Learn to Collaborate in MEC: An Adaptive Decentralized Federated Learning Framework

Yatong Wang, *Member, IEEE*, Zhongyi Wen, Yunjie Li, *Senior Member, IEEE*, and Bin Cao, *Senior Member, IEEE*

***Abstract*—Decentralized federated learning (DFL) has emerged as a conducive paradigm, facilitating a distributed privacy-preserving data collaboration mode in mobile edge computing (MEC) systems to bolster the expansion of artificial intelligence applications. Nevertheless, the dynamic wireless environment and the heterogeneity among collaborating nodes, characterized by skewed datasets and uneven capabilities, present substantial challenges for efficient DFL model training in MEC systems. Consequently, the design of an efficient collaboration strategy becomes essential to facilitate practical distributed knowledge sharing and cost reduction for MEC. In this paper, we propose an adaptive decentralized federated learning framework that enables heterogeneous nodes to learn tailored collaboration strategies, thereby maximizing the efficiency of the DFL training process in collaborative MEC systems. Specifically, we present an effective option critic-based collaboration strategy learning (OCSL) mechanism by decomposing the collaboration strategy model into two sub-strategies: local training strategy and resource scheduling strategy. In addressing inherent issues such as large-scale action space and overestimation in collaboration strategy learning, we introduce the option framework and a dual critic network-based approximation method within the OCSL design. We theoretically prove that the learned collaboration strategy achieves the Nash equilibrium. Extensive numerical results demonstrate the effectiveness of the proposed method in comparison with existing baselines.**

***Index Terms*—Federated learning, mobile edge computing, option-critic, multi-agent reinforcement learning.**

I. Introduction

THE rapid development of mobile edge computing (MEC), characterized by the seamless integration of computing capabilities into diverse edge devices, is a key factor driving the substantial increase in data generation. As per the forecast by the international data corporation, the global data volume is projected to reach 180 zettabytes by 2025, with approximately 70% of the data being produced by various edge devices [1]. Inevitably, the growth of data poses significant communication costs and overhead to the conventional cloud-based centralized data processing mode. Nevertheless, there has been a parallel development in application specific integrated circuits (ASIC) and the computing power of user devices, which has pushed the horizon of processing the data at the edge of the network. Furthermore, advanced MEC technologies facilitate the emergence of a new generation of artificial intelligence (AI) applications, such as Industry 4.0, healthcare, smart city, autonomous vehicles, Metaverse, etc. To provide robust and high-quality service, these advanced applications typically require vast amounts of data collecting from various user devices cooperating with the training of machine learning models. However, with the privacy protection of user data being regarded as the primary consideration and various protection acts having been subsequently legislated, a distributed privacy-preserving data collaboration mode is needed for MEC to harness the benefits of AI applications.

To facilitate distributed data collaboration while mitigating concerns regarding data privacy, federated learning (FL) is burgeoning as a conductive paradigm for supporting data-driven AI applications in MEC. As shown in Fig. 1, two general FL architectures have been proposed: centralized federated learning (CFL) [2], [3], [4], [5], [6], [7], [8] and decentralized federated learning (DFL) [9], [10], [11], [12], [13], [14], [15], [16], [17]. Under the CFL architecture, participating nodes (user devices) push their individual local models to a centralized parameter server (base station), which undertakes model averaging to derive a global model. However, CFL is susceptible to a single point of failure issue, and the parameter server might potentially become a communication and computation bottleneck, especially in scenarios involving massive machine-type connections (mMTC) in MEC. In contrast, the DFL architecture operates without a parameter server, with all participating nodes directly exchanging and aggregating models through interactions with their immediate neighbors. Consequently, DFL is characterized by greater robustness and scalability compared to CFL within the realm of MEC. In this paper, our focus is centered on MEC systems within the architecture of DFL.

However, two challenges are posed in the implementation of DFL architecture over collaborative MEC systems. The first challenge stems from the heterogeneity of MEC systems, where nodes including user devices and base stations are typically

Manuscript received 29 December 2023; revised 13 June 2024; accepted 30 July 2024. Date of publication 19 August 2024; date of current version 5 November 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2021YFB1714100, in part by the National Natural Science Foundation of China under Grant U22B2006, and in part by China Postdoctoral Science Foundation under Grant GZB20240942 and Grant 2024M754090. Recommended for acceptance by J. Rodrigues. *(Corresponding author: Yunjie Li.)*

Yatong Wang and Yunjie Li are with the School of Information and Electronics, Beijing Institute of Technology, Beijing 100081, China (e-mail: wangyatong@bit.edu.cn; liyunjie@bit.edu.cn).

Zhongyi Wen is with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: zy_wen@std.uestc.edu.cn).

Bin Cao is with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: caobin65@163.com).

Digital Object Identifier 10.1109/TMC.2024.3439588

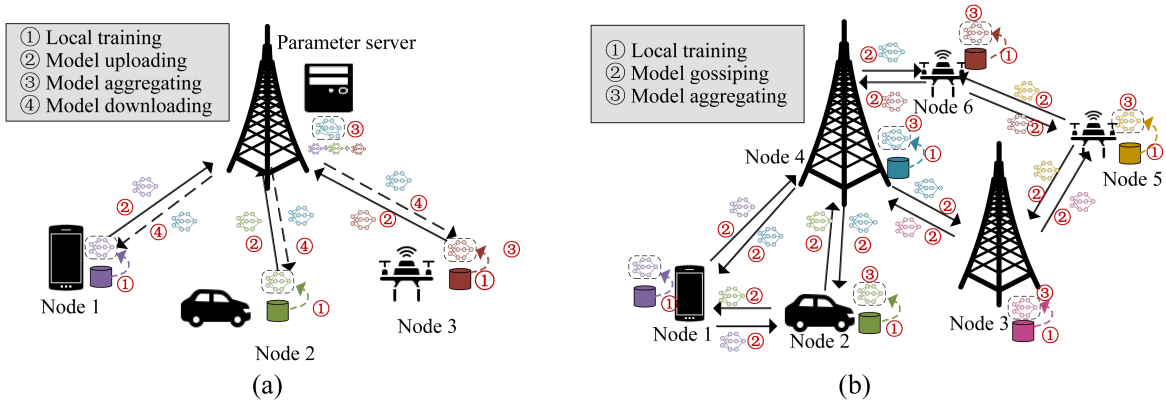


Fig. 1. An illustration of two federated learning architectures in collaborative MEC systems: (a) CFL and (b) DFL.

equipped with skewed data and uneven capabilities such as computational and communication capabilities. Specifically, non-independent identically distributed (Non-IID) data would make the conventional DFL algorithm inconsistent in update optimization directions of participating nodes and/or even fail in biased collaboration. Moreover, uneven capabilities of nodes might affect the learning efficiency of DFL or cause inactivation of some participating nodes, i.e., stragglers, especially greatly limiting the application in mMTC scenarios. The second challenge arises from the inherent dynamic nature of wireless networks, attributed to the mobility of nodes and the time-varying nature of wireless channels. In this scenario, conventional DFL algorithms [9], [10], [11] employing stationary collaboration strategies can become inflexible. Therefore, it necessitates the design of an adaptive collaborative strategy tailored to MEC environments and heterogeneous nodes, thereby enhancing the efficiency of the DFL collaborating process.

In this paper, we propose an adaptive decentralized federated learning (ADFL) framework that enables heterogeneous nodes to learn tailored collaboration strategies, thus maximizing the efficiency of the DFL training process in MEC systems. First, to tackle the challenge of heterogeneity, our ADFL framework introduces a general collaboration strategy model, which is capable of facilitating heterogeneous participating nodes to undertake diverse cooperative actions based on their respective conditions. Then, to effectively deal with the dynamic nature of the wireless MEC environments, we formulate the collaboration strategy learning problem as a stochastic game and solve it from a multi-agent reinforcement learning (MARL) perspective. This approach enables participating nodes to acquire an adaptive collaboration strategy by capturing the implicit dynamic features of the MEC systems. Lastly, we propose an efficient option critic-based collaboration strategy learning (OCSL) mechanism that enables individual nodes to learn their objective-specific adaptive collaboration strategies. The main contributions of this paper can be summarized as follows:

- We design a novel ADFL framework, enabling individual heterogeneous nodes to learn adaptive collaboration strategies within MEC systems. To the best of our knowledge, we are the first to propose the integration of state-of-the-art option-critic techniques to optimize the DFL training process, thus supporting MEC systems.
- We present an effective collaboration strategy model, which is generalized into two sub-strategies: the local training strategy and the resource scheduling strategy, aiming to improve the learning efficiency of the ADFL framework.
- We introduce an option framework in OCSL mechanism design to effectively tackle the inherent issues such as large-scale action space of collaboration strategy learning problem. Furthermore, we address the overestimation issue in OCSL mechanism by presenting the dual critic network-based approximation method. We further theoretically prove that the learned collaboration strategy can achieve the Nash equilibrium.
- We conduct extensive experiments on four real-world datasets to evaluate the performance of our proposed ADFL framework. The results demonstrate that ADFL outperforms state-of-the-art baselines in terms of model accuracy, efficiency, and communication cost.

The remainder of the paper is structured as follows. Section II reviews related works. Section III presents the proposed ADFL framework. In Section IV, we present the OCSL mechanism for improving the learning efficiency of ADFL framework. We perform extensive experiments based on real-world datasets in Section V. Finally, conclusions are drawn in Section VI.

II. Related Work

FL in MEC systems is receiving increasing attention due to its benefits in terms of communication overheads and user data privacy compared to traditional data-sharing-based collaborative learning. Existing works [2], [3], [4], [5], [6], [7], [8] investigated critical problems related to improving the efficiency of FL for the implementation in mobile networks. In [2], the authors focused on optimizing the FL process in wireless networks by jointly addressing device scheduling and bandwidth allocation to minimize time delay and energy consumption. Additionally, a line of works focused on reducing the communication overhead of FL in wireless systems through model compression methods. Specifically, the authors of [4] proposed a communication-efficient FedAvg named CE-FedAvg by

introducing a novel compression technique to greatly reduce the number of rounds to convergence. The authors of [5] introduced an efficient FL algorithm incorporating a weight-based proximal term. This approach enabled a limited number of nodes per round to engage in the training process through an unbiased sampling strategy. In [6], a flexible model compression scheme in FL was designed for MEC systems to save communication costs almost without compromising accuracy. Moreover, the authors of [7] proposed DetFed, a three-layer deterministic FL framework for ultra-reliable and low-latency scenarios. In [8], the authors proposed a multi-flow relay training framework for FL, to jointly reduce energy consumption and improve global model performance. However, it's important to note that in the aforementioned work, the designs of FL for wireless networks are based on a centralized architecture, where the central parameter server can become a bottleneck in the network and weaken its practical applicability.

