CTDS: Centralized Teacher with Decentralized Student for Multi-Agent Reinforcement Learning

Jian Zhao[1], Xunhan Hu[1], Mingyu Yang[1], Wengang Zhou[1], Jiangcheng Zhu[2], and Houqiang Li[1]

[1]University of Science and Technology of China
[2]Huawei Cloud

March 17, 2022

ABSTRACT

Due to the partial observability and communication constraints in many multi-agent 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 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.

1 Introduction

Cooperative multi-agent 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]. To train an 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 [5, 6].

A naive solution to MARL is to convert a cooperative multi-agent 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 [7, 8, 9], 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 [5, 10]. 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 multi-agent systems. Compared to decentralized training, training the agents in a centralized fashion allows the access to the global information and removes inter-agent communication constraints, which is beneficial to capture the coordination pattern and allocate the team reward.

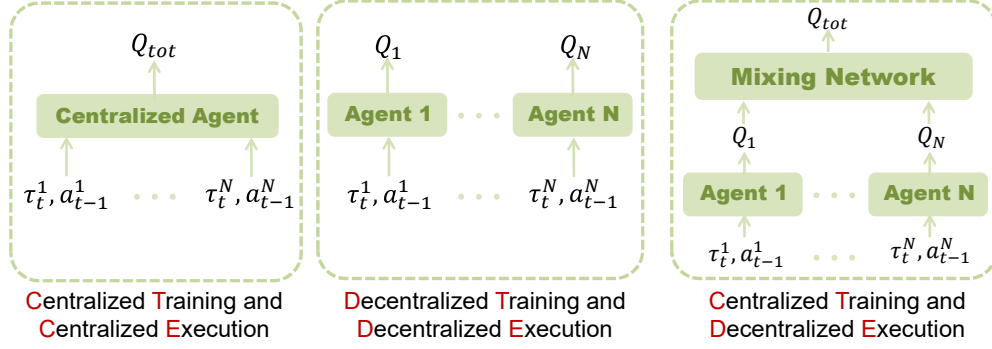[*]These authors contributed equally to this work

Figure 1: An overview of the Centralized Training and Centralized Execution (CTCE), Decentralized Training and Decentralized Execution (DTDE) and Centralized Training and Decentralized Execution (CTDE) framework.

Considering the necessity of decentralized policies and the benefits of centralized training in MARL, the paradigm of centralized training with decentralized execution (CTDE) [11, 12] has gained increasing attention in the last few years. In this paradigm, the policy of each agent is trained with global context in a centralized way and executed only based on local histories in a decentralized way. There are two non-trivial 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 [5, 13, 10, 14, 15, 16, 17] have been proposed recently, among which value based MARL algorithms [5, 10, 15, 16, 17] have shown state-of-the-art performance on challenging tasks, e.g., unit micromanagement in StarCraft II [18]. In existing value-based MARL algorithms, to adapt to the constraint of agents' partial observation during 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 the 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 reword following the Individual-Global-Max (IGM) principle [15]. Following this paradigm, there are many algorithms with different designs on the mixing network. For instance, Value DecompositionNetwork (VDN), the first attempt in this field, adopts the summation operation [5]. Differently, QMIX [10] assigns the nonnegative weights to individual Q-values with a non-linear function of the global state. Later, more and more algorithms employ the variants of attention mechanism as the aggregation function [15, 16, 17].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 a tacit coordination. This is due to the information gap between the training phase and 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 teach model and a student model. Specifically, the teach model estimate the individual Q-value estimation of agents based on the global state, while the student model regress each agent's Q-value with only local and partial observation. With global state, the teacher model is likely to learn a more tacit coordination among agents, which is transfered 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 upon 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.

2 Related Work

According to the relationship between agents, there are cooperative [19] and competitive [20] multi-agent systems. Due to its wide application in practice, cooperative multi-agent reinforcement learning has gained substantial research interests [21]. In a cooperative multi-agent system, a team of agents coordinate with each other and obtain an overall team reward. In view of the coordination among agents, early efforts focus on centralized learning and

execution [22, 23, 7, 24], in which agents can communicate with each other to access full observation of the environment. However, in many real-world scenarios, the partial observability and/or communication constraints among agents necessitate the decentralized execution.

