Depthwise Convolution for Multi-Agent Communication With Enhanced Mean-Field Approximation

Donghan Xie, Zhi Wang, *Member, IEEE*, Chunlin Chen, *Senior Member, IEEE*, and Daoyi Dong, *Fellow, IEEE*

***Abstract*— Multi-Agent settings remain a fundamental challenge in the reinforcement learning (RL) domain due to the partial observability and the lack of accurate real-time interactions across agents. In this article, we propose a new method based on local communication learning to tackle the multi-agent RL (MARL) challenge within a large number of agents coexisting. First, we design a new communication protocol that exploits the ability of depthwise convolution to efficiently extract local relations and learn local communication between neighboring agents. To facilitate multi-agent coordination, we explicitly learn the effect of joint actions by taking the policies of neighboring agents as inputs. Second, we introduce the mean-field approximation into our method to reduce the scale of agent interactions. To more effectively coordinate behaviors of neighboring agents, we enhance the mean-field approximation by a supervised policy rectification network (PRN) for rectifying real-time agent interactions and by a learnable compensation term for correcting the approximation bias. The proposed method enables efficient coordination as well as outperforms several baseline approaches on the adaptive traffic signal control (ATSC) task and the StarCraft II multi-agent challenge (SMAC).**

***Index Terms*— Agent communication, depthwise convolution, mean-field approximation, multi-agent reinforcement learning (MARL), StarCraft II multi-agent challenge (SMAC).**

I. Introduction

BASED on the Markov decision process (MDP) formulation, reinforcement learning (RL) [1], [2] allows agents to solve tasks from direct interactions with the environment, forming an optimal policy to make sequential decisions in a trial-and-error manner [3], [4], [5], [6]. The recent combination of RL with deep learning, referred to as deep RL (DRL) [7], has emerged as a promising direction for the autonomous acquisition of complex behaviors [8], [9], since it can acquire elaborate skills using general-purpose neural network representations from high-dimensional sensory inputs [10], [11], [12], [13]. Many artificial intelligence (AI) applications require the collaboration of multiple agents [14], [15], [16], and successfully scaling RL to multi-agent settings is crucial to building intelligent systems that can productively interact with each other and humans.

Multi-agent RL (MARL) is concerned with coordinating a set of agents toward maximizing each agent's or the group's objective, where each individual can only observe a local part of the shared environment [17]. A fundamental challenge in MARL is to tackle the nonstationarity due to the partial observability and the lack of accurate real-time interactions across agents [18], [19]. Fully centralized control that unifies all agents into a single one is usually infeasible due to the exponential growth of the size of joint action spaces. The simplest option to overcome the curse of dimensionality is to learn an individual action-value function independently for each agent, as in independent Q-learning (IQL) [20], while the learning is often unstable as changes in one agent's policy will affect those of the others. This issue can be mitigated by the centralized training and decentralized execution (CTDE) paradigm [21], [22] that typically leverages centralized critics to approximate the global value function of the joint policy and trains actors restricted to the local observation of a single agent [23], [24], [25], [26], [27], [28]. However, the execution phase can still suffer from non-stationarity due to not accounting for extra information from other agents.

Due to the partial observability and limited channel capacity, a communication protocol is vital to coordinate the behavior of agents and solve the task via information sharing [29], [30], [31]. A straightforward approach is to learn global communication that shares information across all agents, such as differentiable inter-agent learning (DIAL) [32], BiCNet [33], and informative multi-agent communication (IMAC) [34]. The computational cost can be high for all agents communicating with each other, especially with a large number of agents coexisting. Instead, the other kind of approaches attempts to learn informative local communication that needs to efficiently exploit the available communication resources, such

Manuscript received 31 October 2021; revised 20 July 2022 and 14 November 2022; accepted 15 December 2022. Date of publication 29 December 2022; date of current version 4 June 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62006111 and Grant 62073160, in part by the Australian Research Council's Future Fellowship Funding Scheme under Project FT220100656, in part by the Natural Science Foundation of Jiangsu Province of China under Grant BK20200330, and in part by the Alexander von Humboldt Foundation, Germany. *(Corresponding author: Zhi Wang.)*

Donghan Xie is with the Department of Control Science and Intelligence Engineering, School of Management and Engineering, Nanjing University, Nanjing 210093, China, and also with Tencent, Shenzhen 518000, China (e-mail: donghanxie@smail.nju.edu.cn).

Zhi Wang and Chunlin Chen are with the Department of Control Science and Intelligence Engineering, School of Management and Engineering, Nanjing University, Nanjing 210093, China (e-mail: zhiwang@nju.edu.cn; clchen@nju.edu.cn).

Daoyi Dong is with the School of Engineering and Information Technology, University of New South Wales, Canberra, ACT 2600, Australia (e-mail: daoyidong@gmail.com).

Digital Object Identifier 10.1109/TNNLS.2022.3230701

as attentional communication (ATOC) [35], IC3Net [36], and I2C [18]. While these methods exploit extra information from neighboring agents, they do not explicitly learn the effect of joint actions. It could potentially reduce the coordination efficiency since the environment dynamics of an agent depends on actions of the others.

In this article, we propose a new MARL method within a large number of agents coexisting based on local communication learning. First, inspired by the fact that convolution is widely used in extracting local relations [37], we design a new communication protocol that exploits the ability of depthwise convolution [38] to efficiently learn local communication between neighboring agents. Since the environment dynamics of an agent depends on actions of the others, we take the policies of neighboring agents as inputs and explicitly learn the effect of joint actions to facilitate multi-agent coordination. Second, in our method, the mean-field approximation [39] is introduced to approximate the interactions within the population of agents considering the average effect from the neighboring agents of an individual, thus considerably reducing the scale of agent interactions. To more accurately coordinate behaviors of neighboring agents, we enhance the mean-field approximation by a supervised policy rectification network (PRN) that predicts real-time policies from previous information to rectify real-time agent interactions.[1] Besides, since mean-field approximation drops out the second-order remainders, we attempt through learning to compensate for the approximation bias via our communication protocol and obtain a more accurate mean-field estimate of agent interactions.

We extensively evaluate our method on two multi-agent tasks: the adaptive traffic signal control (ATSC) on the simulation of urban mobility (SUMO) platform [19], [40] and the StarCraft II multi-agent challenge (SMAC) [41]. Experimental results show that our method can achieve more efficient multi-agent coordination and outperform several baseline approaches.

In summary, our contributions are threefold.

1) We propose a new protocol that exploits the ability of depthwise convolution to efficiently learn local communication, and we explicitly learn the effect of joint actions to facilitate multi-agent coordination.
2) We exploit the mean-field approximation to reduce the scale of agent interactions, and enhance the mean-field estimate by a supervised PRN and a learnable compensation term.
3) We perform extensive experiments to verify that our method can consistently improve the multi-agent learning performance over several baselines.

The remainder of this article is organized as follows. Section II gives the related work on MARL. Section III introduces preliminaries of MARL and mean-field approximation. Section IV first presents the proposed DCCP and enhanced mean-field approximation, followed by the final integrated algorithm. Experiments on the ATSC and SMAC tasks are conducted in Section V. Section VI presents concluding remarks.

[1]The communication of real-time information during execution may lead to a certain time lag due to bandwidth limitations in real-world applications [30]. Hence, throughout the article, we assume that an individual cannot know the real-time information of its neighbors during execution.

II. Related Work

Communication learning is recognized as a promising way to handle multi-agent systems by sharing extra information (e.g., observations and policies) with each other for coordination [42], [43], [44], [45]. A straightforward approach is global information sharing among all agents. Foerster et al. [32] proposed to learn one-round point-to-point communication, which uses the broadcast messages from the previous time step instead of real-time ones due to communication constraints in the real world. Peng et al. [33] used the bidirectional recurrent neural network (RNN) to maintain the communication protocol that can learn various types of coordination strategies. IMAC [34] learned an efficient protocol that compresses communication messages and schedules more accurate information delivery to overcome the bandwidth limitations. Some recent works, such as targeted multi-agent communication (TarMAC) [30], structured attentive reasoning network (SARNet) [46], and deep implicit coordination graph (DICG) [47], employed an attention mechanism with global communication to learn what messages to send and whom to address these messages to Wang et al. [48] designed an expressive and succinct communication protocol by introducing information-theoretic regularizers for maximizing mutual information between agents' action selection and communication messages. Jin et al. [49] utilized the event-triggered mechanism to reduce a large amount of consumption of continuous communication at the cost of a small amount of computing resources. The computational cost can be high for all agents transmitting a large amount of information to each other, especially with many agents coexisting.

Instead, the other kind of approaches attempts to learn local communication that needs efficiently exploiting the available communication resources. Sukhbaatar et al. [29] extended CommNet to a local variant that allows agents to communicate to others within a certain range only. Jiang and Lu [35] proposed an attentional model that dynamically determines whether the agent should communicate with other agents to cooperate in its observable field. Singh et al. [36] learned when to communicate by a gating mechanism with individualized rewards to gain better performance and scalability. Ding et al. [18] turned to realize peer-to-peer communication using causal inference to learn a prior network that maps the agent's local observation to a belief about whom to communicate with. While these local communication learning methods account for the extra information from other agents, they do not explicitly learn the effect of joint actions, which could potentially reduce coordination efficiency since the environment dynamics of an agent depends on the others.

Another thread of work is to consider the extra information by explicitly learning policies of other agents. Tesauro [50] proposed hyper Q-learning that estimates policies of other agents using Bayesian inference, which is only feasible for discrete or low-dimensional tasks due to the high-computational burden of computing exact Bayesian posteriors. Foerster et al. [51] extended hyper Q-learning to high-dimensional and

continuous state spaces by communicating only two scalars (i.e., the current learning step and the learning rate) among agents, while these two scalars have limited representation power for conjecturing the underlying policies. Yang et al. [39] introduced the mean-field theory [52] into MARL to reduce the scale of agent interactions, using interactions only between an individual and the average effect from its neighboring agents to approximate those interactions within a population of agents.