To address the limitations of CFL, decentralized collaborative architectures have been extensively explored in the field of MEC [9], [10], [11]. In [9], the authors proposed the D-PSGD algorithm and provided the first theoretical analysis, indicating a regime in which DFL might outperform CFL for distributed stochastic gradient descent. Unlike CFL, where nodes communicate with a central parameter server, in DFL, nodes propagate models among their neighbors, where the efficiency of learning is determined by the model collaboration strategy. The allreduce algorithm, introduced in [10], provided conventional wisdom to collaboratively train the models. Additionally, the authors of [11] proposed a CDSGD algorithm to update models by exchanging model parameters with all neighbors in each iteration. However, it's important to note that the aforementioned DFL algorithms employ stationary collaboration training strategies, where the collaboration strategy remains the same at each round. These methods not only work inefficiently with heterogeneous nodes but also limit their adaptability in dynamic wireless environments. Recently, a few works [12], [13], [14], [15], [16] focused on collaborative strategies in wireless networks under fully DFL architecture. In [15], the authors proposed an adaptive exchanging interval algorithm based DFL in MEC system to optimize the trade-off between communication cost and training performance. The authors of [16] proposed a novel DFL for UAV Networks architecture, which enables FL within UAV networks without a central entity. However, these studies only focus on optimizing the selection of collaborating nodes and/or the construction of collaborative topology, overlooking the influence of wireless resource allocation on collaborative learning efficiency. Moreover, most existing DFL works [9], [10], [11], [12], [13], [14], [15], [16] require each node to perform local model training at each iteration step, which may not be cost-efficient for heterogeneous nodes. Therefore, in this paper, we focus on designing an adaptive DFL framework by jointly optimizing the training policy and resource scheduling policy to enhance training efficiency in collaborative MEC systems.

III. ADFL Framework

In this section, we present the proposed ADFL framework as depicted in Fig. 2, which supports the efficient DFL training process in collaborative MEC systems. Its OCSL mechanism enables heterogeneous nodes to learn tailored collaboration strategies by integrating the option-critic methods. At the function layer, our framework supports regression, classification, and clustering functions, which can be applied to a wide spectrum of application areas, such as industry 4.0, healthcare, smart city, autonomous vehicle, and Metaverse. In the following, we first introduce the collaborative MEC system with heterogeneous nodes. Then, we elaborate on the modified iteration processes of DFL training in detail. Finally, we present the proposed collaboration strategy model.

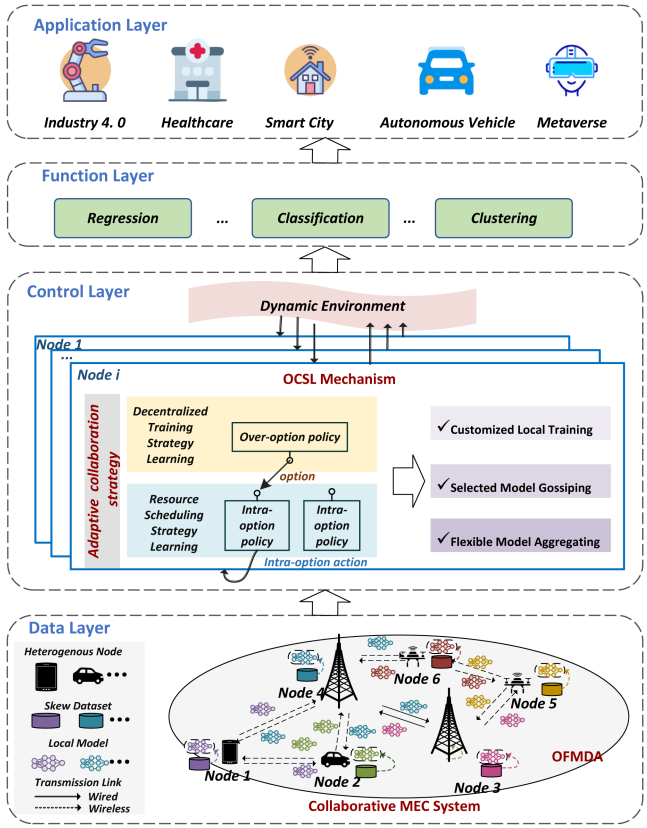


Fig. 2. Overview of the proposed ADFL framework.

A. Collaborative MEC System

We consider a decentralized MEC system as depicted in Fig. 2, where heterogeneous nodes such as multiple edge devices and base stations collaboratively train effective machine learning (ML) models to support AI applications. Let the undirected connected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ denote the topology of the collaborative MEC system, where the set of vertices is denoted as $\mathcal{V} = \{1, \ldots, i, \ldots, |\mathcal{V}|\}$, and the set of edges is represented as $\mathcal{E} = \{e_{i,j}\}_{i,j \in \mathcal{V}, j \neq i}$. The number of collaborating nodes is indicated by $|\mathcal{V}|$, where $i \in \mathcal{V}$ refers to the i-th collaborating node in the MEC system. The set of one-hop neighbors of collaborating node i can be represented as $\mathcal{N}_i = \{j \in \mathcal{V} | e_{i,j} = 1\}$. Moreover, we employ the widely-used Gauss-Markov random model [18], [19] for representing node mobility. Specifically, the velocity $v_i(k)$ and direction $\phi_i(k)$ of the node i in the k-th time slot can

be updated as:

$$v_i(k) = \kappa_1 v_i(k-1) + (1-\kappa_1)\bar{v}_i + \sqrt{1-\kappa_1^2}\phi_i, \quad (1)$$

$$\phi_i(k) = \kappa_2 \phi_i(k-1) + (1-\kappa_2)\bar{\phi}_i + \sqrt{1-\kappa_2^2}\Psi_i, \quad (2)$$

where $0 \leq \kappa_1, \kappa_2 \leq 1$, with κ_1 and κ_2 used to modulate the influence of the previous state. Here, $\bar{v}_i$ represents the average velocity of node i, and $\bar{\phi}_i$ denotes the average direction of node i. Additionally, Φ_i and Ψ_i are modeled as independent Gaussian random variables with respective mean-variance pairs $(\bar{\xi}_{v_i}, s^2_{v_i})$ and $(\bar{\xi}_{\Phi_i}, s^2_{\Phi_i})$, capturing the stochastic nature of the nodes' movements. In ADFL framework, we assume that collaborating nodes possess the identical type of local neural network model, denoted as $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_i, \ldots, \boldsymbol{\theta}_{|\mathcal{V}|}\}$. Here, $\boldsymbol{\theta}_i$ represents the model parameters of node i, and $|\boldsymbol{\theta}_i|$ indicates the parameter size. Let $f_i(\cdot)$ denote the loss function of node i. Note that, in the proposed ADFL framework, the local neural network model and loss function of nodes are tailored to specific collaborative learning tasks. For example, convolutional neural networks and cross-entropy loss functions are common for classification tasks, while recurrent neural networks and mean squared error are typically used for regression tasks. To achieve specific collaborative tasks, heterogeneous collaborating nodes continually enhance the accuracy of their local models by training their local datasets, exchanging parameters with neighboring nodes, and aggregating models in an iterative manner. In this paper, let k represent the iteration step of collaborating nodes. In real-world scenarios, different collaborating nodes exhibit heterogeneity, which can manifest in various perspectives, including variations in data distribution, communication capabilities, and computational capabilities. Therefore, in this paper, we consider a heterogeneous scenario in which each collaborating node $i \in \mathcal{V}$ possesses the following attributes:

- *Heterogeneous dataset*: Due to differences in the locations and environments of collaborating nodes, the local data distributions of these nodes are typically non-IID. The heterogeneous local data of collaborating node i is denoted as $\mathcal{D}_i$, where $|\mathcal{D}_i|$ represents the size of its dataset.
- *Heterogeneous computational capability*: The training of local models depends on the computational capabilities of the nodes, which may be equipped with CPUs and/or GPUs. Let z_i^{mem} and z_i^{core} indicate the memory frequency and the core frequency of the GPUs or CPUs in node i, respectively. As collaborating nodes may perform other computational tasks simultaneously, let $\delta_i(k)$ represent the current computational load of collaborating node i . Therefore, the available computational capability of collaborating node i at the k-th iteration is represented as:

$$C_i^{mem}(k) = (1-\delta_i(k)) z_i^{\text{mem}}, \quad (3)$$

$$C_i^{core}(k) = (1-\delta_i(k)) z_i^{\text{core}}. \quad (4)$$

  We assume that if a node trains a model during iteration k, it utilizes all available computational resources to minimize local training time.

- *Heterogeneous communication capability*: To exchange models with neighboring nodes, each collaborating node should be equipped with communication capabilities. There exist two transmission modes: wired communication and wireless communication. In this paper, wired communication is regarded as the primary way of communication between BSs. Moreover, the wireless communication resources of collaborating nodes without wired links (i.e., BS-device and device-to-device communication) include spectrum resource and transmit power. The number of available RBs of collaborating node i for model gossiping in the k-th iteration is denoted as $N_i(k)$, and the maximum transmit power is denoted by $P_i^{max}(k)$.

B. Modified Iteration Process

In the following, we will elaborate on the three modified steps of each iteration in the adaptive DFL training process, which is composed of local training, model gossiping, and model aggregation.

1) Customized Local Training: Existing works [9], [10], [11] on FL algorithms typically assume that collaborating nodes perform local training using their local datasets in each iteration. However, some collaborating nodes may be reluctant to train models based on their local datasets in specific iteration rounds for two main reasons: For one thing, local model training incurs training costs. Nodes with limited computing resources may prefer to obtain precise model parameters from neighboring nodes instead of performing local training. For another, as the number of local training iterations increases, local models will converge gradually to local optima, and the improvement in model performance becomes small. Therefore, in the later iterations of collaborative learning, local model training based on local datasets may become inefficient.