Considering the necessity of decentralized execution, a paradigm of centralized training with decentralized execution (CTDE) has been widely adopted recently. To the best of our knowledge, value decomposition network (VDN) [5, 25] 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, thus decomposes a central action-value function into a sum of individual Q-values conditioned only on local observations. While achieving decentralization, VDN obtains a sub-optimal solution by simple summation integration and ignores the available global state information during training.

To distinguish the importance of different agents, QATTEN [16] utilizes a multi-head attention structure to weight 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 that take linear monotonic value functions, QMIX [10, 26] relaxes this assumption, and employs a mixing network by leveraging state information to decide the transformation weights, which allows non-linearity operations to decompose the central Q-value. Later on, QTRAN [15] releases the monotonicity restriction in QMIX[27] 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 [17] decomposes the central Q-value into the sum of individual value functions and a non-positive 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.

As far as we know, all the existing CTDE methods only utilize local observations instead of the global observation for learning individual Q-values during training in order to achieve decentralized execution. Intuitively, in a cooperative multi-agent system, it is beneficial when knowing more about other agents. Inspired by the benefit of the global observation, we adopt the concept of knowledge distillation [28, 29].

Knowledge distillation is firstly proposed for network compression [30], 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 aims to imitate the teacher under its guidance. In other words, knowledge distillation suggests training by matching the student's predictions to the teacher's predictions [31, 32]. Due to the advantage of fast optimization, network minimization and comparable performance [33, 32], the technique of knowledge distillation has been widely applied in various ways [31, 34, 35]. In [36], knowledge distillation is firstly leveraged to reinforcement learning to distill the policies to a dramatically smaller and 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 aims 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, a tacit coordination can be learned among the agents at the student side.

3 Preliminaries

3.1 Decentralized Partially Observable MDP

A cooperative multi-agent problem under the partial observability can be modeled as DEC-POMDP [12, 37], which is defined as a tuple $M =< \boldsymbol{N}, \boldsymbol{S}, \boldsymbol{A}, P, \boldsymbol{\Omega}, O, r, \gamma >$, where $\boldsymbol{N} \equiv \{g_1, g_2, \cdots, g_n\}$ is a finite set of agents and $\boldsymbol{S}$ describes a set of the global state s of the environment. At each time step, every agent $g_i \in \boldsymbol{N}$ receives an individual partial observation $o_i \in \boldsymbol{\Omega}$ according to the observation probability function $O_i(o_i|s)$. Each agent g_i has an action-observation history $\tau_i \in \boldsymbol{\tau} \equiv (\boldsymbol{\Omega} \times \boldsymbol{A})^*$ and its individual policy $\pi_i(a_i|\tau_i)$. According to π_i, each agent g_i chooses an action $a_i \in \boldsymbol{A}$, which forms a joint action $\boldsymbol{a} \equiv [a_i]_{i=1}^n$. After the action execution of all agents, it results in a team reward $r(s, \boldsymbol{a})$ and a transition to the next global state $s' \sim P(\cdot|s, \boldsymbol{a})$ by the environment. $\gamma \in [0, 1)$ is a discount factor. The formal objective is to find a joint policy $\boldsymbol{\pi} =< \pi_1, \cdots, \pi_n >$ that maximizes a joint state value function $V^{\pi}(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, \boldsymbol{\pi}]$. In addition, the joint action-value function $Q^{\boldsymbol{\pi}}(s, \boldsymbol{a}) = r(s, \boldsymbol{a}) + \gamma \mathbb{E}_{s'}[V^{\boldsymbol{\pi}}(s')]$ often appears in algorithm expressions to replace the joint state value function $V^{\pi}(s)$.
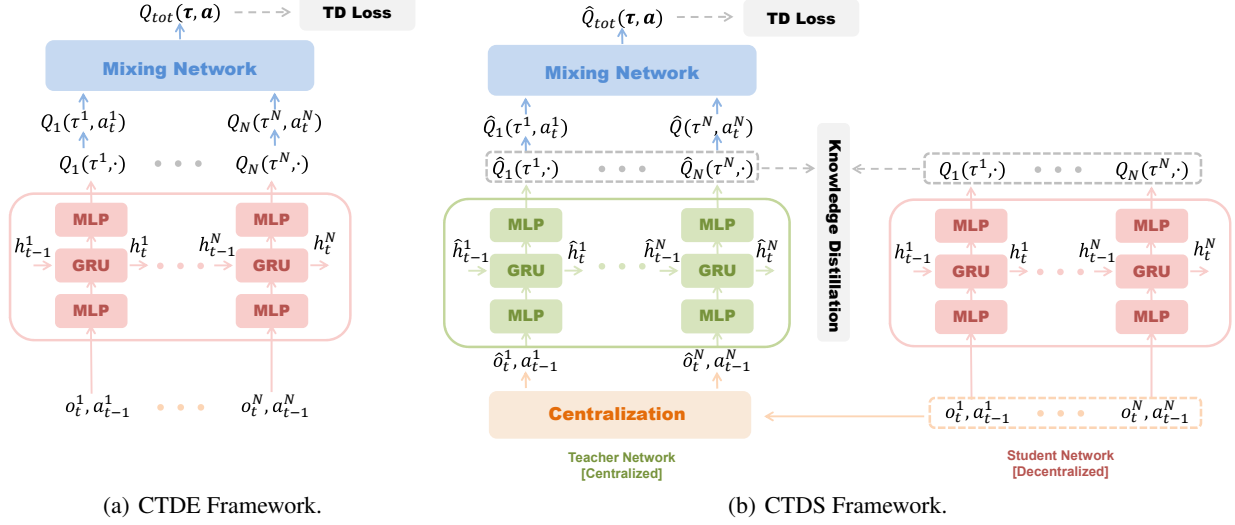
Figure 2: An overview of the CTDE framework and CTDS framework. CTDS consists of the teacher module (on the left) and the student module (on the right).