Here, we exploit the ability of depthwise convolution to efficiently extract local relations and learn local communication between neighboring agents. In contrast to existing local communication learning approaches, we take the policies of neighboring agents as inputs and explicitly learn the effect of joint actions to facilitate coordination efficiency. Further, we enhance the mean-field approximation with a supervised PRN and a learnable compensation term to obtain a more accurate mean-field estimate of agent interactions.

III. Preliminaries

A. Reinforcement Learning

RL is studied to deal with the sequential decision-making problems through the MDP framework. An MDP is concerned with the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the conditional transition probabilities, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. We use $\pi(a|s) : \mathcal{S} \times \mathcal{A} \to [0, 1]$ to denote a stochastic policy that is the probability distribution of executing action a at state s. The goal of RL is to find the optimal policy π^* to maximize its expected return $J(\pi)$ as

$$J(\pi) = \mathbb{E}_{s_0, a_0, \ldots,} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] \tag{1}$$

where $a_t \sim \pi(\cdot|s_t)$.

The action value function is defined as the return of policy π starting from executing action a in state s as

$$Q^{\pi}(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) | s_0 = s, a_0 = a, \pi \right]. \tag{2}$$

Then, the optimal policy can be directly derived as

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} Q^*(s, a) \tag{3}$$

where $Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a)$. Deep Q-learning [7] represents the Q-function $Q(s, a; \boldsymbol{\theta})$ with a neural network parameterized by $\boldsymbol{\theta}$. During training, the transition tuples (s, a, r, s') are stored in a *replay buffer*. The parameters $\boldsymbol{\theta}$ are updated by iteratively sampling a batch of transitions from the buffer and minimizing the squared temporal-difference error as

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_i \left[\left(y_i^{\text{target}} - Q(s, a; \boldsymbol{\theta}) \right)^2 \right] \tag{4}$$

where the target Q-values can be formulated as

$$y_i^{\text{target}} = \begin{cases} r_i, & \text{if terminated,} \\ r_i + \gamma \max\limits_{a_i'} Q\left(s_i', a_i'; \boldsymbol{\theta}^-\right), & \text{otherwise} \end{cases} \tag{5}$$

and $\boldsymbol{\theta}^-$ are parameters of a target network that is periodically frozen and synchronized from $\boldsymbol{\theta}$ for several iterations.

B. Multi-Agent RL (MARL)

MARL involves multiple interacting agents coexisting in a sharing environment, where each individual can only observe a local part of the shared environment. In a multi-agent system, the reward received by an agent depends not only on its own action, but also on the actions taken by the others. Through communicating with each other, agents obtain the observations and policies of others to coordinate their behaviors, and learn to maximize an individual or group objective. Following the state-of-the-art works [14], [26], [27], we formulate the MARL system by a partially observable Markov game [53], which is a multi-agent extension of Markov decision processes (MDPs). A partially observable Markov game for N agents is given by a tuple $(N, \mathcal{S}, \mathcal{A}_1, \ldots, \mathcal{A}_N, T, R_1, \ldots, R_N, O_1, \ldots, O_N, \gamma)$, where $\mathcal{S}$ is the state space of the environment, $\mathcal{A} = \mathcal{A}_1 \times, \ldots, \times \mathcal{A}_n$ is the action space, T is the transition function, $R_i : \mathcal{S} \times \mathcal{A}_i \times \mathcal{S} \mapsto \mathbb{R}$ is the reward function of agent i, $O_i : \mathcal{S} \times \Omega \mapsto [0, 1]$ is a private observation correlated with the state $o_i : \mathcal{S} \mapsto \mathcal{O}_i$, and $\gamma \in [0, 1)$ is the discount factor. The objective for each agent i is to maximize its own return $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$.

At each time step, each agent selects its own action $a_i \in \mathcal{A}_i$ conditioned on its own observation o_i. After executing the joint action $\boldsymbol{a} = \{a_1, \ldots, a_N\}$, each agent receives its own reward $R^i(\boldsymbol{s}, \boldsymbol{a})$. Each agent has its own value function $Q^i(\boldsymbol{s}, \boldsymbol{a})$ with respect to the global state $\boldsymbol{s}$ and joint action $\boldsymbol{a}$ as

$$Q^i(\boldsymbol{s}_t, \boldsymbol{a}_t) = \mathbb{E}_{\boldsymbol{s}_{t+1:\infty}, \boldsymbol{a}_{t+1:\infty}} \left[\sum_{\tau}^{\infty} \gamma^{\tau} r_{\tau}^i | \boldsymbol{s}_t, \boldsymbol{a}_t \right]. \tag{6}$$

This general formalization is a noncooperative setting, i.e., no explicit coalitions are considered. When the multi-agent task is fully cooperative, the reward function is shared among agents as $R^i = R^j (\forall i, j \in \{1, \ldots, N\})$. Therefore, the value function in cooperative settings can be formalized by a total term $Q^{\text{tot}}(\boldsymbol{s}, \boldsymbol{a})$ as in value function factorization approaches [25].

C. Mean-Field Approximation

Under the hypothesis that the global state is available, mean-field approximation proposes to use the interactions between a given agent and a virtual agent whose action is the mean value of the given agent's neighbors, to approximate the value function with respect to joint actions. In MARL, the standard Q-function $Q^i(s, \boldsymbol{a})$ is infeasible to learn since the dimension of joint action $\boldsymbol{a}$ exponentially grows with the number of agents. To reduce the complexity of the interactions among agents, mean-field approximation factorizes the Q-function with pairwise local interactions

$$Q^i(s, \boldsymbol{a}) = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} Q^j\left(s, a^j, a^j\right) \tag{7}$$

where $\mathcal{N}_i$ is the set of the neighbors of agent i with size $|\mathcal{N}_i|$.

The mean action $\bar{a}^i$ is based on the neighborhood $\mathcal{N}_i$, and the action of neighbor j could be expressed by a sum of the mean-action and a small fluctuation $\delta a^{i,j}$ as

$$a^j = \bar{a}^i + \delta a^{i,j}, \quad \text{where} \quad \bar{a}^i = \frac{1}{|\mathcal{N}_i|} \sum_j a^j. \tag{8}$$

Following (7) and (8), the Q-function can be expanded and expressed by Taylor's theorem as

$$\begin{aligned} Q^i(s,\boldsymbol{a}) &= \frac{1}{|\mathcal{N}_i|}\sum_j Q^i\left(s,a^i,a^j\right) \\ &= \frac{1}{|\mathcal{N}_i|}\sum_j\Big[Q^i\left(s,a^i,\bar{a}^i\right)+\nabla_{\bar{a}^{i,j}}Q^i\left(s,a^i,\bar{a}^i\right)\cdot\delta a^{i,j} \\ &\qquad +\frac{1}{2}\delta a^{i,j}\cdot\nabla^2_{\bar{a}^{i,j}}Q^i\left(s,a^i,\tilde{a}^{i,j}\right)\cdot\delta a^{i,j}\Big] \\ &= Q^i\left(s,a^i,\bar{a}^i\right)+\frac{1}{2|\mathcal{N}_i|}\sum_j R^i_{s,a^i}\left(a^j\right) \\ &\approx Q^i\left(s,a^i,\bar{a}^i\right) \end{aligned} \tag{9}$$

where the second-order Taylor polynomial remainder $R^i_{s,a^i}(a^j)$ denotes $\delta a^{i,j}\cdot\nabla^2_{\bar{a}^{i,j}}Q^i(s,a^i,\tilde{a}^{i,j})\cdot\delta a^{i,j}$ with $\tilde{a}^{i,j}=\bar{a}^i+\epsilon^{i,j}\delta a^{i,j}$ and $\epsilon^{i,j}\in[0,1]$. Suppose the Q-function $Q^i(s,a^i,a^j)$ is Ω-smooth, the remainder could be proved that it is bounded within the interval $[-2\,\Omega, 2\,\Omega]$, and is dropped as a fluctuation term.

IV. Our Method

In this section, we first propose a new protocol that efficiently learns local communication via depthwise convolution. Then, we present the enhanced mean-field approximation that uses PRN to rectify real-time agent interactions. Finally, we give the integrated algorithm.

A. Depthwise Convolution-Based Communication Protocol (DCCP)

Due to the partial observability and the lack of accurate real-time interactions across agents, a communication protocol is vital to coordinate the behavior of each individual and to solve the task via sharing the observations and policies. It is straightforward to broadcast essential messages across all agents, while the learning becomes intractable due to the exponential growth of agent interactions when the number of agents increases largely. Moreover, this approach suffers from content redundancy and is unsustainable under bandwidth limitations. Instead, we consider learning a local communication protocol between neighboring agents. Convolution is widely used in extracting local relations, and inspired by this, we introduce the spatial convolution into the local communication protocol. In principle, the convolution can perform a form of system identification, conjecturing parameters of neighboring agents and coordinating the individual's behavior as a function of these parameters.

In MARL, the information to be shared usually has distinguishable channel-wise semantics. For example, different channels of an observation vector may represent different attributions of the environment, or different channels in the output of a Q-network can express outcomes of different actions. Standard convolution cannot preserve the channel-wise semantics since it filters and combines inputs from different channels into a new set of outputs in one step. Therefore, we propose to use one-layer depthwise convolution [38] to filter inputs from the same channel and produce channel-wise outputs that are semantics-invariant with inputs. In each channel, we share the parameters of multiple convolution kernels for all agents, aiming to extract and transfer the channel-wise common knowledge (embedded in the convolution kernels) across agents. Then, each individual has its own agent-specific weights to calculate the weighted sum of outputs from these depthwise convolution kernels.