To address these issues, different from conventional FL algorithms [9], [10], [11], we introduce customized local training mode in FL that allows each collaborating node i to decide whether to perform local model training in each iteration k. Specifically, a binary indicator variable $\omega_{i,0}(k)$ is introduced to represent the local training policy. If collaborating node i conducts local model training in the k-th iteration, $\omega_{i,0}(k)$ is set to 1; Otherwise, it is set to 0. When node i decides to perform local training in the k-th iteration, collaborating node i first samples a batch of data from its local dataset denoted as $\{\xi_m^i\}_{m=1}^M$, where M is the batch size. Collaborating node i then calculates the gradient of its local model parameter $\boldsymbol{\theta}_i$ based on this batch of data, which can be represented as:

$$\triangle \boldsymbol{\theta}_i(k) = \sum_{m=1}^{M} \nabla f_i(\boldsymbol{\theta}_i; \xi_m^i) \quad \forall i \in \mathcal{V}, \quad (5)$$

where $\boldsymbol{\theta}_i$ and $f_i(\cdot)$ represent the parameters and loss function of the local ML model of node i, respectively. Then, each node i updates its local model using stochastic gradient descent, which is represented as:

$$\boldsymbol{\theta}_i\left(k+\frac{1}{2}\right) = \boldsymbol{\theta}_i(k) - \omega_{i,0}(k)\beta \triangle \boldsymbol{\theta}_i(k) \quad \forall i \in \mathcal{V}, \quad (6)$$

where β is the learning rate, and $\omega_{i,0}(k)$ represents the local training indicator of node i. The local model training time [8] for node i at iteration k can be expressed as

$$\tau_i^l(k) = T_i^0 + \frac{W_1}{C_i^{\text{mem}}} + \frac{W_2}{C_i^{\text{core}}}, \tag{7}$$

where T_i^0 is the static time consumption including data loading, data augmentation, etc; The workload cycles of memory access W_1 and arithmetic computation W_2 are measured for a batch of training data; C_i^{mem} and C_i^{core} indicate the available memory frequency and the core frequency of the GPUs or CPUs in node i, respectively.

2) Selected Model Gossiping: Most existing FL algorithms employ stationary strategies during the model gossiping phase. For instance, in D-PSGD [9], collaborating nodes randomly communicate with one of their neighbors in each iteration step, which might limit the potential of nodes with high communication capabilities. Similarly, in CDSGD [11], each collaborating node needs to obtain model parameters from all neighbors in every iteration, consuming significant communication resources and introducing substantial transmission delays. Therefore, in dynamic network environments, a selected model gossiping mode is crucial for improving collaboration efficiency. In contrast, we propose a selected model gossiping policy that allows each node i to decide whether to request model parameters from its neighboring node $j \in \mathcal{N}_i$ at each iteration k. Specifically, we introduce a set of binary indicator variables $\{\omega_{i,j}(k)\}_{j\in\mathcal{N}_i}$ to represented the gossiping policy, where $\omega_{i,j}(k) = 1$ indicates that collaborating node i requests model parameters from neighbor j in iteration k, otherwise $\omega_{i,j}(k) = 0$. Therefore, at k-th iteration, the set of neighbors that request model parameters of collaborating node i can be represented as:

$$\widehat{\mathcal{N}_i}(k) = \{j \in \mathcal{N}_i | \omega_{i,j}(k) = 1\}. \tag{8}$$

For wireless transmission nodes, we assume the system utilizes orthogonal frequency division multiple access technology. Furthermore, at k-th iteration, the set of neighbors that request the model parameters of collaborating node i through wireless transmission can be represented as $\widetilde{\mathcal{N}_i}(k)$. Note that each node uses unicast transmission to send model updates to neighboring nodes. The transmission rate of collaborating node j to collaborating node i can be expressed as:

$$V_{i,j}(k) = \omega_{i,j}(k) a_{j,i}(k) B^U \left(\log_2\left(1 + \frac{p_j(k) h_{i,j}(k)}{B^U N_0(k)}\right)\right), \tag{9}$$

where $\omega_{i,j}(k)$ is a binary variable; $a_{i,j}(k)$ represents the number of RBs of collaborating node $j \in \widetilde{\mathcal{N}_i}(k)$ using to transmit model parameters to node i; B^U is the unit bandwidth of the RB; $p_j(k) = \frac{P_j^{max}(k)}{\sum_{m\in\widetilde{\mathcal{N}_j}(k)} a_{j,m}(k)}$ is the transmit power of node j; $a_{j,m}(k)$ indicates the number of RB using by node i to transmit model parameters to neighbor $j \in \widetilde{\mathcal{N}_i}$ at the k-th iteration; $h_{i,j}(k)$ is the channel gain between nodes i and j; N_0 denotes the noise power spectral density. Similar to that in [12], [13], [14], [15], [16], we assume that no interference exists among nodes, since orthogonal frequency resources are allocated to participating nodes, even if they have overlapped coverage area.

For wired transmission nodes, we consider a fixed delay between nodes, denoted as $\tau_{i,j}^f$ [20]. The transmission delay from node j to node i is expressed as:

$$\tau_{i,j}^c(k) = \begin{cases} \frac{\omega_{i,j}(k)|\boldsymbol{\theta}_i|}{V_{i,j}(k)} & \text{Wireless transmission} \\ \tau_{i,j}^f & \text{Wired transmission.} \end{cases} \tag{10}$$

3) Flexible Model Aggregating: In the model aggregation phase, each collaborating node aggregates and updates its local model based on its own model and the received models. Different from traditional model aggregating policy, i.e Fedavg [21], we introduce an adaptive model aggregating policy, which can be represented as:

$$\boldsymbol{\theta}_i(k+1) = \rho_i(k)\boldsymbol{\theta}_i\left(k+\frac{1}{2}\right) + \sum_{j\in\mathcal{N}_i} \frac{(1-\rho_i(k))\omega_{i,j}(k)}{\sum_{j\in\mathcal{N}_i}\omega_{i,j}(k)} * \boldsymbol{\theta}_j\left(k+\frac{1}{2}\right) \quad \forall i \in \mathcal{V}, \tag{11}$$

where $\rho_i \in (0,1)$ represents the average rate for model aggregation. The rationale behind the decentralized model aggregating rule lies in its ability to optimize the objective function of learning tasks by incorporating both the gradient information from its own data and the weighted parameters received from its neighboring nodes. The average rate of model aggregation plays a crucial role in balancing the contributions from local and neighboring models during the training process. Specifically, ρ_i can be expressed as:

$$\rho_i(k) = \frac{\delta_{acc}^{i,i}(k)}{\delta_{acc}^{i,i}(k) + \sum_{j\in\mathcal{N}_i}\omega_{i,j}(k)\delta_{acc}^{i,j}(k)}, \tag{12}$$

where $\delta_{acc}^{i,i}(k)$ indicates the improvement in testing accuracy of the local model on node i's dataset at iteration k compared to the previous round, and $\delta_{acc}^{i,j}(k)$ measures the accuracy improvement of the model trained by neighbor j on node i's dataset at iteration k compared to the previous round. This formulation ensures that $\rho_i(k)$ dynamically adjusts based on the relative accuracy improvements, making the aggregation process adaptive. It prioritizes models that have shown significant contributions to accuracy improvement, thereby enhancing the overall learning efficiency and effectiveness of the collaborative training process. Note that since the computational requirements for model aggregation are relatively small, the computational delay introduced by model aggregation can be neglected compared to the time spent on model training and model transmission.

C. Adaptive Collaboration Strategy

Due to the dynamic nature and the heterogeneity of nodes in MEC systems, an adaptive collaboration strategy is crucial for enhancing the efficiency of collaborative learning. Therefore, we introduce an adaptive collaboration strategy $\boldsymbol{\pi} = (\boldsymbol{\pi_1}, \ldots, \boldsymbol{\pi_i}, \ldots, \boldsymbol{\pi}_{|\mathcal{V}|})$ with $\boldsymbol{\pi_i} = \{\pi_i^\Omega, \pi_i^\omega\}$ for each node, which encompasses two sub-strategies: decentralized training

strategy π_i^{Ω} and resource scheduling strategy π_i^{ω}. In the following, we will introduce the two sub-strategies in detail.

Definition 1: Decentralized Training Strategy: The decentralized training strategy $\pi_i^{\Omega}(k)$ for collaborating node i at the k-th iteration represents the probability distribution of training decision $\boldsymbol{\omega}_i(k)$:

$$\boldsymbol{\omega}_i(k) \sim \pi_i^{\Omega} = [\omega_{i,0}(k), \omega_{i,1}(k), \ldots, \omega_{i,j}(k), \ldots, \omega_{i,|\mathcal{N}_i|}(k)], \tag{13}$$

where $\omega_{i,0}(k)$ represents the local training indicator. If node i performs local model training at iteration k, then $\omega_{i,0}(k) = 1$; otherwise, $\omega_{i,0}(k) = 0$. $\omega_{i,j}(k)$ is the model parameter request indicator. If node i requests model parameters from its neighbor $j \in \mathcal{N}_i$ at iteration k, then $\omega_{i,j}(k) = 1$; Otherwise, $\omega_{i,j}(k) = 0$.

Definition 2: Resource Scheduling Strategy: The training strategy π_i^{ω} for collaborating node i represents the probability distribution of scheduling decision $\boldsymbol{a}_i(k)$ at the k-th iteration under $\boldsymbol{\omega}_i(k)$:

$$\boldsymbol{a}_i(k) \sim \pi_i^{\omega} = [a_{i,j}(k)]_{j \in \widetilde{\mathcal{N}_i}(k)}, \tag{14}$$

where $a_{i,j}(k)$ represents the number of RBs using by node i to transmit model parameters to neighbor $j \in \widetilde{\mathcal{N}_i}$ at the k-th iteration.