3.2 Deep Multi-Agent Q-Learning of CTDE Paradigm

CTDE is a popular paradigm of cooperative multi-agent reinforcement learning [5, 10, 15, 16, 17, 12]. 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. Multi-agent Q-learning algorithms represent the individual action-value function of the agent g_i with a neural network parameterized by θ_i, denoted as $Q_i(\tau_i, a_i|\theta_i)$. Define $\boldsymbol{\theta} = [\theta_i]_{i=1}^N$ 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.*,

$$Q_{tot} = f(Q_1, Q_2, \cdots, Q_n, s|\phi), \tag{1}$$

where ϕ 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 Individual-Global-Max (IGM), such that the following holds:

$$\arg\max_{\boldsymbol{a}} Q_{tot}(\boldsymbol{\tau}, \boldsymbol{a}) = \begin{bmatrix} \arg\max_{a_1} Q_1(\tau_1, a_1) \\ \vdots \\ \arg\max_{a_n} Q_n(\tau_n, a_n) \end{bmatrix}. \tag{2}$$

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

$$L(\boldsymbol{\theta}, \phi) = \mathbb{E}_{(\boldsymbol{\tau}, \boldsymbol{a}, r, \boldsymbol{\tau}') \in D}[(r + \gamma V(\boldsymbol{\tau}'|\boldsymbol{\theta}^-, \phi^-) - Q_{tot}(\boldsymbol{\tau}, \boldsymbol{a}|\boldsymbol{\theta}, \phi))^2], \tag{3}$$

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

4 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 about why our method works.

4.1 The CTDS Framework

In this subsection, we present a new framework called Centralized Teacher with Decentralized Student (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 multi-agent 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 decentralize execution, 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 mixing network to integrate the individual Q-values into the overall Q-value function. Obviously, if an agent could have 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 Figure 2. The teacher module is dedicated to allocating the team reward through centralized training, while the student module utilizes the partial observations to approximate the individual Q-value estimated by the teacher model. In the following, we will give a detailed description about the design of the teacher module and student module, and the knowledge distillation mechanism between them, respectively.