Fig. 1 illustrates the proposed communication protocol via depthwise convolution. In this article, we assume that the neighboring relationship between agents keeps fixed during learning. Let $\boldsymbol{x}^i=[x^i_1,x^i_2,\ldots,x^i_M]^T$ denote the M-channel input vector of agent i $(i=1,\ldots,N)$, and let $\mathcal{N}_i$ denote the set of the ith agent's neighbors. For a given channel m, we use K convolution kernels of size $n\times n$ to extract the local relations between agent i and its neighbors $\{j\}_{j\in\mathcal{N}_i}$, which outputs a K-dimensional hidden vector $\boldsymbol{u}^i_m$ as $\boldsymbol{u}^i_m=\mathrm{Conv}(x^i_m,\{x^j_m\}_{j\in\mathcal{N}_i})$. We use a different set of K convolution kernels for each channel, resulting in $M\cdot K$ convolution kernels in total. Then, agent i uses its own agent-specific weight vector $\boldsymbol{w}^i_m$ to calculate the weighted sum of outputs from the K convolution kernels as $z^i_m=\boldsymbol{u}^{i\,T}_m\cdot\boldsymbol{w}^i_m$. We gather the outputs of all M channels to obtain the output vector of the DCCP as $\boldsymbol{z}^i=\mathrm{DCCP}(\boldsymbol{x}^i,\{\boldsymbol{x}^j\}_{j\in\mathcal{N}_i})$.

Through the depthwise convolution channel by channel, the channel-wise semantics is well preserved and the output $\boldsymbol{z}^i$ of the communication protocol is semantics-invariant with the input $\boldsymbol{x}^i$. The convolution kernels are shared among all agents to modulate the common knowledge for behavior coordination, while each individual maintains its agent-specific weights to combine the convolution kernels for promoting diversity of individual behaviors.

B. Enhanced Mean-Field Approximation

In MARL, the learning of an agent's optimal policy depends on the dynamics of the others that are a part of the environment. In the article, we use the mean-field approximation [39] to model the behaviors of other agents for each individual. It approximately treats the interactions within agents as the interaction between an individual and a virtual agent averaged by other agents, which transmits messages across agents with a reduced scale of agent interactions.

In mean-field Q-learning (MF-Q), the ith agent's Q-function of the global joint action is first factorized using only pairwise interactions between neighboring agents as

$$Q^i(\boldsymbol{s},\boldsymbol{a})=\frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i}Q^i\left(\boldsymbol{s},a^i,a^j\right) \tag{10}$$

where $\boldsymbol{s}$ is the global state, $\boldsymbol{a}$ is the joint action of agents, and $|\cdot|$ denotes the cardinality of the set. By the Taylor's theorem, the Q-function can be expanded and approximated as

$$Q^i(\boldsymbol{s},\boldsymbol{a})\approx Q^i\left(\boldsymbol{s},a^i,\frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i}a^j\right). \tag{11}$$

Since an individual cannot know real-time policies of its neighboring agents, MF-Q uses previous actions of the neighbors

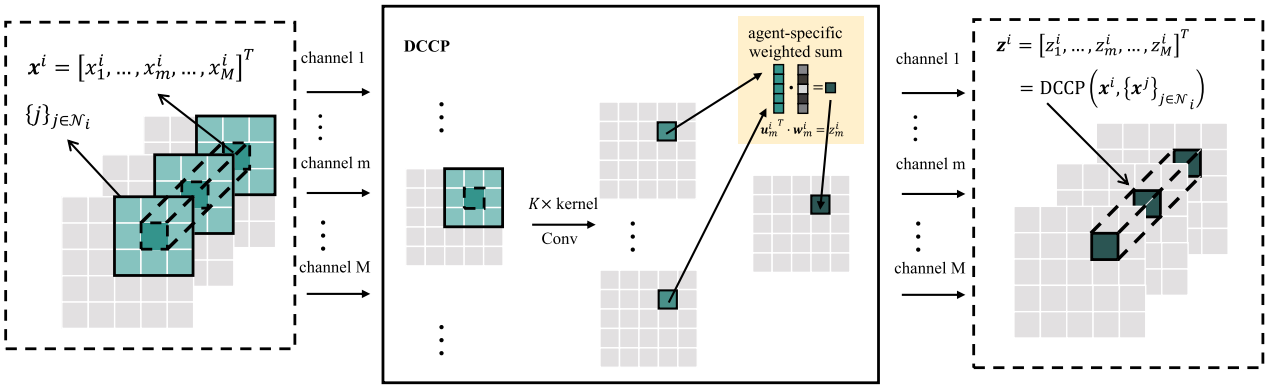


Fig. 1. Illustration of DCCP. We use a different set of K convolution kernels for each channel, and each kernel is shared for all agents. In channel m, the input of the ith agent, x_m^i, is transformed to a K-dimensional vector $\boldsymbol{u}_m^i$ by convolution over the local area surrounding this agent. Then, the agent-specific weight vector $\boldsymbol{w}_m^i$ is used to calculate the weighted sum of $\boldsymbol{u}_m^i$ to obtain the final output z_m^i.

to estimate the current action $\tilde{a}_t^i$ as

$$\tilde{a}_t^i = \mu_t^i\left(\boldsymbol{s}_t, \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i} a_{t-1}^j\right) \tag{12}$$

where μ^i is the policy function derived from Q^i. In place of the real action a_t^j, the estimated one $\tilde{a}_t^j$ is used to make the real-time decision a_t^i as

$$a_t^i = \mu_t^i\left(\boldsymbol{s}_t, \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i} \tilde{a}_t^j\right). \tag{13}$$

However, the estimated action $\tilde{a}_t$ has no guaranteed similarity with the real one a_t for two reasons: 1) the same policy function μ is used in both action estimation and real decision making and 2) the estimated action $\tilde{a}_t$ is predicted from previous real action a_{t-1} that may have no explicit correlation with the current real action a_t. Moreover, MF-Q assumes that each agent has access to the global state of the system, which might be problematic in multi-agent settings where each agent can only observe a local part of the shared environment.

To more accurately coordinate behaviors of neighboring agents, we enhance the mean-field approximation by a supervised PRN that predicts real-time actions from previous information to rectify real-time agent interactions. Since the policy is derived from the Q-values, we estimate the agent's real-time Q-values $\hat{\boldsymbol{q}}_t^i$ from previous partial observations $\boldsymbol{o}_{t-1}$ and Q-values $\boldsymbol{q}_{t-1}$ of itself and its neighboring agents as

$$\hat{\boldsymbol{q}}_t^i = f_{\boldsymbol{\phi}}^{\mathrm{PRN}}\left((\boldsymbol{o}_{t-1}^i, \boldsymbol{q}_{t-1}^i), \left\{\left(\boldsymbol{o}_{t-1}^j, \boldsymbol{q}_{t-1}^j\right)\right\}_{j\in\mathcal{N}_i}\right) \tag{14}$$

where $f_{\boldsymbol{\phi}}^{\mathrm{PRN}}$ is the PRN function parameterized by weights $\boldsymbol{\phi}$. It is trained in a supervised regression manner to make the estimated Q-values $\hat{\boldsymbol{q}}_t^i$ more accurate about the ground truth $\boldsymbol{q}_t^i$, and the loss function is formalized as

$$\mathcal{L}_{\boldsymbol{\phi}}^{\mathrm{PRN}} = \sum_{i\in N}\left\|\boldsymbol{q}_t^i - \hat{\boldsymbol{q}}_t^i\right\|^2. \tag{15}$$

Fig. 2 illustrates the network structure of PRN in detail. First, we use an encoder that is shared across agents to embed

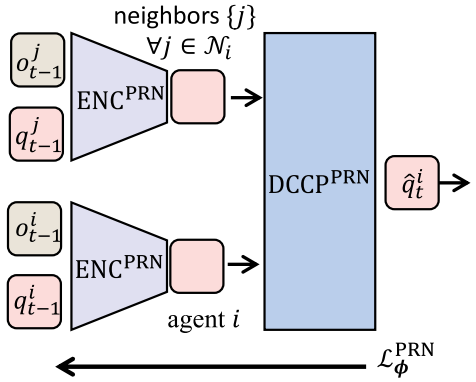


Fig. 2. Illustration of the PRN.

the input $(\boldsymbol{o}_{t-1}^i, \boldsymbol{q}_{t-1}^i)$ into a latent vector $\boldsymbol{v}^i$ with the same dimension as the action space as $\boldsymbol{v}^i = \mathrm{ENC}^{\mathrm{PRN}}(\boldsymbol{o}_{t-1}^i, \boldsymbol{q}_{t-1}^i)$. Then, we feed the latent vectors of agent i and its neighbors into our DCCP for information sharing and behavior coordination, and obtain the predicted real-time Q-values $\hat{\boldsymbol{q}}_t^i$ as $\hat{\boldsymbol{q}}_t^i = \mathrm{DCCP}^{\mathrm{PRN}}(\boldsymbol{v}^i, \{\boldsymbol{v}^j\}_{j\in\mathcal{N}_i})$. In short, the PRN function can be described as

$$\hat{\boldsymbol{q}}_t^i = \mathrm{DCCP}^{\mathrm{PRN}}\left(\mathrm{ENC}^{\mathrm{PRN}}\left((\boldsymbol{o}_{t-1}^i, \boldsymbol{q}_{t-1}^i), \left\{\left(\boldsymbol{o}_{t-1}^j, \boldsymbol{q}_{t-1}^j\right)\right\}_{j\in\mathcal{N}_i}\right)\right). \tag{16}$$

C. Integrated Algorithm

With the above, Fig. 3 depicts the architecture of our method that consists of three parts: the observation prediction network (OPN) with DCCP, the PRN with enhanced mean-field approximation, and the value function network (VFN).