The inherent heterogeneity of the collaborating nodes and dynamic environments of MEC systems poses a challenge for conventional FL methods, particularly in terms of optimizing learning efficiency. Therefore, in our proposed ADFL framework, we aim to improve the learning efficiency of each collaborating node by learning an optimal adaptive collaboration strategy. Specifically, the learning efficiency of distributed collaborative learning in MEC systems is defined as follows:

$$E_i(k) = \frac{acc_i(\boldsymbol{\theta}_i(k)) - acc_i(\boldsymbol{\theta}_i(k-1))}{\tau_i(k)}, \tag{15}$$

where $E_i(k)$ represents the learning efficiency of collaborating node i at iteration k; $acc_i(\boldsymbol{\theta}_i(k))$ denotes the accuracy of the model at iteration k; $acc_i(\boldsymbol{\theta}_i(k)) - acc_i(\boldsymbol{\theta}_i(k-1))$ represents the change in model accuracy for node i after iteration k; $\tau_i(k)$ denotes the time required for collaborating node i at iteration k. The exact overhead for collaborative learning mainly consists of local model training time, model transmission time, and model aggregation time, where the model aggregation time is negligible compared to the other two. Therefore, the time for collaborating node i at iteration k is represented as:

$$\tau_i(k) = \tau_i^l(k) + \max\{\tau_{i,0}^c(k), \ldots, \tau_{i,j}^c(k), \ldots, \tau_{i,|\widetilde{\mathcal{N}_i}(k)|}^c(k)\}. \tag{16}$$

Due to the intricate relationships between collaborating nodes, an effective collaboration strategy is crucial for the convergence of the collaborative training process. Therefore, the objective of ADFL framework is to improve the long-term collaborative learning efficiency of collaborating nodes by learning an optimal collaboration strategy $\boldsymbol{\pi}^*$. Then, the problem of node i can be mathematically formulated as

$$\max \quad \mathbb{E}_{\boldsymbol{\pi}_i|\boldsymbol{\pi}_{-i}}\left[\lim_{k\to\infty}\frac{1}{K}\sum\nolimits_{k=1}^{K}\gamma^k E_i(k)\right] \tag{17}$$

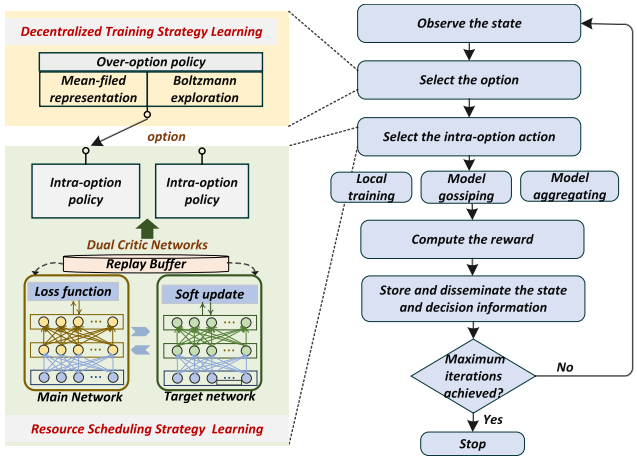


Fig. 3. Overview of the proposed OCSL mechanism.

$$\text{s.t. } \omega_{i,j}(k) \in \{0,1\}, \forall i \in \{0\} \cup \mathcal{V}, j \in \mathcal{N}_i(k), \forall k \tag{17.1}$$

$$a_{i,j}(k) \in \mathbb{N}, \forall i \in \mathcal{V}, j \in \widetilde{\mathcal{N}_i}(k), \forall k \tag{17.2}$$

$$\sum\nolimits_{j \in \widetilde{\mathcal{N}_i}(k)} a_{i,j}(k) \leq N_i(k), \forall i \in \mathcal{V}, \forall k \tag{17.3}$$

In objective function (17), γ is a discount factor; $\boldsymbol{\pi_i}$ represents the stochastic collaboration strategy of node i; $\boldsymbol{\pi_{-i}}$ denotes the joint collaboration strategy of all collaborating nodes except node i. Constraint (17.1) shows the binary nature of the training decisions. Constraint (17.2) specifies that the variable $a_{i,j}^s(k)$ is an integer. In constraint (17.3), the maximal number of RB $N_i(k)$ of node i is restricted. Motivated by game theoretical modeling of multi-agent systems and the stochastic nature of collaboration strategy (see Definitions 1 and 2), the problem (17) is a typical Markov game. The effectiveness of reinforcement learning has been recognized in recent studies focused on addressing Markov games across diverse domains [22]. However, it can be observed that the actions of the problem (17) have an extremely large number of possibilities, characterized by $2^{|\mathcal{N}_i|} \times E_i^{|\widetilde{\mathcal{N}_i}|}$. The extremely high action space poses a significant challenge for traditional multi-agent reinforcement learning (MARL) methods that especially struggle with problems featuring large action spaces due to heightened exploration and exploitation complexities. To address these challenges, an efficient mechanism should be designed to reduce the extremely high dimensionality of actions while ensuring the convergence of the collaboration strategy.

IV. Proposed Methods

In this section, we propose an efficient option critic-based collaboration strategy learning (OCSL) mechanism for improving the training efficiency of the ADFL framework, as illustrated in Fig. 3. The emerging option-critic framework [23] can facilitate a more structured approach in overcoming the inefficiencies of traditional reinforcement learning in scenarios with high-dimensional action spaces. For instance, in a complex robotic control task, traditional reinforcement learning algorithms often struggle due to the vast number of possible actions at each step.

By leveraging the option-critic framework, the problem can be broken down into a hierarchy of sub-tasks or options, which not only reduces the complexity of the action space but also accelerates learning by enabling the reuse of learned options across different tasks. In the following, we first introduce the modeling of the problem (17) as a Markov game within the option framework. We then elaborated on the two core learning modules in the proposed OCSL mechanism followed by the analysis of the proposed mechanism.

A. Markov Game With Options

Markov game problem with the options can be defined by the tuple $\{\mathcal{V}, \mathcal{S}, \{\Omega_i\}_{i\in\mathcal{V}}, \{\mathcal{A}_i\}_{i\in\mathcal{V}}, \mathcal{P}, \{R_i\}_{i\in\mathcal{V}}, \gamma\}$, where $\mathcal{V}$ represents the set of $|\mathcal{V}|$ collaborating nodes; $\mathcal{S}$ represents the state space observed by collaborating nodes; Ω_i indicates the temporally extended actions space (as known as option space) for collaborating node i; $\mathcal{A}_i$ represents the intra-option action space for collaborating node i; $\mathcal{P}$ represents the transition probability from a state $\boldsymbol{s}$ in $\mathcal{S}$ to any state $\boldsymbol{s'}$ in $\mathcal{S}'$ through options and actions; R_i is the immediate reward received by collaborating node i after executing action; and $\gamma \in (0, 1]$ is the discount factor. Detailed designs are elaborated in the following.

State: The state space $\mathcal{S}_i$ describes the state of the intelligent agent , which is represented as:

$$\boldsymbol{s}(k) = [acc_i(k), C_i^{mem}(k), C_i^{core}(k), N_i(k), P_i^{max}(k)]_{i\in\mathcal{V}}, \tag{18}$$

where $acc_i(k)$ is the accuracy of node i at iteration step k; $C_i^{mem}(k)$ and C_i^{core} denotes the computational capability of node i; $N_i(k)$ indicates the available number of RBs; $P_i^{max}(k)$ is the maximum transmit power of node i at iteration k.

Option: The options framework provides a paradigm for a reinforcement learning agent to systematically solve complex problems in a hierarchical way by introducing temporally extended actions. The temporally extended action $\boldsymbol{\omega}_i \in \Omega_i$ of node i, also known as Markov option can be represented as the decentralized training decision of node i (see Definition 1). The option can initiate at a state $\boldsymbol{s}$ that is part of its initiation set $\boldsymbol{s} \in I_{\omega_i}$. Note that all options of nodes can be selected at any state s in our considered system. The collaborating node i selects an option according to the over-option policy π_i^{Ω}. Let π_{-i}^{Ω} denote the joint over-option policy of all nodes except node i.

Intra-Option Action: When an option $\boldsymbol{\omega}_i$ is chosen, a series of intra-option actions $\boldsymbol{a}_i$ will be executed based on the intra-option policy π_i^{ω}. Let π_{-i}^{ω} denote the joint intra-option actions policy of all nodes except node i. Additionally, at iteration step k, the intra-option action $\boldsymbol{a}_i(k) \in \mathcal{A}_i$ of collaborating node i is expressed as the resource scheduling decision of node i (see Definition 2).

Transition Probability: After executing the action, the state $\boldsymbol{s}(k)$ transitions to the next state $\boldsymbol{s}(k+1)$, where the transition probability p satisfies:

$$\sum\nolimits_{\boldsymbol{s}(k+1)\in\mathcal{S}} p(\boldsymbol{s}(k+1)|\boldsymbol{s}(k), \boldsymbol{\omega}_1(k) \times \boldsymbol{a}_1(k), \ldots, \boldsymbol{\omega}_{|\mathcal{V}|}(k) \times \boldsymbol{a}_{|\mathcal{V}|}(k)) = 1. \tag{19}$$

Reward: As feedback for taking a specific action, collaborating node i will receives an immediate reward $R_i(k)$, which is defined as:

$$R_i(k) = E_i(k) = \frac{acc_i(\boldsymbol{\theta}_i(k)) - acc_i(\boldsymbol{\theta}_i(k-1))}{\tau_i(k)}. \tag{20}$$

By examining, we can observe that the options and the inter-option actions within the Markov option-based framework can be transferred to the decentralized training decisions and resource allocation decisions in our problem. Additionally, the over-option policy and intra-option policy in the Markov option-based framework correspond to the sub-strategies of the collaborative strategy (see Definitions 1 and 2), i.e., decentralized training strategy π_i^{Ω} and resource scheduling strategy π_i^{ω}. The decomposition provided by this option framework effectively transforms the originally vast-action-space reinforcement learning problem into a sequential learning problem involving two sub-policies. Therefore, each collaborating node aims to maximize the long-term learning efficiency of collaborative learning. Next, we will introduce the two strategy learning methods design.