4.1.1 Teacher Module

The teacher module consists of two parts: one is the Q-network for estimating individual action-value function 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 that each agent has an infinite sight range to receive the observation of all the agents. For each agent g_i, the Q-network takes the centralized observation $\hat{o}_i^t$, last action a_i^{t-1} and last hidden state $\hat{h}_i^{t-1}$ as inputs, fits the perfect individual action-value $\hat{Q}_i(\hat{\tau}_i, \cdot)$ through the multi-layer 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 $\hat{Q}_{tot}$ by utilizing the global state information. The parameters of the teacher module are updated by the TD loss defined in equation 3 iteratively.

4.1.2 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 $\hat{o}_i^t$ in the teacher 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.

4.1.3 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 $\hat{Q}_i$ 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 [36]. Formally, the MSE loss is calculated as follows:

$$L_{\text{MSE}} = \Sigma_{i=1}^n \Sigma_{a \in A} (\hat{Q}_i(\hat{\tau}_i, a) - Q_i(\tau_i, a))^2. \tag{4}$$

4.1.4 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 ϵ-greedy from the $\hat{Q}_i$, which means choosing random actions with probability ϵ, otherwise choosing actions with the maximal $\hat{Q}_i$. 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 detailed procedure of our framework below.

4.2 Theoretical Analysis

In this subsection, 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 τ_i (accessible to the student module) and the complement information $\tau_i^* = \hat{\tau}_i^t \setminus \tau_i^t$ (we assume that τ_i and τ_i^* are independent). 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:

$$
\begin{aligned}
\text{(Teacher network)} \quad & f_T : (\tau_i, \tau_i^*) \to \hat{Q}_i, \\
\text{(Student network)} \quad & f_S : \tau_i \to Q_i.
\end{aligned}
\tag{5}
$$

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 τ_i^*, the execution module of CTDE tends to make non-optimal actions for the current state. The centralized trajectory releases the restriction above, however, it's usually unavailable during execution.

CTDS takes both sides into consideration. For a certain decentralized trajectory τ_i, the teacher module can give a centralized Q-value $f_T(\tau_i, \tau_i^*)$ with any possible implicit knowledge τ_i^*. If the expectation of 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 student module. The comprehensiveness and stability provided by the instruction from the teacher module is the principal cause of the student module's performance promotion.

Theorem 1. For any decentralized trajectory τ_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 student network is

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

where θ_S is the parameters of the student module. It can be proven that:

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

Therefore,

$$
\underset{\theta_S}{\arg\min} \sum_i \mathbb{E}_{\tau_i, \tau_i^*} [f_S(\tau_i) - f_T(\tau_i, \tau_i^*)]^2 = \underset{\theta_S}{\arg\min} \sum_i \mathbb{E}_{\tau_i} [f_S(\tau_i) - \mathbb{E}_{\tau_i^*} f_T(\tau_i, \tau_i^*)]^2.
\tag{8}
$$

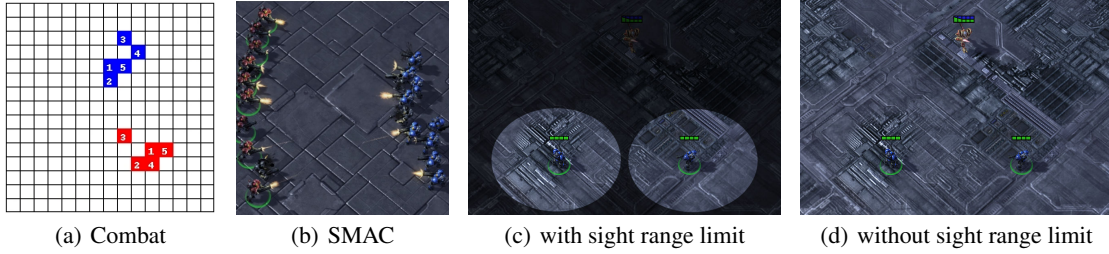| (a) Combat | (b) SMAC | (c) with sight range limit | (d) without sight range limit |

Figure 3: (a)&(b) Visualisation of the two experimental environments. (c)&(d) An illustration about the agents with partial and global observation in SMAC.