The first module, OPN, employs DCCP to share observations across neighboring agents, which addresses the partial observability issue with low-computational cost. In real-world applications, the communication of real-time information may lead to a certain time lag due to bandwidth limitations [30]. Analogous to PRN, we predict real-time observations from previous information to more accurately coordinate behaviors of neighboring agents. In detail, we use the previous observations $\boldsymbol{o}_{t-1}$ and the previous Q-values $\boldsymbol{q}_{t-1}$ of an agent and its

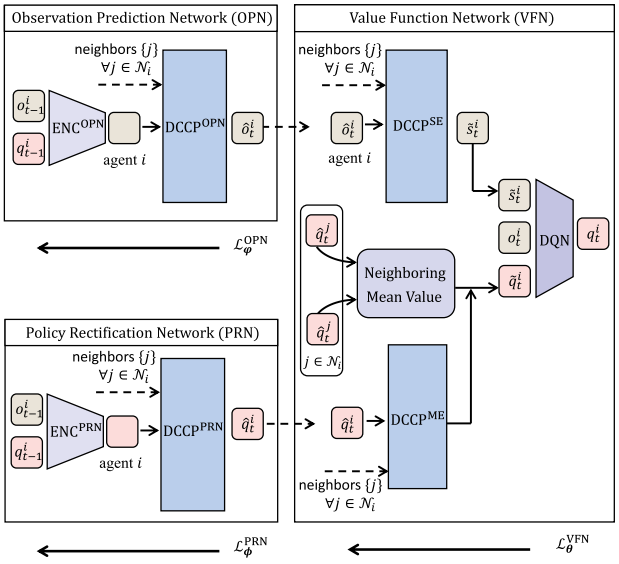


Fig. 3. Network architecture of our method.

neighbors to predict the agent's real-time observation $\hat{\boldsymbol{o}}_t^i$ as

$$\hat{\boldsymbol{o}}_t^i = f_{\boldsymbol{\varphi}}^{\mathrm{OPN}}\left(\left(\boldsymbol{o}_{t-1}^i, \boldsymbol{q}_{t-1}^i\right), \left\{\left(\boldsymbol{o}_{t-1}^j, \boldsymbol{q}_{t-1}^j\right)\right\}_{j\in\mathcal{N}_i}\right) \tag{17}$$

where $f_{\boldsymbol{\varphi}}^{\mathrm{OPN}}$ denotes the OPN function parameterized by weights $\boldsymbol{\varphi}$. This prediction problem is analogous to learning the state transition model since the Q-values can be considered as the action information equivalently, and is trained in a supervised regression manner using the current observation $\boldsymbol{o}_t^i$ as the ground truth as

$$\mathcal{L}_{\boldsymbol{\varphi}}^{\mathrm{OPN}} = \sum_{i\in N}\left\|\boldsymbol{o}_t^i - \hat{\boldsymbol{o}}_t^i\right\|^2. \tag{18}$$

We use another encoder that is also shared across agents to embed the agent's input $(\boldsymbol{o}_{t-1}^i, \boldsymbol{q}_{t-1}^i)$ into a latent vector with the same dimension as the observation space. Then, we feed the latent vectors of an agent and its neighbors into our DCCP to obtain the predicted real-time observation as

$$\hat{\boldsymbol{o}}_t^i = \mathrm{DCCP}^{\mathrm{OPN}}\left(\mathrm{ENC}^{\mathrm{OPN}}\left(\left(\boldsymbol{o}_{t-1}^i, \boldsymbol{q}_{t-1}^i\right), \left\{\left(\boldsymbol{o}_{t-1}^j, \boldsymbol{q}_{t-1}^j\right)\right\}_{j\in\mathcal{N}_i}\right)\right). \tag{19}$$

The PRN has been described in the last section, and finally, we present the VFN. It evaluates the Q-values $\boldsymbol{q}_t^i$ from the predicted real-time observations $\hat{\boldsymbol{o}}_t$ and the rectified real-time agent interactions $\hat{\boldsymbol{q}}_t$ of an agent and its neighbors, and the agent's real-time observation $\boldsymbol{o}_t^i$ as

$$\boldsymbol{q}_t^i = f_{\boldsymbol{\theta}}^{\mathrm{VFN}}\left(\boldsymbol{o}_t^i, \left(\hat{\boldsymbol{o}}_t^i, \hat{\boldsymbol{q}}_t^i\right), \left\{\left(\hat{\boldsymbol{o}}_t^j, \hat{\boldsymbol{q}}_t^j\right)\right\}_{j\in\mathcal{N}_i}\right) \tag{20}$$

where $f_{\boldsymbol{\theta}}^{\mathrm{VFN}}$ denotes the VFN function parameterized by weights $\boldsymbol{\theta}$, and $\boldsymbol{\theta}$ are shared across agents except for the agent-specific weights in the two DCCPs.

More concretely, VFN contains three parts: the observation sharing for global state estimation (SE), the policy sharing for compensating the mean-field approximation bias, and the deep Q-network (DQN). First, we feed the predicted real-time observations $\hat{\boldsymbol{o}}_t$ of an agent and its neighbors for information sharing and coordination, and obtain the global SE $\tilde{\boldsymbol{s}}_t^i$ as

$$\tilde{\boldsymbol{s}}_t^i = \mathrm{DCCP}^{\mathrm{SE}}\left(\hat{\boldsymbol{o}}_t^i, \left\{\hat{\boldsymbol{o}}_t^j\right\}_{j\in\mathcal{N}_i}\right). \tag{21}$$

Second, to simplify agent interactions, MF-Q expands the Q-function in (11) using Taylor's theorem and drops out the second-order remainders. We attempt to exploit DCCP to compensate for this approximation bias by sharing the predicted real-time Q-values $\hat{\boldsymbol{q}}_t$ across neighboring agents, and obtain a more accurate mean-field estimate (ME) $\tilde{\boldsymbol{q}}_t^i$ as

$$\tilde{\boldsymbol{q}}_t^i = \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i}\hat{\boldsymbol{q}}_t^j + \mathrm{DCCP}^{\mathrm{ME}}\left(\hat{\boldsymbol{q}}_t^i, \left\{\hat{\boldsymbol{q}}_t^i\right\}_{j\in\mathcal{N}_i}\right) \tag{22}$$

which consists of the rectified mean-value of neighboring agents plus a DCCP-based compensation term. Since this DCCP is trained using the DQN loss directly, it has the potential to implicitly compensate for the second-order remainders in an end-to-end manner.

Third, the DQN takes the concatenation of the real-time observation $\boldsymbol{o}_t^i$, the estimated global state $\tilde{\boldsymbol{s}}_t^i$, and the enhanced mean-field estimate $\tilde{\boldsymbol{q}}_t^i$ as input as

$$\boldsymbol{s}_t^i = \left[\boldsymbol{o}_t^i, \tilde{\boldsymbol{s}}_t^i, \tilde{\boldsymbol{q}}_t^i\right]. \tag{23}$$

The output of the DQN is denoted as $\boldsymbol{q}_t^i = Q(\cdot|\boldsymbol{s}_t^i)$, and the loss function is formalized as the Bellman residual as

$$\mathcal{L}_{\boldsymbol{\theta}}^{\mathrm{VFN}} = \sum_{i\in N}\left(y_t^i - Q\left(a_t^i|\boldsymbol{s}_t^i\right)\right)^2 \tag{24}$$

in which the bootstrapped target y_t^i is calculated as

$$y_t^i = \begin{cases} r_t^i, & \text{if terminate} \\ r_t^i + \gamma \max\limits_{a'} Q^{\mathrm{target}}\left(a'|\boldsymbol{s}_{t+1}^i\right), & \text{otherwise} \end{cases} \tag{25}$$

where Q^{target} is the target network with parameters copied from some previous version of the DQN.

Together, the loss function of our method is aggregated as

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\varphi}) = \mathcal{L}_{\boldsymbol{\theta}}^{\mathrm{VFN}} + \lambda_1\mathcal{L}_{\boldsymbol{\phi}}^{\mathrm{PRN}} + \lambda_2\mathcal{L}_{\boldsymbol{\varphi}}^{\mathrm{OPN}} \tag{26}$$

where λ_1 and λ_2 are coefficients that balance the influence of three modules' loss functions.[2] Correspondingly, the integrated algorithm is summarized as shown in Algorithm 1. The network parameters $\boldsymbol{\theta}$, $\boldsymbol{\phi}$, and $\boldsymbol{\varphi}$ are shared across agents except for the agent-specific weights $\boldsymbol{w}^i$. Specifically, the agent-specific weights $\boldsymbol{w}^i$ are updated as

$$\begin{aligned}\boldsymbol{w}^i \leftarrow \boldsymbol{w}^i - \alpha\nabla_{\boldsymbol{w}^i}\Big[&\left(y_\tau^i - Q\left(\boldsymbol{s}_\tau^i, a_\tau^i\right)\right)^2 \\ &+ \lambda_1\left\|\boldsymbol{q}_\tau^i - \hat{\boldsymbol{q}}_\tau^i\right\|^2 + \lambda_2\left\|\boldsymbol{o}_\tau^i - \hat{\boldsymbol{o}}_\tau^i\right\|^2\Big] \quad \forall i \in N.\end{aligned} \tag{27}$$

[2]The supervised prediction of OPN and PRN involves online learning, which can be stabilized by the experience replay mechanism, analogous to the training of deep Q-network [7]. Moreover, we may increase the update frequency of OPN and PRN to stabilize the supervised prediction modules before focusing on training the RL module VFN.