B. Decentralized Training Strategy Learning

We focus on the decentralized training strategy learning for collaborating nodes in this subsection. We first introduce the state-option value function $Q_i^{\Omega}(\boldsymbol{s}, \boldsymbol{\omega})$, which is used to evaluate how good the option $\boldsymbol{\omega}_i$ is under a specific state $\boldsymbol{s}$ with collaboration policy. Therefore, it can be expressed as

$$Q_i^{\Omega}(\boldsymbol{s}, \boldsymbol{\omega}) = \mathbb{E}_{\pi_i^{\Omega}, \pi_{-i}^{\Omega}, \pi_i^{\omega}, \pi_{-i}^{\omega}} \left[\sum\nolimits_{k=1}^{\infty} \gamma^k R_i(k) \right], \tag{21}$$

where the state-option value function is the discounted return expected over all the trajectories starting at state $\boldsymbol{s}$ and option $\boldsymbol{\omega}_i$; $\boldsymbol{\omega} = [\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_i, \ldots, \boldsymbol{\omega}_{|\mathcal{V}|}]$ denotes the joint options of nodes. We can observe that the dimension of joint options in the standard state-option value function in (21) is extremely high, which is 2^W with $W = \sum_{i=1}^{|\mathcal{V}|}(1 + |\mathcal{N}_i|)$. we first reduce the dimension of joint options in the standard state-option value function by factorizing the standard state-option value of node i into the sum of pairwise local interactions:

$$Q_i^{\Omega}(\boldsymbol{s}, \boldsymbol{\omega}) = \frac{1}{|\mathcal{N}(i)|} \sum_{j\in\mathcal{N}(i)} Q_i^{\Omega}(\boldsymbol{s}, \omega_i, \omega_j), \tag{22}$$

where $Q_i^{\Omega}(\boldsymbol{s}, \omega_i, \omega_j)$ is the pairwise local state-option value function for the collaborating node i and its neighbor $j \in \mathcal{N}(i)$. The decomposition of pairwise local state-option value functions has been shown to significantly reduce the complexity of interactions between agents and the dimensionality of joint actions, while implicitly preserving global interactions [24]. However, decomposing pairwise state-option value functions still requires interaction between each pair of collaborating nodes, limiting the scalability of collaborative learning in MEC systems. Therefore, we further adopt mean-field theory [25] to approximately characterize the state-option value function. The definition of the mean-field state-option value function based on mean-field theory is given below.

Definition 3: The mean-field state-option value function $\bar{Q}_i^{\Omega}(\boldsymbol{s}, \boldsymbol{\omega}_i, \bar{\boldsymbol{\omega}}_i)$ is approximated by computing the average option

of the neighbors of node i, given by $\bar{\boldsymbol{\omega}}_i = \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)}\hat{\boldsymbol{\omega}}_j$, where $\hat{\boldsymbol{\omega}}_j$ is the one-hot encoded option $\boldsymbol{\omega}_j$ of collaborating node j.

When the pairwise local state-option value functions are M-smooth, the standard state-option value function $Q_i^{\Omega}(\boldsymbol{s},\boldsymbol{\omega})$ can be represented by the mean-field state-option value function and a bounded value $b \in [-2M, 2M]$ [26]:

$$Q_i^{\Omega}(\boldsymbol{s},\boldsymbol{\omega}) = \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)} Q_i^{\Omega}(\boldsymbol{s},\omega_i,\omega_j) = \bar{Q}_i^{\Omega}(\boldsymbol{s},\boldsymbol{\omega}_i,\bar{\boldsymbol{\omega}}_i) + b \approx \bar{Q}_i^{\Omega}(\boldsymbol{s},\boldsymbol{\omega}_i,\bar{\boldsymbol{\omega}}_i). \quad (23)$$

Therefore, by employing the mean-field state-option value function to approximate the standard state-option value function, the dimensionality of joint actions can be significantly reduced. According to Bellman's equation, the cumulative discounted mean-field state-option value function $\bar{Q}_i^{\Omega}(\boldsymbol{s}(k),\boldsymbol{\omega}_i(k),\bar{\boldsymbol{\omega}}_i(k))$ at iteration k can be represented as (24) shown at the bottom of this page , where the last term of the equation involves the value of executing an action in the context of a state-option pair, namely Q-function Q_i^{ω}. The definition of the Q-function Q_i^{ω} will be discussed in the Section IV-C. With the state-option value function, the objective-specific decentralized training strategy of node i can adopt the widely used Boltzmann exploration policy:

$$\pi_i^{\boldsymbol{\Omega}}(\boldsymbol{\omega}_i|\boldsymbol{s},\bar{\boldsymbol{\omega}}_i) = \frac{\exp(\xi\bar{Q}_i^{\Omega}(\boldsymbol{s},\boldsymbol{\omega}_i,\bar{\boldsymbol{\omega}}_i)}{\sum_{\boldsymbol{\omega}'_i\in\Omega_i}\exp(\xi\bar{Q}_i^{\Omega}(\boldsymbol{s},\boldsymbol{\omega}_i,\bar{\boldsymbol{\omega}}_i))}, \quad (25)$$

where ξ is the temperature parameter of the policy.

C. Resource Scheduling Strategy Learning

After the execution of an option based on the decentralized training strategy, the node further needs to determine how to schedule resources. Therefore, we introduce the Q-function to evaluate the goodness of choosing a particular intra-option action under a specific state-option pair. The definition of Q-function for node i is expressed as follows:

$$Q_i^{\omega}(\boldsymbol{s}(k),\boldsymbol{\omega}(k),\boldsymbol{a}(k)) = R_i(\boldsymbol{s}(k),\boldsymbol{\omega}(k),\boldsymbol{a}(k)) + \gamma * \sum_{\boldsymbol{a}_i}[\pi_i^{\omega}(\boldsymbol{a}_i|\boldsymbol{s}(k),\bar{\boldsymbol{a}}_i(k))Q_i^{\omega}(\boldsymbol{s}(k+1),\boldsymbol{\omega}(k+1),\boldsymbol{a}(k+1))], \quad (26)$$

where the $\boldsymbol{a} = [\boldsymbol{a}_1,\ldots,\boldsymbol{a}_i,\ldots,\boldsymbol{a}_{|\mathcal{V}|}]$ denotes the joint over-option actions of all nodes. Furthermore, we can also employ the mean field method to compress the action space of the standard Q-function. Similarly, the definition of the mean-field Q-function is given below.

Definition 4: The mean-field Q-function $\bar{Q}_i^{\omega}(\boldsymbol{s},\boldsymbol{\omega}_i,\bar{\boldsymbol{\omega}}_i,\boldsymbol{a}_i,\bar{\boldsymbol{a}}_i)$ is approximated by computing the average intra-option actions of the neighbors of collaborating node i, given by $\bar{\boldsymbol{a}}_i = \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)}\hat{\boldsymbol{a}}_j$, where $\hat{\boldsymbol{a}}_j$ denotes the one-hot encoded intra-option action $\boldsymbol{a}_j$ of collaborating node j.

Furthermore, based on bellman equation, the mean-flied Q-function be represented as:

$$\bar{Q}_i^{\omega}(\boldsymbol{s}(k),\boldsymbol{\omega}_i(k),\bar{\boldsymbol{\omega}}_i(k),\boldsymbol{a}_i(k),\bar{\boldsymbol{a}}_i(k)) = R_i(\boldsymbol{s}(k),\boldsymbol{\omega}(k),\boldsymbol{a}(k)) + \gamma\sum_{\boldsymbol{a}_i}[\pi_i^{\omega}(\boldsymbol{a}_i|\boldsymbol{s}(k),\bar{\boldsymbol{a}}_i(k)) * \bar{Q}_i^{\omega}(\boldsymbol{s}(k+1),\boldsymbol{\omega}_i(k+1), \bar{\boldsymbol{\omega}}_i(k+1),\boldsymbol{a}_i(k+1),\bar{\boldsymbol{a}}_i(k+1))], \quad (27)$$

where the second term in the equation involves the value of the Q-function from the next time iteration. In conventional Q-learning methods, a Q-table should be constructed to store all possible Q-function $\bar{Q}_i^{\omega}(\boldsymbol{s},\boldsymbol{\omega}_i,\bar{\boldsymbol{\omega}}_i,\boldsymbol{a}_i,\bar{\boldsymbol{a}}_i)$ according to (27). However, constructing Q-table becomes inefficient due to the high-dimensional action space. Therefore, we adopt differentiable parameterized function approximators to approximate the Q-function. Specifically, we employ the nonlinear approximation method by designing a multi-layer neural network. Let $\widehat{Q}_i^{\omega}(\boldsymbol{s},\boldsymbol{\omega}_i,\bar{\boldsymbol{\omega}}_i,\boldsymbol{a}_i,\bar{\boldsymbol{a}}_i;\vartheta_i)$ denote the parameterized Q function weighted by ϑ_i. However, there are insufficient experience samples at the beginning of the training process, which will lead to an overestimation of the state-option value function and degrade the quality of the decentralized training strategy. Therefore, we design dual critic networks to tackle the overestimation issue, as depicted in Fig. 3. In the dual critic networks, the parameterized Q-function network $\widehat{Q}_i^{\omega}(\boldsymbol{s},\boldsymbol{\omega}_i,\bar{\boldsymbol{\omega}}_i,\boldsymbol{a}_i,\bar{\boldsymbol{a}}_i;\vartheta_i)$ and parameterized target network $\widehat{Q}_i'^{\omega}(\boldsymbol{s},\boldsymbol{\omega}_i,\bar{\boldsymbol{\omega}}_i,\boldsymbol{a}_i,\bar{\boldsymbol{a}}_i;\vartheta_i')$ should be trained, where ϑ_i and ϑ_i' are the two networks' parameters, respectively. The replay buffer in the dual critic networks is used to store the experiences $< \boldsymbol{s}(k),\boldsymbol{\omega}_i(k),\bar{\boldsymbol{\omega}}_i(k),\boldsymbol{a}_i(k),\bar{\boldsymbol{a}}_i(k),R_i(k),\boldsymbol{s}(k+1) >$ for network training. Specifically, node i samples a mini-batch $\{\epsilon_i^b\}_{b=1}^{B}$ of experiments from the replay buffer, where B is the batch size. The parameterized Q-network can be trained by minimizing the TD error. Therefore, the loss function of $\widehat{Q}_i^{\omega}(\boldsymbol{s},\boldsymbol{\omega}_i,\bar{\boldsymbol{\omega}}_i,\boldsymbol{a}_i,\bar{\boldsymbol{a}}_i;\vartheta_i)$ of collaborating node i can be expressed as:

$$\mathcal{L}_i(\vartheta_i) = \frac{1}{B}\sum_{\epsilon_i^b}(R_i(\boldsymbol{s}(k),\boldsymbol{\omega}(k),\boldsymbol{a}(k)) + \gamma\sum_{\boldsymbol{a}_i}[\pi_i^{\boldsymbol{\omega}}(\boldsymbol{a}_i|\boldsymbol{s}(k+1), \bar{\boldsymbol{a}}_i(k+1)) * \widehat{Q}_i'^{\omega}(\boldsymbol{s}(k+1),\boldsymbol{\omega}_i(k+1),\bar{\boldsymbol{\omega}}_i(k+1),\boldsymbol{a}_i(k+1),$$

$$\bar{Q}_i^{\Omega}(\boldsymbol{s}(k),\boldsymbol{\omega}_i(k),\bar{\boldsymbol{\omega}}_i(k)) = \sum_{\boldsymbol{a}_i}[\pi_i^{\omega}(\boldsymbol{a}_i|\boldsymbol{s},\bar{\boldsymbol{a}}_i)R_i(\boldsymbol{s}(k),\boldsymbol{\omega}_i(k),\boldsymbol{a}_i)] + \gamma\sum_{\omega_i}[\pi_i^{\Omega}(\omega_i|\boldsymbol{s}(k+1),\bar{\boldsymbol{\omega}}_i(k+1))*$$

$$Q_i^{\Omega}(\boldsymbol{s}(k+1),\boldsymbol{\omega}_i(k+1),\bar{\boldsymbol{\omega}}_i(k+1))] = \sum_{\boldsymbol{a}_i}[\pi_i^{\omega}(\boldsymbol{a}_i|\boldsymbol{s}(k),\bar{\boldsymbol{a}}_i(k))R_i(\boldsymbol{s}(k),\boldsymbol{\omega}_i(k),\boldsymbol{\omega}_{-i}(k),\boldsymbol{a}_i(k),\boldsymbol{a}_{-i}(k),)]$$

$$+\gamma\sum_{\omega_i,\boldsymbol{a}_i}[\pi_i^{\Omega}(\omega_i|\boldsymbol{s}(k+1),\bar{\boldsymbol{\omega}}_i(k+1))\pi_i^{\omega}(\boldsymbol{a}_i|\boldsymbol{s}(k+1),\bar{\boldsymbol{a}}_i(k+1)) * Q_i^{\omega}(\boldsymbol{s}(k+1),\boldsymbol{\omega}_i(k+1),\boldsymbol{a}_i)], \quad (24)$$

$$\bar{\boldsymbol{a}}_i(k+1);\vartheta_i')] - \widehat{Q}_i^{\omega}(\boldsymbol{s}(k),\boldsymbol{\omega}_i(k),\bar{\boldsymbol{\omega}}_i(k),\boldsymbol{a}_i(k),\bar{\boldsymbol{a}}_i(k);\vartheta_i))^2 \tag{28}$$

Then, the update rule of the parameterized Q-function can be expressed as

$$\vartheta_i = \vartheta_i - \eta(k)\nabla\mathcal{L}_i(\vartheta_i), \tag{29}$$

where the $\eta(k)$ is the learning rate of the parameterized Q-function. We employ the soft update methods to periodically update the target Q-function network, which can be expressed as

$$\vartheta_i' = \alpha\vartheta_i + (1-\alpha)\vartheta_i, \tag{30}$$

where α is the soft update factor of the target network. With two independent estimation networks, the proposed dual critic networks can achieve an unbiased estimation of the Q-function by selecting actions with the opposite network, which helps mitigate the overestimation problem.

Based on the approximation of Q-function, we further provide the construction method of the resource scheduling strategy. Similarly, since the policy is discrete, we adopt the Boltzmann exploration policy to construct the resource scheduling strategy, which can be expressed as:

$$\pi_i^{\boldsymbol{\omega}}(\boldsymbol{a}_i|\boldsymbol{s},\bar{\boldsymbol{a}}_i) = \frac{\exp(\xi\widehat{Q}_i^{\omega}(\boldsymbol{s},\boldsymbol{\omega}_i,\bar{\boldsymbol{\omega}}_i,\boldsymbol{a}_i,\bar{\boldsymbol{a}}_i;\vartheta_i))}{\sum_{\boldsymbol{a}_i'\in\mathcal{A}_i}\exp(\xi\widehat{Q}_i^{\omega}(\boldsymbol{s},\boldsymbol{\omega}_i,\bar{\boldsymbol{\omega}}_i,\boldsymbol{a}_i,\bar{\boldsymbol{a}}_i;\vartheta_i))}, \tag{31}$$

where ξ is the temperature of the policy.

D. Algorithm of OCSL Mechanism

The details of the proposed OCSL mechanism for node i are elucidated in Algorithm 1. First, node i initializes critical parameters including the local DFL model parameters θ_i, parameters of the dual critic networks $\boldsymbol{\vartheta}_i'$, mean option $\bar{\boldsymbol{\omega}}_i$, and mean intra-option action $\bar{\boldsymbol{a}}_i$. Subsequently, in each iteration, node i observes the current state through (18) (line 3) and gets the option $\bar{\boldsymbol{\omega}}_i$ and intra-option action $\bar{\boldsymbol{a}}_i$ according to the two sub-strategies of adaptive collaborative strategy respectively (lines 4-5). Executing in accordance with the collaborative strategy, the node undertakes the local training phase (lines 6-8). Notably, deviating from conventional DFL algorithms where local training transpires at every iteration, node i opts for training the local model solely when $\omega_i^0(k)=1$. Following this, each node i embarks on the gossiping phase (lines 9-12), pulling the model $\boldsymbol{\theta}^j(k+\frac{1}{2})$ from its neighbor $j\in\mathcal{N}(i)$ if $\omega_{i,j}(k)=1$ in the collaboration policy. Similarly, it pushes its model $\boldsymbol{\theta}^i(k+\frac{1}{2})$ to the neighbor $j\in\mathcal{N}(i)$ if $\omega_{j,i}(k)=1$. Then, the node aggregates the pulled model parameters employing a specified formula in the averaging phase (line 13). The node then proceeds to calculate the reward based on (20) (lines 14). Note that all nodes should periodically exchange state, option and intra-option action information with their neighbors through dedicated signaling messages, which ensures that each node has up-to-date information. Following this, node i observes the state of the next step, computes the mean option and intra-option action, and archives the experience data in the replay buffer (lines 15-18). Finally, the node undertakes the training of the dual critic networks (lines 19-23). This iterative process persists until convergence or until reaching the maximum number of iteration steps K.

Algorithm 1: OCSL Mechanism (Node i).

Input: $\mathcal{G}$, $\mathcal{D}_i$, E_i, P_i^{max}, γ, B, η, α, ξ, K.
Output: collaborative strategy $\boldsymbol{\pi_i} = \{\pi_i^{\Omega}, \pi_i^{\omega}\}$

1: Initialize $\boldsymbol{\theta}_i$, $\boldsymbol{\vartheta}_i$, $\boldsymbol{\vartheta}_i'$, $\bar{\boldsymbol{\omega}}_i$, $\bar{\boldsymbol{a}}_i$
2: **for** $k=1,2,\ldots,K$ **do**
3: Observe the current state $\boldsymbol{s}(k)$ by (18)
4: Get the option $\boldsymbol{\omega}_i(k)$ according to decentralized training strategy π_i^{Ω} in (25)
5: Get the intra-option action $\boldsymbol{\omega}_i(k)$ according to resource scheduling strategy π_i^{ω} in (31)
6: **if** $\omega_{i,0}(k)=1$ **then**
7: Perform local model training $\boldsymbol{\theta}_i(k+\frac{1}{2})$ by (6)
8: **end if**
9: **for** $j\in\mathcal{N}(i)$ **do**
10: Pull model $\boldsymbol{\theta}_j(k+\frac{1}{2})$ from neighbor j if $\omega_{i,j}(k)=1$
11: Push model $\boldsymbol{\theta}_i(k+\frac{1}{2})$ to neighbor j if $\omega_{j,i}(k)=1$
12: **end for**
13: Average the model $\boldsymbol{\theta}_i(k+1)$ by (11)
14: $R_i(k) \leftarrow$ Calculate the reward by (20)
15: Observe the next state $\boldsymbol{s}(k+1)$
16: Compute mean option action $\bar{\boldsymbol{\omega}}_i(k)$ by Definition (3)
17: Compute mean intra-option action $\bar{\boldsymbol{a}}_i(k)$ by Definition (4)
18: Store experiences in buffer $<\boldsymbol{s}(k),\boldsymbol{\omega}_i(k),\bar{\boldsymbol{\omega}}_i(k),\boldsymbol{a}_i(k),\bar{\boldsymbol{a}}_i(k),R_i(k),\boldsymbol{s}(k+1)>$
19: **if** $k\geq B$ **then**
20: Sample B experiences from the replay buffer
21: Train $\widehat{Q}_i^{\omega}(\boldsymbol{s},\boldsymbol{\omega}_i,\bar{\boldsymbol{\omega}}_i,\boldsymbol{a}_i,\bar{\boldsymbol{a}}_i;\vartheta_i)$ by minimizing (28)
22: Train $\widehat{Q}_i'^{\omega}(\boldsymbol{s},\boldsymbol{\omega}_i,\bar{\boldsymbol{\omega}}_i,\boldsymbol{a}_i,\bar{\boldsymbol{a}}_i;\vartheta_i')$ by (30)
23: **end if**
24: **end for**

Convergence Analysis: In the following, we theoretically establish the convergence of the OCSL algorithm. Nash equilibrium is a standard solution in game theory, where no node can improve its reward by unilaterally deviating from the collaboration policy, ensuring the convergence of the OCSL mechanism. Therefore, we focus on the OCSL mechanism in the context of Nash equilibrium. Our goal here is therefore to design an effective OCSL mechanism by learning an optimal collaborative strategy $\boldsymbol{\pi}^* = (\boldsymbol{\pi_1^*},\ldots,\boldsymbol{\pi_i^*},\ldots,\boldsymbol{\pi_{|\mathcal{V}|}^*})$ for each node to maximize its own long-term average reward with Nash equilibrium.