Scenario	VDN[5]			QMIX[10]			QPLEX[17]		
	Baseline	Teacher	Student	Baseline	Teacher	Student	Baseline	Teacher	Student
5m_vs_6m	6.9 ± 0.2	73.8 ± 0.9	$\mathbf{55.0 \pm 0.5}$	21.2 ± 0.4	66.9 ± 0.5	$\mathbf{48.1 \pm 1.0}$	23.8 ± 1.4	62.5 ± 0.9	$\mathbf{56.2 \pm 3.7}$
9m_vs_11m	0.6 ± 0.0	9.4 ± 0.6	$\mathbf{12.5 \pm 1.1}$	6.9 ± 0.1	9.4 ± 0.2	$\mathbf{11.2 \pm 0.1}$	$\mathbf{3.1 \pm 0.1}$	0.6 ± 0.0	0.6 ± 0.0
27m_vs_30m	0.0 ± 0.0	6.2 ± 0.1	$\mathbf{1.9 \pm 0.0}$	8.8 ± 1.0	58.8 ± 2.4	$\mathbf{42.5 \pm 1.5}$	0.0 ± 0.0	60.9 ± 5.8	$\mathbf{49.2 \pm 8.9}$
MMM2	0.0 ± 0.0	15.6 ± 1.2	$\mathbf{2.5 \pm 0.1}$	20.6 ± 1.2	80.6 ± 1.7	$\mathbf{28.1 \pm 1.7}$	0.0 ± 0.0	26.6 ± 0.8	$\mathbf{1.6 \pm 0.1}$
2c_vs_64zg	44.4 ± 4.7	70.6 ± 3.6	$\mathbf{57.5 \pm 2.6}$	69.4 ± 2.6	82.5 ± 0.4	$\mathbf{80.6 \pm 0.7}$	81.9 ± 0.6	93.1 ± 0.1	$\mathbf{87.5 \pm 0.4}$
Average	10.4	35.1	**25.9**	25.4	59.6	**42.1**	21.8	48.7	**39.0**

Table 1: The performance of the test win rate percentage (including mean and standard deviation) of different base models under different scenarios of StarCraft II. "Baseline" represents the original base model; "Teacher" and "Student" represent the teacher and student module of CTDS, respectively. The one with higher win rate of "Student" and "Baseline" is highlighted with bold.

5 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 in https://github.com/cathyhxh/CTDS.

5.1 Settings

In this paper, we test our method on two environments: Combat (a grid world environment introduced by [38]) and StarCraft II micromanagement (SMAC) [18].

In the experiment implementation, the different 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 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 the distance of 2. For the teacher module, perfect observation $\hat{o}_i$ is not limited by the sight range, that is to say, each agent can access the full observation of all the agents. Figure3 shows the examples of the agent with partial and global observations in the SMAC, respectively. In SMAC, we set partial sight range as 2 and perfect sight range as infinity. In Combat, partial sight range is 2 and perfect sight range is 4.