Algorithm 1 DCCP-Based MARL With Enhanced Mean-Field Approximation

Initialize exploration ratio ϵ, learning rate α, and coefficients λ_1, λ_2
Define a transition e_t^i $(i = 1, \ldots, N)$ in replay buffer $\mathcal{B}$ as

$$e_t^i = \left\{\boldsymbol{o}_{t-1}^i, \boldsymbol{q}_{t-1}^i, \boldsymbol{o}_t^i, \boldsymbol{a}_t^i, r_t^i, \boldsymbol{q}_t^i, \boldsymbol{o}_{t+1}^i\right\}$$

Randomly initialize the parameters $\boldsymbol{\phi}$, $\boldsymbol{\varphi}$, $\boldsymbol{\theta}$
while not converge **do**
  Initialize $\boldsymbol{q}_0^i$ and $\boldsymbol{o}_0^i$ for each agent i
  for $t = 1, \ldots, T$ (terminal) **do**
    for each agent $i = 1, \ldots, N$ **do**
      With probability ϵ select a random action a_t^i, otherwise $a_t^i = \arg\max_a \boldsymbol{q}_t^i$
    end for
    Take joint action $[a_t^1, \ldots, a_t^N]$, obtain r_t^i and $\boldsymbol{o}_{t+1}^i$
    Store transitions e_t^i into $\mathcal{B}$
  end for
  Sample a minibatch of transitions e_τ^i from $\mathcal{B}$
  Calculate $\hat{\boldsymbol{o}}_\tau^i$, $\hat{\boldsymbol{q}}_\tau^i$, and y_τ^i using (19), (16), and (25)
  Perform a gradient descent step as

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \sum_{i \in N} \nabla_{\boldsymbol{\theta}} \left(y_\tau^i - Q\left(\boldsymbol{s}_\tau^i, a_\tau^i\right)\right)^2$$
$$\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \alpha\lambda_1 \sum_{i \in N} \nabla_{\boldsymbol{\phi}} ||\boldsymbol{q}_\tau^i - \hat{\boldsymbol{q}}_\tau^i||^2$$
$$\boldsymbol{\varphi} \leftarrow \boldsymbol{\varphi} - \alpha\lambda_2 \sum_{i \in N} \nabla_{\boldsymbol{\varphi}} ||\boldsymbol{o}_\tau^i - \hat{\boldsymbol{o}}_\tau^i||^2$$

  Update $Q^{\text{target}} \leftarrow Q$ every η steps
end while

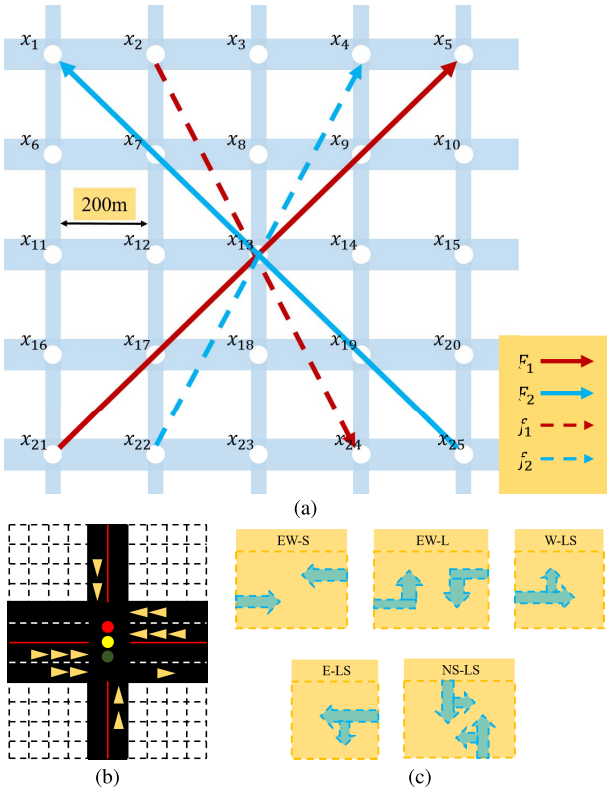


Fig. 4. Simulation environment of ATSC. (a) Traffic gird of 25 intersections with four example flows. (b) Intersection. (c) Possible actions.

V. Experiments

We conduct experiments applying the proposed method to a mixed-cooperative ATSC task on the SUMO platform and to a fully cooperative SMAC task, to imply the method's ability in general cooperative circumstances to achieve efficient coordination to achieve high returns. We compare our method to the following baselines: the CTDE-based QMIX [25], [26] and the global communication learning-based nearly decomposable Q-functions (NDQ) [48] for the fully-cooperative SMAC; the global communication learning-based TarMAC [30] and the local communication learning-based IC3Net [36] for both tasks.[3] Then, we perform an ablation study to verify the respective effectiveness of the two components in our method. IQL [20] is set as the baseline approach that removes the two components from our method, and DCCP is a variant of our method that only removes the enhanced mean-field approximation. The effect of DCCP is demonstrated by comparing DCCP with IQL, and the effect of the enhanced mean-field approximation is demonstrated by comparing our method with DCCP. Since the mean-field approximation is partly enhanced by DCCP, we do not consider the variant of only removing the DCCP component. All results are averaged over ten seeds. The shaded area represents the 95% confidence interval for evaluation curves, and the standard errors are presented for numerical results.

[3]The value function factorization methods, e.g., QMIX and NDQ, use a mixture network to predict the global value function and are only feasible for fully-cooperative tasks. Hence, we do not evaluate them in the mixed-cooperative ATSC environments where each agent has its own objective of controlling the local traffic situation.

A. Adaptive Traffic Signal Control

We conduct ATSC in a 5 × 5 synthetic grid map on the standard traffic simulation platform SUMO, where each agent (traffic intersection) observes a local part of the shared environment and controls phases of its traffic signals. The objective of ATSC is to eliminate the traffic congestion in each intersection to generate low congestion of the whole traffic system, i.e., minimizing the queue length and time delay. The mixed cooperation occurs as each agent takes the traffic situations in its local intersection and in the whole traffic network as the objective. Fig. 4(a) shows a synthetic traffic network that consists of multiple homogeneous intersections in a 5 × 5 grid map. Let E, N, W, and S denote east, north, west, and south, respectively. Each intersection consists of two E-W two-lane streets and two N-S one-lane avenues, as shown in Fig. 4(b). Each intersection has five available phases (corresponding to the actions in RL settings) that are EW-S, EW-L, W-LS, E-LS, and NS-LS, as shown in Fig. 4(c).

Following the same experiment setting in [54] and [55], we use the traffic flow to describe the vehicles in the simulation. The traffic flow corresponds to generating vehicles arriving at the traffic network with an origin and a destination, i.e., an origin-destination (O-D) pair. An O-D pair is denoted as x_j-x_k, where the origin x_j and the destination x_k denote two different intersections in the traffic network as shown in Fig. 4(a). In the simulation, the peak-hour traffic dynamics

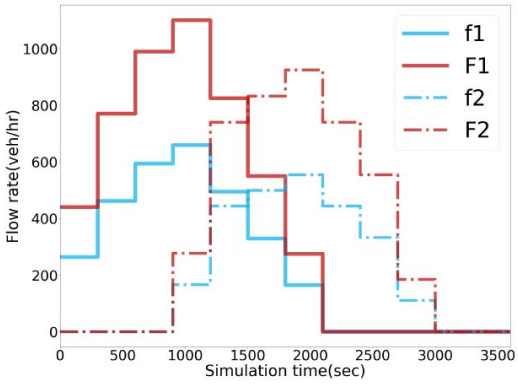


Fig. 5. Example of the time-varying function of traffic flows in ATSC.

is constituted by four time-varying traffic flows: F_1, F_2, f_1, and f_2, each of which generates vehicles according to three different O-D pairs. We show the flows F_1 and f_1 and their O-D pairs in Fig. 4(a). F_1 consists of three O-D pairs across the main streets: x_1-x_{25}, x_{11}-x_{15}, and x_{21}-x_5. f_1 consists of three O-D pairs across the side avenues: x_2-x_{24}, x_3-x_{23}, and x_4-x_{22}. Similarly, the flows F_2 and f_2 generate vehicles according to O-D pairs that are opposite to those of F_1 and f_1, respectively. In the simulation, the vehicles are generated from the four traffic flows with time-varying flow rates. Fig. 5 presents the flow rate function in our simulation of the four traffic flows.

The local observation of the ith agent is defined as

$$o_t^i = \left\{\left(\text{time\_delay}_t[l], \text{wave}_t[l]\right)\right\}_{l\in L_i} \tag{28}$$

where l is an incoming lane of the ith intersection, and L_i is the set of all incoming lanes of intersection i. $\text{time\_delay}_t[l]$(s) measures the cumulative delayed time of the first vehicle in lane l at step t, and $\text{wave}_t[l]$(veh) measures the total number of approaching vehicles along lane l within 50 m to intersection i at step t. The action of the ith agent is represented as a one-hot vector that denotes selecting one traffic phase from Fig. 4(c) as

$$a_t^i \in \{\text{EW-S, EW-L, W-LS, E-LS, NS-LS}\}. \tag{29}$$

Specifically, when the next action is different from the last one, an all-yellow phase will appear and last for 2 s to ensure a safe switch between different phases. The reward of the ith agent is defined as the weighted sum of the queue length and time delay of the vehicles at the corresponding intersection as

$$r_t^i = -\sum_{l\in L_i}\left(\text{queue\_len}_{t+1}[l] + w\cdot \text{time\_delay}_{t+1}[l]\right) \tag{30}$$

where w is the trade-off coefficient that is set as 0.2, $\text{queue\_len}_{t+1}[l]$ is the measured number of vehicles in the waiting queue, and the $\text{time\_delay}_{t+1}[l]$ is the cumulative delay time (in seconds) of the first vehicle along each incoming lane l at the next time step.

We set the episode length to 720 where each learning step takes 5 s in the simulator. After training for 1 M steps, we evaluate the tested algorithms for one episode. The performance metrics are the queue length and time delay at the traffic intersections in evaluation. For each metric, we record the average value over all steps of the evaluation episode and the final value at the end of the evaluation. The kernel size in our DCCP is fixed as 3 × 3, and each agent can only access the information of its neighbors within the 3 × 3 area. Tables I and II present training hyperparameters and the network architecture, respectively.