Assumption 1: To prove the convergence of the algorithm, we begin by stating the following commonly used assumptions [26]:

1) The learning rate $\eta(k)$ of collaborating nodes in (29) decreases over time and satisfies $\sum_{k=0}^{\infty}\eta(k)=\infty$ and $\sum_{k=0}^{\infty}(\eta(k))^2<\infty$.
2) Each option and action of collaborating nodes can be accessed infinitely frequently and can be chosen with a non-zero probability.

3) The collaboration strategy $\boldsymbol{\pi}$ of collaborating nodes is greedy in the limit with infinite exploration, and the state set is finite.
4) During the learning process, the Nash equilibrium $\boldsymbol{\pi}^* = (\pi_1^*, \ldots, \pi_{|\mathcal{V}|}^*)$ can be considered as a) a global optimal value or b) a saddle point.

To prove the convergence of OCSL mechanism, we then introduce the following Lemma.

Lemma 1: In Markov games, if Assumption 1 holds, the Q-function values of nodes computed by the update rule in (27) will converge to the Nash Q-values $\boldsymbol{Q}_* = [Q_*^1, \ldots, Q_*^{|\mathcal{V}|}]$.

Please see Theorem 1 in [26] for a detailed derivation of proof for Lemma 1. We include it here to stay self-contained.

Theorem 1: For any collaborative learning task defined as in (27), the proposed OCSL algorithm converges to the Nash equilibrium under the Assumption 1.

Proof: With Lemma 1, we now present the proof of Theorem 1 by demonstrating that our proposed OCSL mechanism satisfies all the conditions of Assumption 1. Given that the learning rate $\eta(k)$ of the main Q-network is $\eta(k) = \frac{1}{n(\boldsymbol{a_i},k)+1}$, where $n(\boldsymbol{a_i}, k)$ is the total number of times that action $\boldsymbol{a_i}$ has been selected in k iterations. Therefore, it satisfies the first condition of Assumption 1, i.e., $\sum_{k=0}^{\infty} \frac{1}{n(\boldsymbol{a_i},k)+1} = \infty$ and $\sum_{k=0}^{\infty} \frac{1}{(n(\boldsymbol{a_i},k)+1)^2} < \infty$. Since the proposed ADFL algorithm employs a stochastic Boltzmann exploration strategy, it satisfies the second condition of the assumption. Next, it needs to be demonstrated that the collaboration policy $\boldsymbol{\pi}$ becomes greedy in the limit of infinite exploration. Starting from (25), the policy $\boldsymbol{\pi}$ is shown to be greedy with respect to $\bar{Q}_i^{\omega}(\boldsymbol{s}, \boldsymbol{\omega}_i, \bar{\boldsymbol{\omega}}_i, \boldsymbol{a}_i, \bar{\boldsymbol{a}}_i)$. This implies that the probability of choosing a decision with a higher Q-function value is higher than that of other decisions. Furthermore, as the parameter η gradually tends to zero, the collaboration policy still has a chance to choose non-dominant decisions. Therefore, the policy $\boldsymbol{\pi}$ satisfies the third condition of Assumption 1. Under the fourth condition of Assumption 1, according to the update rule in (27), a contraction mapping is formed on the complete space from $\mathcal{Q}$ to the fixed point of Nash Q-values Q^* in the entire space of the Markov game [27]. This implies that

$$\begin{aligned}\gamma \parallel \Delta(\boldsymbol{s}, \boldsymbol{\omega}, \boldsymbol{a}) \parallel_{\infty} &= \gamma \parallel \bar{Q}_i^{\omega}(\boldsymbol{s}, \boldsymbol{\omega}_i, \bar{\boldsymbol{\omega}}_i, \boldsymbol{a}_i, \bar{\boldsymbol{a}}_i) - Q_* \parallel_{\infty} \\ \geq &\parallel R_i(k) + \gamma \sum_{\boldsymbol{a}_i} [\pi_i^{\omega}(\boldsymbol{a}_i | \boldsymbol{s}(k), \bar{\boldsymbol{a}}_i(k)) * \bar{Q}_i^{\omega}(\boldsymbol{s}(k+1), \\ &\boldsymbol{\omega}_i(k+1), \bar{\boldsymbol{\omega}}_i(k+1), \boldsymbol{a}_i(k+1), \bar{\boldsymbol{a}}_i(k+1))] \parallel_{\infty} . \end{aligned} \tag{32}$$

Hence, according to Theorem 1 in [28], $\Delta(\boldsymbol{s}, \boldsymbol{\omega}, \boldsymbol{a})$ converges to zero with probability 1, and therefore, OCSL can converge to the Nash equilibrium of the Markov game. ■

Complexity Analysis: The complexity of the collaboration policy learning process in each iteration is $\mathcal{O}(|\Omega_i| + |\mathcal{A}_i| + Bmn_w)$. The first term is the complexity of generating the decentralized training strategy in Definition 1, where $|\Omega_i|$ is the number of options for collaborating node i. The section term is the complexity of generating the resource scheduling strategy in Definition 2, where $|\mathcal{A}_i|$ is the number of intra-option actions for collaborating node i. The third term represents the complexity of the training process of the dual neural network, where m and n_w are the number of features and parameters of the two neural networks, respectively.

V. Performance Evaluation

This section presents the experimental results of the proposed ADFL framework, which enables heterogeneous nodes to learn tailored collaboration strategies, thus maximizing the efficiency of the DFL training process in MEC systems. In the following, we detail the experimental setup, datasets, baselines, and experimental evaluations.

A. Experimental Setup

1) Environments: In this paper, we consider a heterogeneous collaborative MEC system, where the number of collaborating nodes varies from 4 to 16, with the default setting being 8 unless otherwise specified. Three types of collaborating nodes are considered: base stations, Type I nodes, and Type II nodes. The number of base stations is chosen from the set $\{1, 2, 3\}$ and the wired transmission delay between base stations $\tau_{i,j}^f = 40$ ms [20]. Moreover, the number of resource blocks (RBs) available for collaborative learning is distributed uniformly in the range $[20, 30]$. The maximum transmit power of base stations is 600 mW. The type I node has a set of RBs distributed uniformly in the range $[10, 20]$ with a maximum transmit power of 200 mW. The type II node is allocated a set of RBs distributed uniformly in the range $[5, 10]$, and has a maximum transmit power of 100 mW. The computational load $\delta_i(k)$ of collaborating nodes is randomly distributed in the range $[0, 70\%]$. For the computation setups, the maximum computing frequencies are 2.4 GHz, 1.8 GHz, and 1.6 GHz for base stations, type I nodes, type II nodes, respectively. In the experiments, we set the parameters for the Gauss-Markov random model as follows: κ_1 and κ_2 are set as 0.5. The mean velocities of users $\bar{v}_i$ range randomly between 0 and 2m/s. Moreover, the average directions of users $\bar{\phi}_i$ range randomly between 0 and 0.2π radians. Φ_i and Ψ_i follow Gaussian distributions with mean 0 and variance 0.2, denoted as $\mathcal{N}(0, 0.2)$. The wireless channel power gain $h_{i,j}(k)$ between collaborating nodes i and j follows an exponential distribution with a mean of $\varsigma_0(d_0/d_{i,j}(k))^4$, where $\varsigma_0 = -40$ dB is the reference distance of 1 m path loss constant. The distance between nodes is randomly distributed in the range $[1, 100]$ m. The other parameters are largely consistent with previous work [29], [30], [31].

2) Dataset and Local Models: In this simulation, the performance of the proposed algorithm is evaluated using four datasets: MNIST, FMNIST, CIFAR-10, and Sensorless Drive Diagnosis (DIAG):

- *MNIST* [32] is a handwritten digits 0-9 dataset with a training set of 60,000 examples, and a test set of 10,000 examples.
- *FMNIST* [33]: is a fashion categories dataset consisting of 70,000 examples, where each example is a 28x28 grayscale image, associated with a label from 10 classes.
- *CIFAR-10* [34]: is an image classification dataset containing 60,000 samples with 3,072 features. It consists of 10 categories, including airplanes, cars, birds, etc., with 6,000 samples per category.

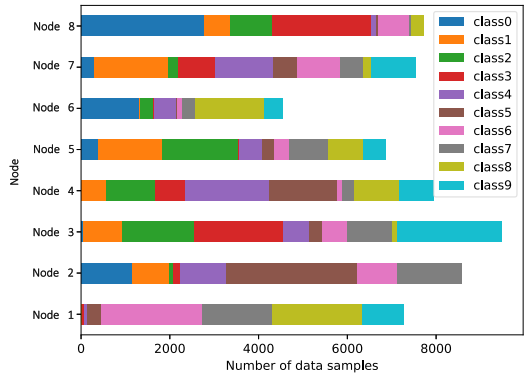


Fig. 4. Partitions of CIFAR-10 dataset under non-iid settings.

- *DIAG* [35]: is extracted from electric current drive signals, which results in 11 different conditions for detecting defective equipment. The dataset consists of 58509 samples with 49 features with 11 classes.

Given that data is collected by different nodes in a collaborative MEC system, we apply the non-identically distributed data partition settings for each dataset, as illustrated in Fig. 4. Specifically, the distribution of each class in the local dataset of each cooperating node follows a Dirichlet distribution [36] with parameter $\alpha_{dir} \in [0, 1]$. In the experiments, the parameter α_{dir} varies from 0.3 to 0.7, where a smaller value of α_{dir} indicates a higher degree of data distribution heterogeneity among the nodes. Moreover, the number of samples in each cooperating node is imbalanced, with variances following the Log-Normal distribution [37]. Additionally, each cooperating node's local dataset is split in a ratio of 7:1:2 for training, validation, and testing. To train the collaborative learning task, we employ the LeNet-5 [32], ResNet18 [38], VGG-19 [39], and FT-transformer [40] as the local model for MNIST, FMNIST, CIFAR-10, and DIAG datasets, respectively.

3) Baselines and Evaluation Metrics: To evaluate the performance of the proposed ADFL algorithm, four baselines are introduced in the experiments, including Solo, D-PSGD [9], CDSGD [11], IDEA [17]. The details of the algorithms are summarized below.