The algorithms we use are based on the Pymarl algorithm library [18]. 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 hyper-parameters are kept the same as default hyper-parameters in Pymarl. Following the base models [5, 10, 17], 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 a 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 500k steps in Combat.
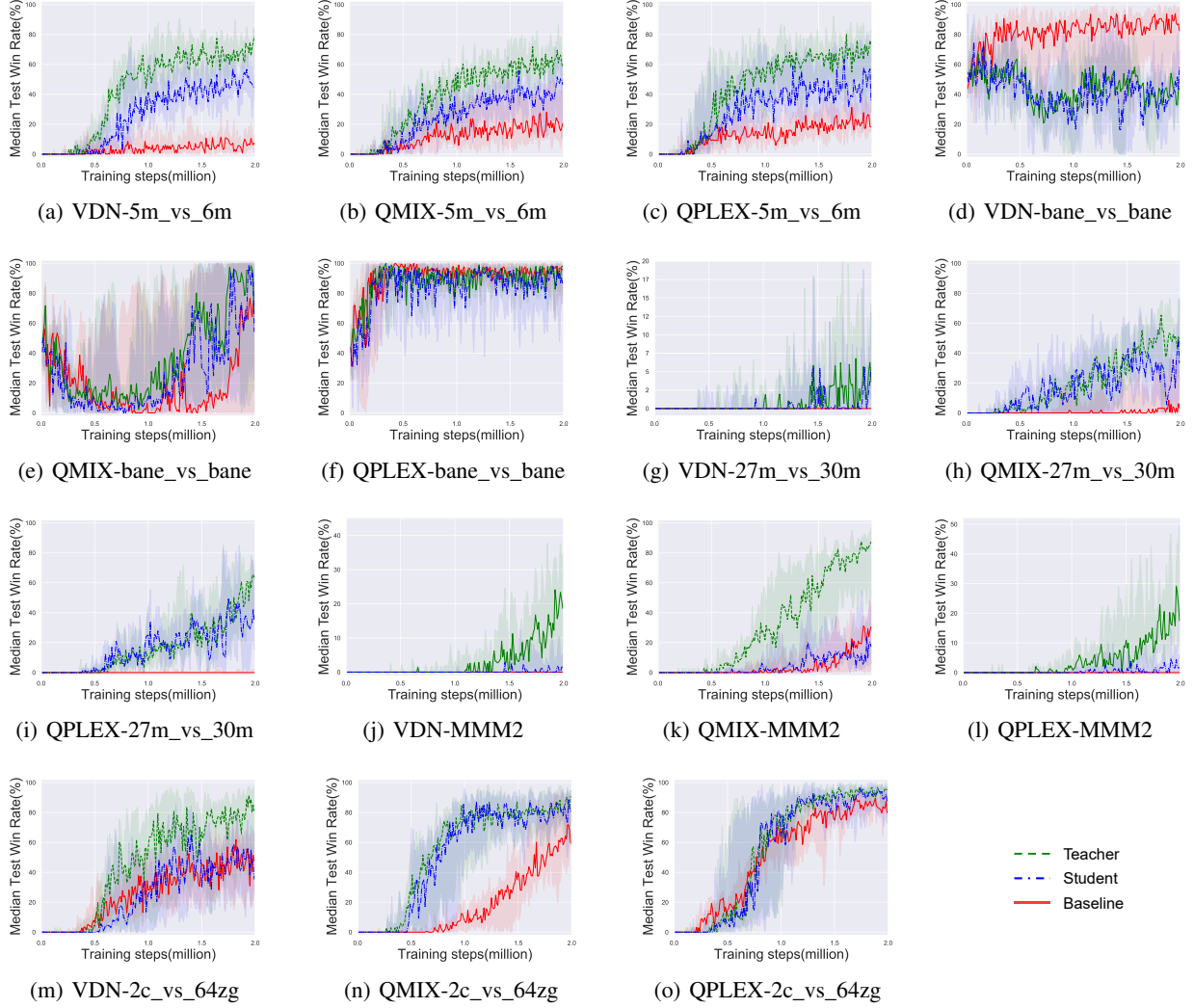
Figure 4: Test win rates of the base models (*i.e.*, VDN, QMIX and QPLEX) and the corresponding ones with CTDS algorithm on five different maps of StarCraft II. "Baseline" represents the performance of the original base model; "Teacher" and "Student" represent the performance of teacher and student module of CTDS, respectively. "Baseline" and "Student" achieve decentralized execution. The solid/dashed line shows the median win rate and the shadow area represents the max-min win rate in 5 different random seeds.

5.2 Performance Evaluation and Discussion

Our CTDS framework can be applied upon 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 [5], QMIX [10] and QPLEX [17].