TABLE I
TRAINING HYPERPARAMETERS OF OUR METHOD IN ATSC

Task	replay buffer size	batch size	λ_1	λ_2	α	optimizer
ATSC	4000	240	0.5	0.5	1e-6	RMSProp

TABLE II
NETWORK ARCHITECTURE OF OUR METHOD IN ATSC

Modules	Encoder - layer size			DCCP	
	input	hidden	output	# kernels	kernel size
OPN	$17\ (12+5)$	256	12	10	3×3
PRN	$17\ (12+5)$	256	5	10	3×3
VFN	$29\ (12+12+5)$	128	5	10	3×3

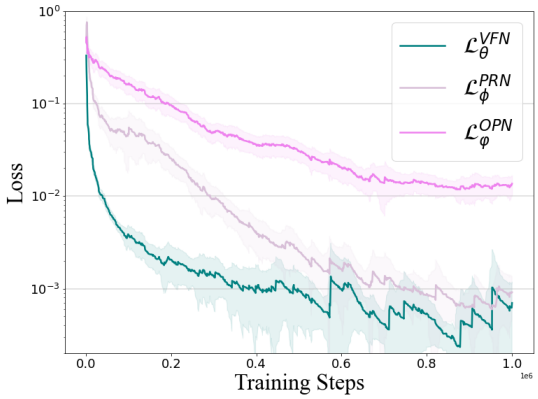
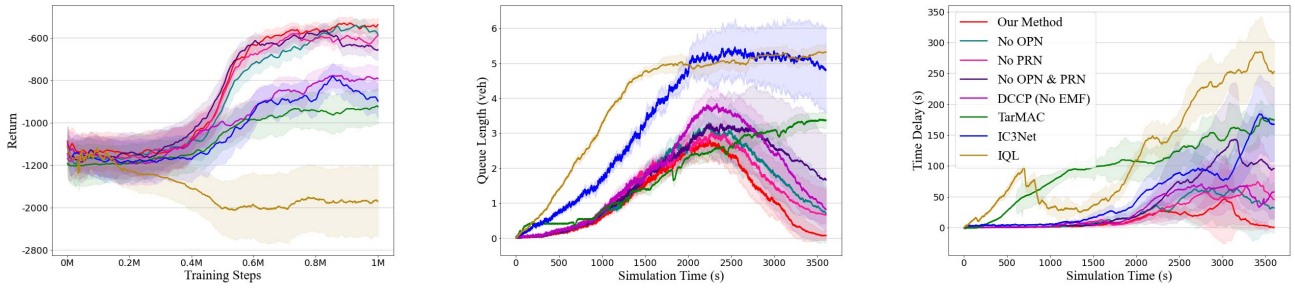


Fig. 6. Training losses of the three modules of our method in ATSC.

First, we simply record the communication overhead. IQL incurs no communication, and TarMAC takes 24∗25 rounds of communication as each agent needs to receive the information from the other 24 agents. In contrast, IC3Net and our method only cost $8 * 25$ rounds of communication as each agent only needs to receive the information from its eight neighbors within the convolution kernel. It can be observed that local communication saves communication resources compared to the global protocol, especially when the scale of the neighborhood is far smaller than the total scale.

Fig. 6 shows the training loss curves of the OPN, PRN, and VFN modules in our method. It can be observed that the three modules are stably trained as the losses generally decrease during training. Fig. 7 presents the evaluation curves of the received return, queue length, and time delay at the intersections, and Table III gives the numerical results in terms of the average and final values in evaluation. Overall, our method performs the best and solves the ATSC task with its efficient coordination. Using our method, the traffic congestion quickly decreases after peak flows, and is almost



(a) (b) (c)

Fig. 7. Evaluation results of the received return, queue length, and time delay in ATSC. (a) Received return. (b) Queue length. (c) Time delay.

TABLE III

NUMERICAL RESULTS OF THE AVERAGE QUEUE LENGTH, FINAL QUEUE LENGTH, AVERAGE TIME DELAY, AND FINAL TIME DELAY IN ATSC

Methods	avg_queue_len (veh)	final_queue_len (veh)	avg_time_delay (s)	final_time_delay (s)
IC3Net	3.39 ± 0.27	4.84 ± 0.76	66.42 ± 9.26	218.91 ± 25.43
TarMAC	1.80 ± 0.06	3.37 ± 0.11	97.06 ± 3.23	174.96 ± 11.45
IQL	3.91 ± 0.07	5.32 ± 0.11	118.35 ± 7.01	253.22 ± 26.12
DCCP (No EMF)	1.76 ± 0.10	0.87 ± 0.32	27.30 ± 4.51	58.14 ± 23.42
No OPN	1.50 ± 0.14	0.69 ± 0.40	22.58 ± 6.21	32.12 ± 13.55
No PRN	1.40 ± 0.11	0.68 ± 0.36	22.17 ± 5.76	45.82 ± 15.57
No OPN & PRN	1.73 ± 0.19	1.66 ± 0.58	40.03 ± 9.29	95.81 ± 19.96
Our Method	$\mathbf{1.16 \pm 0.08}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{8.67 \pm 1.72}$	$\mathbf{0.03 \pm 0.03}$

(a) (b) (c) (d)

Fig. 8. Example screenshots of evaluation combat scenarios SMAC. (a) 3 m combat scenario. (b) 8 m combat scenario. (c) 5 m versus 6 m combat scenario. (d) 2 c versus 64 zg combat scenario.

eliminated at the end as both the final queue length and time delay are approximately zero. IC3Net obtains unsatisfactory performance as the traffic congestion quickly increases after peak flows and remains at a high level for a long time. Since IC3Net simply averages the messages from neighboring agents, it cannot differentiate valuable information that helps cooperative decision making. TarMAC performs slightly better than IC3Net, which is supposed to benefit from allowing each agent to actively select which agents to address messages to.

Next, we conduct a comprehensive ablation study. IQL performs the worst due to the lack of communication across agents. The distinct superiority of DCCP over IQL verifies the effectiveness of our communication protocol. Moreover, DCCP mostly obtains better performance than IC3Net and TarMAC, which demonstrates the advantage of our communication protocol over other communication learning approaches. At last, the performance improvement of our method over DCCP shows that the enhanced mean-field approximation can further facilitate multi-agent coordination. Further, we perform the ablation study to investigate the effects of the OPN and PRN modules on our method, corresponding to the variants of removing OPN (No OPN), removing PRN (No PRN), and removing both (No OPN and PRN).[4] The results show that the performance of our method degenerates when removing either module of OPN and PRN, and degenerates more when removing both modules. It successfully verifies that both the OPN and PRN modules boost the multi-agent coordination due to facilitating more accurate interactions between agents.

B. StarCraft II Multi-Agent Challenge

As benchmark testbeds, SMAC provides fully cooperative battles for a group of agents learning to defeat the opponent team. Typically, the game is framed as a competitive

[4]We ablate the OPN loss $\mathcal{L}_{\varphi}^{\text{OPN}}$ or the PRN loss $\mathcal{L}_{\phi}^{\text{PRN}}$ in (26) when removing the OPN or the PRN module, respectively, and we ablate both losses when removing the two modules together. During ablation, we keep the network architecture unchanged.

TABLE IV
ENVIRONMENT SETTINGS OF THE FOUR EVALUATION COMBAT SCENARIOS IN SMAC

Scenarios	Difficulty	Controlled agents	Opponents	Max steps	Observation size	Action size
3m	very-hard	3 marines	3 marines	60	30	9
8m	very-hard	8 marines	8 marines	120	80	14
5m_vs_6m	very-hard	5 marines	6 marines	70	55	12
2c_vs_64zg	very-hard	2 colossi	64 zerglings	400	332	70

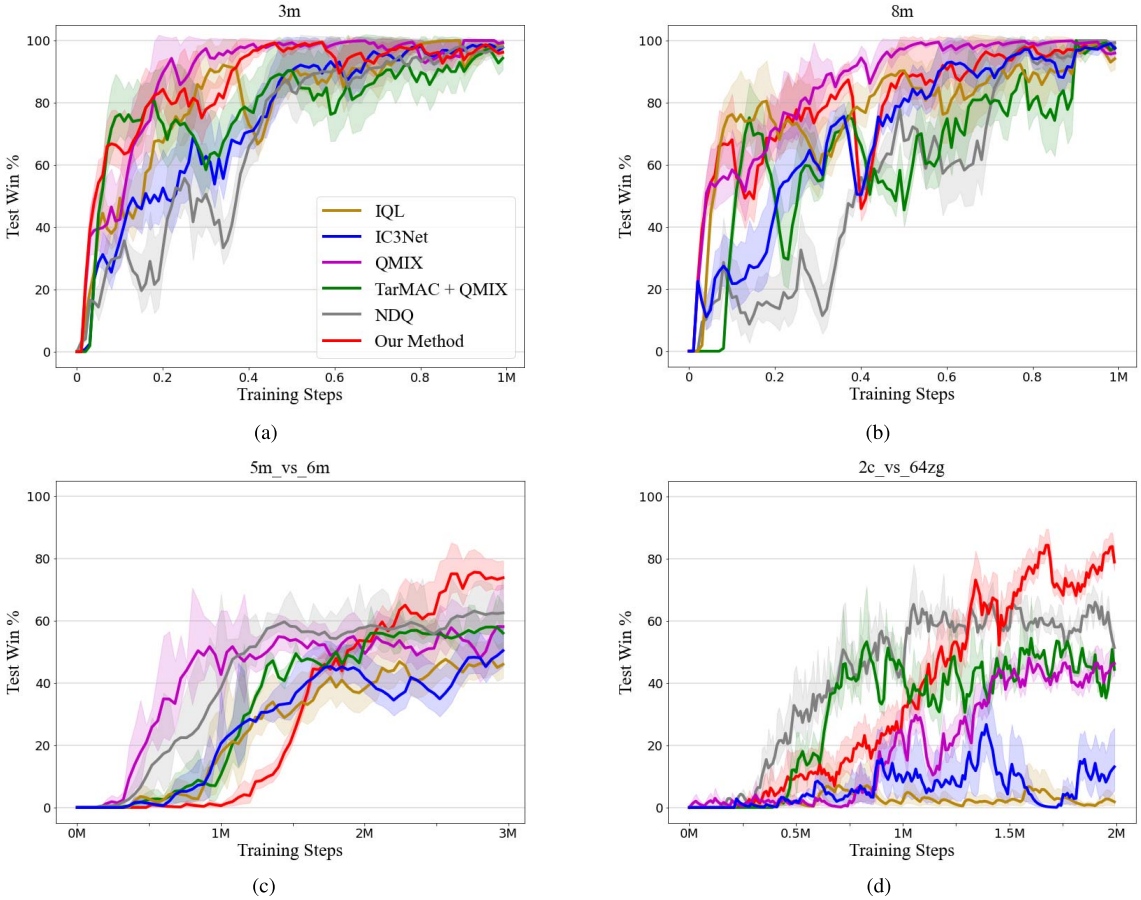


Fig. 9. Win rate during training in SMAC. (a) 3 m, *very-hard*. (b) 8 m, *very-hard*. (c) 5 m versus 6 m, *very-hard*. (d) 2 c versus 64 zg, *very-hard*.

