mach. Learn., 1993, pp. 336–337.

[28] C. S. de Witt et al., “Is independent learning all you need in the StarCraft multi-agent challenge?,” 2020, arXiv:2011.09533.

[29] C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. Bayen, and Y. Wu, “The surprising effectiveness of PPO in cooperative, multi-agent games,” 2021, arXiv:2103.01955.

[30] Y. Wang, B. Han, T. Wang, H. Dong, and C. Zhang, “DOP: Off-policy multi-agent decomposed policy gradients,” in Proc. Int. Conf. Learn. Representations, 2020.

[31] T. Rashid, G. Farquhar, B. Peng, and S. Whiteson, “Weighted QMIX: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning,” in Proc. Annu. Conf. Neural Inf. Process. Syst., 2020, vol. 33, pp. 10199–10210.

[32] A. Mahajan, T. Rashid, M. Samvelyan, and S. Whiteson, “MAVEN: Multi-agent variational exploration,” in Proc. Annu. Conf. Neural Inf. Process. Syst., 2019, vol. 32, pp. 7613–7624.

[33] N. C. Garcia, P. Morerio, and V. Murino, “Learning with privileged information via adversarial discriminative modality distillation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 42, no. 10, pp. 2581–2593, Oct. 2020.

[34] X. Wang, R. Zhang, Y. Sun, and J. Qi, “Adversarial distillation for learning with privileged provisions,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, no. 3, pp. 786–797, Mar. 2021.

[35] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” 2015, arXiv:1503.02531.

[36] Y. Kim and A. M. Rush, “Sequence-level knowledge distillation,” in Proc. Conf. Empirical Methods Natural Lang. Process., 2016, pp. 1317–1327.

[37] M. Phuong and C. Lampert, “Towards understanding knowledge distillation,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 5142–5151.

[38] J. Yin, D. Joo, J. Bae, and J. Kim, “A gift from knowledge distillation: Fast optimization, network minimization and transfer learning,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 4133–4141.

[39] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker, “Learning efficient object detection models with knowledge distillation,” in Proc. Annu. Conf. Neural Inf. Process. Syst., 2017, pp. 742–751.

[40] S. I. Mirzadeh, M. Farajtabar, A. Li, N. Levine, A. Matsukawa, and H. Ghasemzadeh, “Improved knowledge distillation via teacher assistant,” in Proc. AAAI Conf. Artif. Intell., 2020, pp. 5191–5198.

[41] E. Ilhan, J. Gow, and D. Perez, “Student-initiated action advising via advice novelty,” IEEE Trans. Games, vol. 14, no. 3, pp. 522–532, Sep. 2021.

[42] A. A. Rusu et al., “Policy distillation,” in Proc. Int. Conf. Learn. Representations, 2016.

[43] F. A. Oliehoek and C. Amato, A Concise Introduction to Decentralized POMDPs. Berlin, Germany: Springer, 2016.

[44] A. Koul, “MA-gym: Collection of multi-agent environments based on OpenAI gym,” 2019. [Online]. Available: https://github.com/koulanurag/ma-gym

[45] K. Kurach et al., “Google Research Football: A novel reinforcement learning environment,” in Proc. AAAI Conf. Artif. Intell., 2020, pp. 4501–4510.

Mingyu Yang received the B.E. degree in electronic engineering and information science in 2021 from the University of Science and Technology of China, Hefei, China, where he is currently working toward the M.S. degree in information and communication engineering with the Department of Electronic Engineering and Information Science. His research interests include reinforcement learning and multi-agent system.

Wengang Zhou (Senior Member, IEEE) received the B.E. degree in electronic information engineering from Wuhan University, Wuhan, China, in 2006, and the Ph.D. degree in electronic engineering and information science from the University of Science and Technology of China (USTC), Hefei, China, in 2011. From September 2011 to 2013, he was a Postdoc Researcher with Computer Science Department, The University of Texas at San Antonio. He is currently a Professor with EEIS Department, USTC. He has authored or coauthored more than 100 papers in ACM/IEEE transactions and CCF Tier-A International Conferences in his research interests, which include multimedia information retrieval, computer vision, and computer game.

Dr. Zhou is the recipient of the Best Paper Award for ICIMCS 2012. He was the Publication Chair of IEEE ICME 2021. He is the Associate Editor and the Lead Guest Editor of IEEE TRANSACTIONS ON MULTIMEDIA.

Jiangcheng Zhu received the B.Sc. degree in automation from Zhejiang University, Hangzhou, China, in 2012, the M.S. degree from the University of California at San Diego, San Diego, CA, USA, in 2014, and the Ph.D. degree in automation from Zhejiang University in 2019.

He is currently a Senior AI Scientist with Huawei Cloud. His current research interests include reinforcement learning and large-scale AI system.

Houqiang Li (Fellow, IEEE) received the B.S., M.Eng., and Ph.D. degrees in electronic engineering from the University of Science and Technology of China, Hefei, China, in 1992, 1997, and 2000, respectively.

He is currently a Professor with the Department of Electronic Engineering and Information Science, University of Science and Technology of China. He has authored and coauthored more than 200 papers in journals and conferences. His research interests include reinforcement learning, multimedia search, image/video analysis, video coding, and communication.

Dr. Li is the winner of National Science Funds (NSFC) for Distinguished Young Scientists, the Distinguished Professor of Changjiang Scholars Program of China, and the Leading Scientist of Ten Thousand Talent Program of China. He is the Associate Editor (AE) for IEEE TRANSACTIONS ON MULTIMEDIA and was the AE of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY. He was the General Co-Chair of ICME 2021 and the TPC Co-Chair of VCIP 2010. He was the recipient of National Technological Invention Award of China (second class) in 2019, National Natural Science Award of China (second class) in 2015, the Best Paper Award for VCIP 2012, the Best Paper Award for ICIMCS 2012, and the Best Paper Award for ACM MUM in 2011.