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

The following are the experiment results in StarCraft II. Table 1 shows the final performance in terms of the test 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 thus violates decentralized execution; "Student" means the performance of the student module, which distills the knowledge from the teacher and allows decentralized execution.
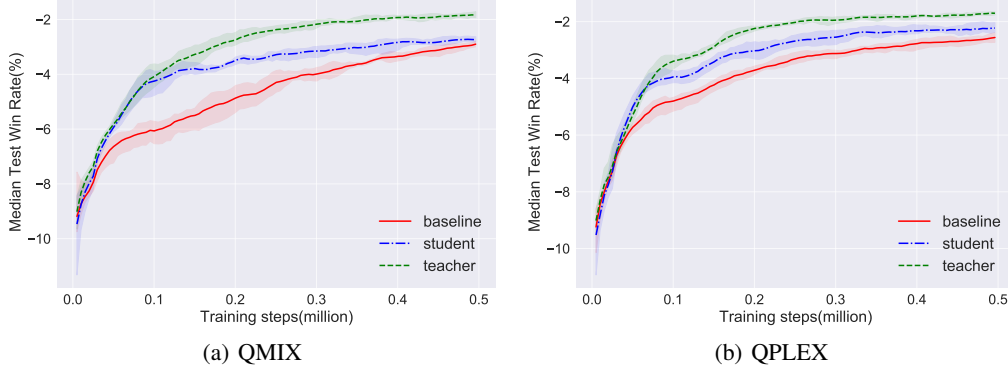
(a) QMIX

(b) QPLEX

Figure 5: Test win rates of the base models (QMIX and QPLEX) and the corresponding ones with CTDS algorithm in Combat. "Baseline" represents the performance of the original base model; "Teacher" and "Student"represent the performance of teacher and student module of CTDS, respectively. "Baseline" and "Student" achieve decentralized execution. The solid/dashed line shows the median win rate and the shadow area represents the max-min win rate in 5 different random seeds.

It can be seen that the teacher always achieves a higher win percentage than the student, which is consistent with our assumption that the 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. Figure 4 demonstrates the learning curve of the base model, the teacher module and the student module on different scenarios. The above results indicate that CTDS framework can boost the performance of the original algorithms to a considerable margin in most scenarios.

5.3 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 5.2 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 Figure 6, 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 which 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 of sight range from 8 to 1. Our CTDS framework thereby has remarkable advantages in the case of the very limited observations. Overall, the generalization of our framework with the varying of sight range
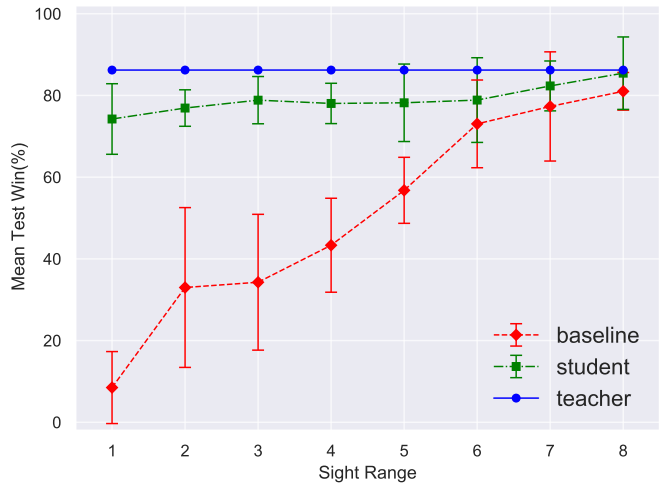

Figure 6: The performance of the test win rate percentage (including mean and standard deviation) of the different sight range. The experiment takes QMIX as the base model on the 2c_vs_64zg task after 1 million steps of training. "Baseline" represents the performance of the original base model which achieves decentralized execution; "Teacher" and "Student" represent the performance of teacher and student module of CTDS, respectively.

indicates the efficacy of the knowledge distillation
mechanism.

6 Conclusion

In this paper, we propose a novel cooperative multi-agent reinforcement learning 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 based on only local observations by distilling the knowledge learned by the teacher module. In this way, CTDS makes full use of the 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. In the future, we will further improve the representations of the full range observations during training to achieve better performance and pay attention to improving efficiency as well.

References

[1] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial Informatics*, 9(1):427–438, 2012.

[2] Chongjie Zhang and Victor Lesser. Coordinated multi-agent reinforcement learning in networked distributed pomdps. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 25, 2011.

[3] Maximilian Hüttenrauch, Adrian Šošić, and Gerhard Neumann. Guided deep reinforcement learning for swarm systems. *arXiv preprint arXiv:1709.06011*, 2017.

[4] L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

[5] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

[6] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E. Taylor. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems (AAMAS)*, 33(6):750–797, 2019.