problem: an agent takes the role of a human player, making macromanagement decisions and performing micromanagement as a puppeteer that issues orders to individual units from a centralized controller. In SMAC, each unit's actions are conditioned on local observations instead of the global game state. Then, in several challenging combat scenarios, the group of these independent agents battles an opposing army under the centralized control of the build-in game AI. These challenges require agents to learn cooperative behaviors under partial observability, e.g., *focus fire* and *avoid overkill*. Each scenario consists of two armies of units battling to defeat each other, one controlled by the learner and the other controlled by the built-in game AI. An episode ends when all units of either army have died or when a predefined step limit is reached. The goal is to maximize the win rate of the learned policies, i.e., the expected ratio of games won to games played.

TABLE V
TRAINING HYPERPARAMETERS OF OUR METHOD IN SMAC. T IS THE NUMBER OF MAXIMUM STEPS IN EACH LEARNING EPISODE

Task	replay buffer size	batch size	λ_1	λ_2	α	optimizer
SMAC	$32 \times T$	32	0.05	0.01	5e-4	Adam

Agents receive local observations drawn within their field of view, which is called *sight range*. Agents can only observe other agents if they are both alive and located within the sight range. For agent i, the observation is formalized as

$$o_t^i = \Big\{\big(\text{distance}_{i,j}, \Delta x_{i,j}, \Delta y_{i,j}, \text{health}_j, \text{sheild}_j, \text{unit type}_j\big)\Big\}_{\forall j \in} \mathcal{U} \quad (31)$$

where j is a unit in the unit set $\mathcal{U}$ within the battle, $\text{distance}_{i,j}$ is the distance between units i and j, and $\Delta x_{i,j}$ and $\Delta y_{i,j}$ are

TABLE VI
NETWORK ARCHITECTURE OF OUR METHOD IN SMAC

Scenarios	Modules	Encoder - layer size					DCCP	
		input	hidden #1	hidden #2	LSTM size	output	# kernels	kernel size
3m	OPN	39 (30 + 9)	256	64	64	30	10	3×3
	PRN	39 (30 + 9)			64	9		
	VFN	69 (30 + 30 + 9)			-	9		
8m	OPN	94 (80 + 14)	256	64	64	60	10	5×5
	PRN	94 (80 + 14)			64	14		
	VFN	174 (80 + 80 + 14)			-	14		
5m_vs_6m	OPN	67 (55 + 12)	256	64	64	55	10	3×3
	PRN	67 (55 + 12)			64	12		
	VFN	122 (55 + 55 + 12)			-	12		
2c_vs_64zg	OPN	402 (332 + 70)	256	64	64	32	10	3×3
	PRN	402 (332 + 70)			64	70		
	VFN	734 (332 + 332 + 70)			-	70		

TABLE VII
EVALUATION WIN RATE OF THE TRAINED MODELS IN SMAC

Methods	3m, hard	8m, hard	5m_vs_6m, hard	2c_vs_64zg, hard
IQL	$99 \pm 0.03\%$	$99 \pm 0.69\%$	$48 \pm 5.26\%$	$7 \pm 4.91\%$
IC3Net	$98 \pm 0.93\%$	$99 \pm 0.67\%$	$50 \pm 5.01\%$	$27 \pm 8.62\%$
QMIX	$\mathbf{100 \pm 0.14\%}$	$99 \pm 0.06\%$	$58 \pm 8.11\%$	$48 \pm 3.11\%$
TarMAC + QMIX	$98 \pm 1.99\%$	$100 \pm 0.12\%$	$59 \pm 2.50\%$	$54 \pm 8.20\%$
NDQ	$98 \pm 1.41\%$	$\mathbf{100 \pm 0.08\%}$	$63 \pm 5.73\%$	$66 \pm 2.17\%$
Our Method	$\mathbf{100 \pm 0.38\%}$	$99 \pm 0.93\%$	$\mathbf{76 \pm 4.49\%}$	$\mathbf{84 \pm 3.18\%}$

the offsets between i and j in axes x and y, respectively. health_j, sheild_j, and unit type_j denote the attributes of unit j. Specifically, if unit j is not in the sight range of i, the above observation vectors related to unit j are padded with 0.

The available actions consist of move[direction] (four directions: north, south, east, and west), attack[enemy_id], stop, and no_op. Dead agents can only take no_op action while alive agents cannot. Specifically, when unit j is out of the *shooting range* of unit i, the action of attacking j is not available. We formalize the action space of agent i as

$$a_t^i \in \{\text{no\_op}, \{\text{move[direction}d\text{]}\}, \quad \{\text{attack unit } j\}_{j \in \mathcal{U}}\}. \tag{32}$$

The reward contains three parts of the damage to enemies, killing enemies, and winning the battle as

$$r_t^i = \sum_{j \in E} \left(\Delta\text{health}_j + \Delta\text{shield}_j\right) + 10 \cdot \Delta\#\text{ dead enemies} + 200 \cdot \text{win flag} \tag{33}$$

where $\Delta\#$ dead enemies is the increased number of dead enemies, and win flag is the flag of winning the combat.

Since many QMIX-based approaches are verified to achieve good performance in fully cooperative tasks, we take the following state-of-the-art MARL algorithms as baselines: QMIX, TarMAC+QMIX, and NDQ.[5] We evaluate the methods on four combat maps: 3 m, 8 m, 5 m versus 6 m, and 2 c versus 64 zg. In these maps, the neighboring relationship between agents is predefined according to their initial positions, and the convolution kernel of DCCP is with the same size as the kernel in ATSC. With this neighborhood formalization, we are capable of evaluating our method beyond the grid-world environments. Fig. 8 shows example screenshots of the four combat scenarios, and Table IV presents the settings of the evaluation combat scenarios. The difficulty of StarCraft II built-in AI is set to *very-hard* in all maps. In each episode, positive rewards are given for the positive health point difference between the controlled agent team and the opponent, and otherwise, the reward is zero. A large positive reward is given for winning the episode by eliminating the opponent, and otherwise, the reward is zero. We evaluate the tested methods per 100 k steps during training, and in each evaluation, we run the game for 20 episodes to calculate the win rate. Tables V and VI present training hyperparameters and the network architecture, respectively.

Fig. 9 presents evaluation curves of the win rate in the scenarios with *very-hard* built-in AIs, and Table VII shows numerical results of evaluating the trained models. It is observed that our method generally achieves the highest and the most stable win rate in these combat maps. The performance gap in terms of the final win rate is more pronounced in the complex combat maps of 5 m versus 6 m and 2 c versus 64 zg, which demonstrates the capability of our method for efficiently facilitating coordination across agents. Moreover, our method learns the effect of joint actions without using the privileged state information from the environment, while

[5]NDQ itself is based on QMIX.

SMAC can provide access to this information for methods like TarMAC. This makes our method applicable for a wider range of scenarios, as in many multi-agent tasks we do not have access to privileged full state information even during training.

VI. Conclusion

In this article, we propose a new MARL method based on local communication learning. We facilitate efficient coordination between neighboring agents by exploiting the ability of depthwise convolution to learn a local communication protocol, and by enhancing the mean-field approximation with a supervised PRN and a learnable compensation term. Empirical results and an ablation study show that our method achieves efficient coordination and outperforms several baseline approaches on the ATSC and SMAC tasks. Our future work will focus on learning more efficient communication protocols using graph structures and attention mechanisms. Another insightful direction would be to develop efficient local communication protocols for more complex multi-agent systems where the neighborhood may vary with a dynamic communication topology.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[2] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

[3] J.-A. Li et al., "Quantum reinforcement learning during human decision-making," *Nature Human Behaviour*, vol. 4, no. 3, pp. 294–307, Jan. 2020.

[4] Y. Zheng et al., "Efficient policy detecting and reusing for non-stationarity in Markov games," *Auto. Agents Multi-Agent Syst.*, vol. 35, no. 1, pp. 1–29, Apr. 2021.

[5] H. Li, Z. Qichao, and D. Zhao, "Deep reinforcement learning-based automatic exploration for navigation in unknown environment," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 6, pp. 2064–2076, Jun. 2020.

[6] Z. Wang, C. Chen, H.-X. Li, D. Dong, and T.-J. Tarn, "Incremental reinforcement learning with prioritized sweeping for dynamic environments," *IEEE/ASME Trans. Mechatronics*, vol. 24, no. 2, pp. 621–632, Apr. 2019.

[7] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[8] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[10] Z. Wang, C. Chen, and D. Dong, "Lifelong incremental reinforcement learning with online Bayesian inference," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 8, pp. 4003–4016, Aug. 2022.

[11] J. Pan, X. Wang, Y. Cheng, and Q. Yu, "Multisource transfer double DQN based on actor learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2227–2238, Jun. 2018.

[12] B. Luo, Y. Yang, H.-N. Wu, and T. Huang, "Balancing value iteration and policy iteration for discrete-time control," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 50, no. 11, pp. 3948–3958, Nov. 2020.

[13] Z. Wang, C. Chen, and D. Dong, "A Dirichlet process mixture of robust task models for scalable lifelong reinforcement learning," *IEEE Trans. Cybern.*, early access, May 17, 2022, doi: 10.1109/TCYB.2022.3170485.

[14] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *Proc. 36th Int. Conf. Mach. Learn.*, vol. 97, Jun. 2019, pp. 2961–2970.

[15] Y. Liu, W. Wang, Y. Hu, J. Hao, X. Chen, and Y. Gao, "Multi-agent game abstraction via graph attention neural network," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 7211–7218.

[16] J. Jiang, C. Dun, T. Huang, and Z. Lu, "Graph convolutional reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–13.

[17] O. Vinyals et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[18] Z. Ding, T. Huang, and Z. Lu, "Learning individually inferred communication for multi-agent cooperation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020.

[19] Z. Yu et al., "MaCAR: Urban traffic light control via active multi-agent communication and action rectification," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 2491–2497.

[20] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proc. Int. Conf. Mach. Learn.*, 1993, pp. 330–337.

[21] R. Lowe et al., "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6379–6390.

[22] J. Foerster et al., "Counterfactual multi-agent policy gradients," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 2974–2982.

[23] P. Sunehag et al., "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proc. Int. Conf. Auto. Agents Multiagent Syst.*, 2018, pp. 2085–2087.

[24] W. Qiu, H. Chen, and B. An, "Dynamic electronic toll collection via multi-agent deep reinforcement learning with edge-based graph convolutional networks," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 4568–4574.

[25] T. Rashid et al., "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4295–4304.

[26] T. Rashid, G. Farquhar, B. Peng, and S. Whiteson, "Weighted QMIX: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 10199–10210.

[27] T. Gupta, A. Mahajan, B. Peng, W. Boehmer, and S. Whiteson, "UneVEn: Universal value exploration for multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, vol. 139, 2021, pp. 3930–3941.

[28] T. Zhang, Y. Li, C. Wang, G. Xie, and Z. Lu, "FOP: Factorizing optimal joint policy of maximum-entropy multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, vol. 139, 2021, pp. 12491–12500.

[29] S. Sukhbaatar, A. Szlam, and R. Fergus, "Learning multi-agent communication with backpropagation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2244–2252.

[30] A. Das et al., "TarMAC: Targeted multi-agent communication," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 1538–1546.

[31] M. Lv, B. De Schutter, C. Shi, and S. Baldi, "Logic-based distributed switching control for agents in power-chained form with multiple unknown control directions," *Automatica*, vol. 137, Mar. 2022, Art. no. 110143.

[32] J. Foerster et al., "Learning to communicate with deep multi-agent reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2137–2145.

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

[34] R. Wang et al., "Learning efficient multi-agent communication: An information bottleneck approach," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 9908–9918.

[35] J. Jiang and Z. Lu, "Learning attentional communication for multi-agent cooperation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 7254–7264.

[36] A. Singh, T. Jain, and S. Sukhbaatar, "Learning when to communicate at scale in multiagent cooperative and competitive tasks," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–15.

[37] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[38] A. G. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[39] Y. Yang et al., "Mean field multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5571–5580.

[40] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO—Simulation of urban mobility: An overview," in *Proc. 3rd Int. Conf. Adv. Syst. Simul.*, 2011, pp. 1–6.

[41] M. Samvelyan et al., "The StarCraft multi-agent challenge," in *Proc. Int. Conf. Auto. Agents Multiagent Syst.*, 2019, pp. 2186–2188.

[42] X. Jin, Y. Shi, Y. Tang, H. Werner, and J. Kurths, "Event-triggered fixed-time attitude consensus with fixed and switching topologies," *IEEE Trans. Autom. Control*, vol. 67, no. 8, pp. 4138–4145, Aug. 2022.

[43] M. Lv, W. Yu, J. Cao, and S. Baldi, "A separation-based methodology to consensus tracking of switched high-order nonlinear multiagent systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 10, pp. 5467–5479, Oct. 2022.
[44] X. Jin, Y. Shi, Y. Tang, and X. Wu, "Event-triggered attitude consensus with absolute and relative attitude measurements," *Automatica*, vol. 122, Dec. 2020, Art. no. 109245.
[45] M. Lv, W. Yu, J. Cao, and S. Baldi, "Consensus in high-power multiagent systems with mixed unknown control directions via hybrid Nussbaum-based control," *IEEE Trans. Cybern.*, vol. 52, no. 6, pp. 5184–5196, Jun. 2022.
[46] M. Rangwala and R. Williams, "Learning multi-agent communication through structured attentive reasoning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 10088–10098.
[47] S. Li et al., "Deep implicit coordination graphs for multi-agent reinforcement learning," in *Proc. Int. Conf. Auto. Agents Multiagent Syst.*, 2021, pp. 764–772.
[48] T. Wang, J. Wang, C. Zheng, and C. Zhang, "Learning nearly decomposable value functions via communication minimization," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–15.
[49] X. Jin, S. Mao, L. Kocarev, C. Liang, S. Wang, and Y. Tang, "Event-triggered optimal attitude consensus of multiple rigid body networks with unknown dynamics," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 5, pp. 3701–3714, Sep. 2022.
[50] G. Tesauro, "Extending Q-learning to general adaptive multi-agent systems," in *Proc. Adv. Neural Inf. Process. Syst.*, 2004, pp. 871–878.
[51] J. Foerster et al., "Stabilising experience replay for deep multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1146–1155.
[52] C. Domb, *Phase Transitions and Critical Phenomena*. Amsterdam, The Netherlands: Elsevier, 2000.
[53] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," *Mach. Learn. Proc.*, vol. 1994, pp. 157–163, Jul. 1994.
[54] T. Chu, J. Wang, L. Codecà, and Z. Li, "Multi-agent deep reinforcement learning for large-scale traffic signal control," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 3, pp. 1086–1095, Mar. 2019.
[55] T. Chu, S. Chinchali, and S. Katti, "Multi-agent reinforcement learning for networked system control," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–17.

Donghan Xie received the B.E. degree in mechanical engineering from the School of Mechanical Engineering, Shandong University, Jinan, China, in 2018, and the M.S. degree from the Department of Control Science and Intelligence Engineering, School of Management and Engineering, Nanjing University, Nanjing, China, in 2021.

His current research interests include multi-agent reinforcement learning and machine learning.

Zhi Wang (Member, IEEE) received the B.E. degree in automation from Nanjing University, Nanjing, China, in 2015, and the Ph.D. degree in machine learning from the Department of Systems Engineering and Engineering Management, The City University of Hong Kong, Hong Kong, China, in 2019.

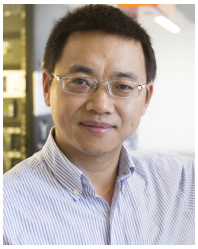
He is currently an Associate Research Fellow with the Department of Control Science and Intelligence Engineering, School of Management and Engineering, Nanjing University. He holds visiting positions at the University of New South Wales, Canberra, ACT, Australia, and the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China. His current research interests include reinforcement learning, machine learning, and robotics.

Dr. Wang served as an Associate Editor for the IEEE International Conference on Systems, Man, and Cybernetics 2021 and 2022, and the IEEE International Conference on Networking, Sensing, and Control 2020.

Chunlin Chen (Senior Member, IEEE) received the B.E. degree in automatic control and the Ph.D. degree in control science and engineering from the University of Science and Technology of China, Hefei, China, in 2001 and 2006, respectively.

He was a Visiting Scholar at Princeton University, Princeton, NJ, USA, from 2012 to 2013. He is currently a Full Professor and the Vice Dean at the School of Management and Engineering, Nanjing University, Nanjing, China. He had visiting positions at the University of New South Wales, Canberra, ACT, Australia, and The City University of Hong Kong, Hong Kong, China. His recent research interests include reinforcement learning, mobile robotics, and quantum control.

Dr. Chen is the Chair of Technical Committee on Quantum Cybernetics, IEEE Systems, Man, and Cybernetics Society.



Daoyi Dong (Fellow, IEEE) received the B.E. degree in automatic control and the Ph.D. degree in engineering from the University of Science and Technology of China, Hefei, China, in 2001 and 2006, respectively.

He was an Alexander von Humboldt Fellow at automatic control and complex systems (AKS), University of Duisburg-Essen, Duisburg, Germany. He was with the Institute of Systems Science, Chinese Academy of Sciences, Beijing, China, and with Zhejiang University, Hangzhou, China. He had visiting positions at Princeton University, Princeton, NJ, USA; RIKEN, Wako-Shi, Japan; and The University of Hong Kong, Hong Kong. He is currently a Scientia Associate Professor at the University of New South Wales, Canberra, ACT, Australia. His research interests include quantum control and machine learning.

Dr. Dong is the Member-at-Large, the Board of Governors, and was the Associate Vice President for Conferences and Meetings, IEEE Systems, Man and Cybernetics Society. He was awarded the ACA Temasek Young Educator Award by the Asian Control Association and was a recipient of Future Fellowship, the International Collaboration Award and the Australian Post-Doctoral Fellowship from the Australian Research Council, and a Humboldt Research Fellowship from the Alexander von Humboldt Foundation of Germany. He served as an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS from 2015 to 2021. He is currently an Associate Editor of the IEEE TRANSACTIONS ON CYBERNETICS and a Technical Editor of the IEEE/ASME TRANSACTIONS ON MECHATRONICS.