[7] Openai, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, et al. Dota 2 with large scale deep reinforcement learning. *arXiv: Learning*, 2019.

[8] Deheng Ye, Zhao Liu, Mingfei Sun, Bei Shi, Peilin Zhao, Hao Wu, Hongsheng Yu, Shaojie Yang, Xipeng Wu, Qingwei Guo, et al. Mastering complex control in moba games with deep reinforcement learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[9] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multiagent reinforcement learning. In *Annual Conference in Neural Information Processing Systems (NeurIPS)*, volume 29, pages 2137–2145, 2016.

[10] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 4295–4304, 2018.

[11] Landon Kraemer and Bikramjit Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190:82–94, 2016.

[12] Frans A Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis. Optimal and approximate q-value functions for decentralized pomdps. *Journal of Artificial Intelligence Research*, 32:289–353, 2008.

[13] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pages 6379–6390, 2017.

[14] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 2974–2982, 2018.

[15] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 5887–5896, 2019.

[16] Yaodong Yang, Jianye Hao, Ben Liao, Kun Shao, Guangyong Chen, Wulong Liu, and Hongyao Tang. Qatten: A general framework for cooperative multiagent reinforcement learning. *arXiv preprint arXiv:2002.03939*, 2020.

[17] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex dueling multi-agent q-learning. 2021.

[18] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft Multi-Agent Challenge. *CoRR*, abs/1902.04043, 2019.

[19] Afshin OroojlooyJadid and Davood Hajinezhad. A review of cooperative multi-agent deep reinforcement learning. *arXiv preprint arXiv:1908.03963*, 2019.

[20] PJ t Hoen and Johannes A La Poutré. A decommitment strategy in a competitive multi-agent transportation setting. In *International Workshop on Agent-Mediated Electronic Commerce*, pages 56–72. Springer, 2003.

[21] Wenhao Li, Xiangfeng Wang, Bo Jin, Dijun Luo, and Hongyuan Zha. Structured cooperative reinforcement learning with time-varying composite action space. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, (1):1–1, 2021.

[22] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, page 2252–2260, 2016.

[23] Peng Peng, Ying Wen, Yaodong Yang, Quan Yuan, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069*, 2017.

[24] Deheng Ye, Guibin Chen, Wen Zhang, Sheng Chen, Bo Yuan, Bo Liu, Jia Chen, Zhao Liu, Fuhao Qiu, Hongsheng Yu, et al. Towards playing full moba games with deep reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:621–632, 2020.

[25] Jianhao Wang, Zhizhou Ren, Beining Han, Jianing Ye, and Chongjie Zhang. Towards understanding linear value decomposition in cooperative multi-agent q-learning. In *arxiv:cs.LG*, 2021.

[26] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, volume 33, pages 10199–10210, 2020.

[27] Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. Maven: Multi-agent variational exploration. In *Annual Conference in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[28] Nuno C. Garcia, Pietro Morerio, and Vittorio Murino. Learning with privileged information via adversarial discriminative modality distillation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 42(10):2581–2593, 2020.

[29] Xiaojie Wang, Rui Zhang, Yu Sun, and Jianzhong Qi. Adversarial distillation for learning with privileged provisions. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 43(3):786–797, 2021.

[30] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[31] Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1317–1327, 2016.

[32] Mary Phuong and Christoph Lampert. Towards understanding knowledge distillation. In *International Conference on Machine Learning (ICML)*, pages 5142–5151, 2019.

[33] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4133–4141, 2017.

[34] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. Learning efficient object detection models with knowledge distillation. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 742–751, 2017.

[35] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 34, pages 5191–5198, 2020.

[36] Andrei A Rusu, Sergio Gomez Colmenarejo, Çaglar Gülçehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. In *International Conference on Learning Representations (ICLR)*, 2016.

[37] Frans A Oliehoek and Christopher Amato. *A concise introduction to decentralized POMDPs*. Springer, 2016.

[38] Anurag Koul. ma-gym: Collection of multi-agent environments based on openai gym. `https://github.com/koulanurag/ma-gym`, 2019.