- *Solo* is a baseline algorithm where each collaborating node performs local model training only using its local dataset without interacting with neighbors. Thus, it provides a lower bound on performance in the absence of data collaboration.
- *D-PSGD* [9] is a distributed parallel stochastic gradient descent algorithm for solving large-scale machine learning problems, where each collaborating node randomly pulls the model parameters from one of its neighbors in each iteration.
- *CDSGD* [11] achieves data parallelization and distributed computation in a collaborative way, where each collaborating node pulls models from all neighbors in each iteration.
- *IDEA* [17] is a distributed federated learning approach based on RL, designed to facilitate the participation of heterogeneous nodes in the collaborative data process.

TABLE I
BMTA WITH STANDARD DEVIATION OF ALL COLLABORATING NODES

Settings	MNIST	FMNIST	CIFAR-10	DIAG
Solo	82.70±4.57	80.13±4.72	70.75±5.71	74.27±5.01
D-PSGD	90.31±2.96	89.88±4.03	81.72±4.86	79.36±4.37
CDSGD	91.75±2.83	88.16±3.26	82.03±4.71	85.24±4.33
IDEA	93.36±1.67	91.48±2.77	85.97±3.93	85.03±4.12
ADFL	**95.67±1.28**	**93.74±2.23**	**88.47±3.72**	**88.47±3.23**

- *ADFL* is the proposed adaptive decentralized federated learning framework for collaborative MEC systems, where each node learns the optimal collaboration strategy through the proposed OCSL mechanism.

To comprehensively evaluate the framework's performance, the following metrics are mainly used: i) *Best Mean Testing Accuracy (BMTA)* is the average of the best accuracy achieved by collaborating nodes on the testing dataset. ii) *Average Learning Efficiency* is the average of the learning efficiency among all the participating nodes during the training process: $ALE = \frac{\sum_{k=1}^{K}\sum_{i\in\mathcal{V}} E_i(k)}{K\cdot|\mathcal{V}|}$. iii) *Average Execution Time* is the mean execution of each iteration, including local training and model transmission. iv) *Communication Cost* measures the average amount of model data transmitted by collaborating nodes in each iteration.

4) Implementation Details: We employ the SGD optimizer for local model training with a batch size of 128, momentum of 0.9, and weight decay of 1e-4. The learning rate for the initial model training is set to 0.01. Then the learning rate decays by a factor of 10 when the iteration steps are $\frac{K}{3}$ or $\frac{2K}{3}$, where K is the total number of iterations. The dual critic networks are two 4-layer MLP networks with each hidden layer having 64 neurons. The parameters are set as follows: $B = 10$, $\gamma = 0.9$, and $\alpha = 0.5$. To conduct experiments, we make use of a server with i9 CPU @ 3.5 GHz (8 cores), 128 GB RAM, and 2 NVIDIA RTX 3090 GPU.

B. Simulation Results and Analysis

First, we compare the proposed ADFL with the four baselines in terms of model effectiveness. Table I summarizes the average BMTA with standard deviation value for all collaborating nodes under four datasets. As expected, the BMTA of all DFL methods, such as D-PSGD, CDSGD, IDEA, and the proposed ADFL, is significantly higher than the Solo method, validating the effectiveness of collaborative learning frameworks. Furthermore, the average BMTA for all nodes shows that the ADFL outperforms D-PSGD, CDSGD, and IDEA algorithms. This is attributed to the fact that stationary collaborative training strategies, such as D-PSGD, CDSGD, and IDEA algorithms, leading to a reduction in the accuracy of their local models. In contrast, the proposed ADFL can adaptively adjust the collaboration strategy for each node based on feedback at each iteration, maximizing the efficiency of collaborative learning.

Table II shows BMTA of all collaborating nodes across different algorithms and α_{dir} settings, where smaller α_{dir} values

TABLE II
BMTA OF COLLABORATING NODES UNDER DIFFERENT α_{dir} SETTINGS

Settings	α_{dir} 0.3	0.4	0.5	0.6	0.7
Solo	81.44	82.79	83.43	84.32	84.61
D-PSGD	89.19	90.94	92.17	92.51	92.32
CDSGD	90.25	91.42	92.59	92.41	92.59
IDEA	92.39	93.16	94.55	94.97	95.18
ADFL	**95.37**	**95.54**	**96.61**	**95.17**	**96.62**

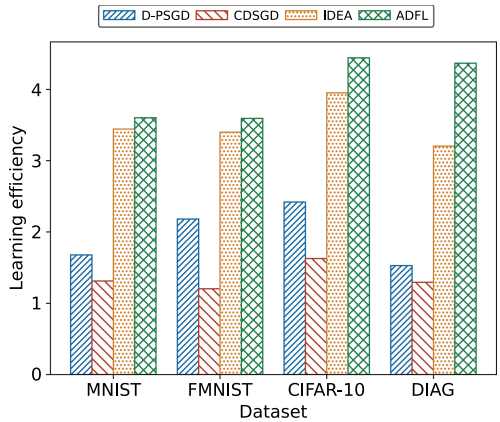


Fig. 5. Learning efficiency comparison among different algorithms under four datasets.

indicate greater disparities in local data distributions. ADFL consistently outperforms all other algorithms across all α_{dir} settings, demonstrating its effectiveness in handling non-IID data distributions. At $\alpha_{dir} = 0.3$, ADFL achieves a BMTA of 95.37%, significantly higher than the other methods, indicating its robustness in highly heterogeneous environments. Solo, D-PSGD, CDSGD, and IDEA show lower BMTA values, highlighting the advantages of ADFL's adaptive and collaborative learning mechanisms. These results underscore ADFL's superior performance in diverse data distribution scenarios, making it highly suitable for real-world distributed learning systems.

We then compare the average learning efficiency of collaborating nodes under different algorithms. As depicted in Fig. 5, the ADFL algorithm exhibits the highest learning efficiency, followed by IDEA and D-PSGD, while CDSGD demonstrates the lowest learning efficiency. The superior performance of ADFL can be attributed to its objective of learning an adaptive collaboration strategy by maximizing efficiency. In contrast, D-PSGD and CDSGD employ stationary collaboration strategies. This suggests that the proposed adaptive collaboration strategy can effectively capture the dynamics of environmental changes and implicit relationships between collaborating nodes and their local datasets.

In the following, we analyze the average execution time of each iteration under different algorithms, including local training time and communication time. Fig. 6 shows that the proposed ADFL framework has the shortest executed time compared to the other three algorithms, validating the effectiveness of the proposed adaptive collaboration strategy. Specifically, the average local training time for CDSGD and D-PSGD is almost the same. In contrast, the average local training time of the proposed ADFL is lower than those under all datasets. This is because, unlike other algorithms that require local training in each iteration, the proposed adaptive collaboration strategy can decide whether to conduct local training in specific iteration rounds based on the current state. Therefore, compared to the CDSGD and D-PSGD algorithms, the local training time of the proposed ADFL algorithm is significantly reduced. Additionally, it can be observed that in collaborative MEC systems, communication time is much higher than local training time, which emphasizes the importance of improving collaborative learning efficiency. As expected, Fig. 6 shows that the communication time of the proposed ADFL algorithm is the smallest, followed by IDEA and D-PSGD, while CDSGD has the longest communication time. This observation aligns with the earlier analysis, reinforcing the efficiency gains achieved by the proposed ADFL algorithm in both local training time and communication time.

Subsequently, we examine the average communication cost per iteration of all collaborating nodes under different algorithms. In this experiment, the number of collaborating nodes in the MEC system increases from 4 to 16. Communication cost refers to the amount of model data (GB) transmitted. The size of the VGG-19 model is 79.46 MB. From Fig. 7, it can be observed that the communication cost of all algorithms increases with the growth of the number of nodes. This is because, with an increase in collaborating nodes, the communication frequency per round of collaboration increases, leading to an increase in the transmitted data volume. Furthermore, it can be observed that the communication cost of the proposed ADFL is significantly lower than the other three algorithms. This is because the proposed collaboration strategy, learned based on the OCSL mechanism, can effectively explore implicit relationships between nodes and adapt to the dynamic nature of the environment.

Finally, we analyze the convergence performance of the ADFL algorithm. In examining the two design enhancements presented in Section IV to improve ADFL's performance, namely option-based decomposition and dual critic networks approximation, we conduct an ablation study to assess their effects on Q-value estimations. Fig. 8 presents the results on CIFAR-10 under ADFL, utilizing both option-based decomposition and dual critic networks, along with three variants: i) No decomposition + Dual critic networks approximation (No+Dual); ii) Option-based decomposition + Single critic network approximation (Option+Single); and iii) No decomposition + Single critic network (No+Single). From Fig. 8, we can see that ADFL converges faster than the No+Dual variant, indicating that option-based decomposition accelerates the convergence process. This efficiency gain is attributed to the reduction in the action space achieved through option-based decomposition, enhancing the effectiveness of the Boltzmann exploration policy. Furthermore, Fig. 8 also demonstrates that ADFL effectively mitigates the overestimation issue when compared with the Option+Single variant. The introduction of a target network enables ADFL to separate action selection and evaluation operations into

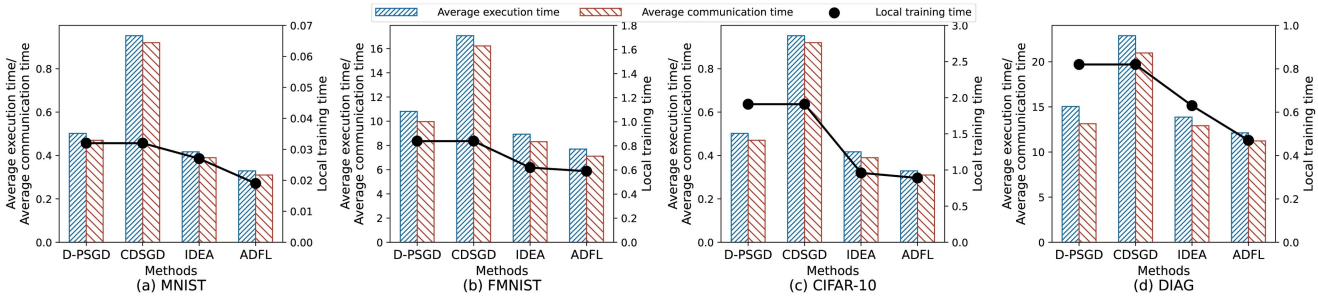


Fig. 6. Average run time comparison among different algorithms under four datasets.

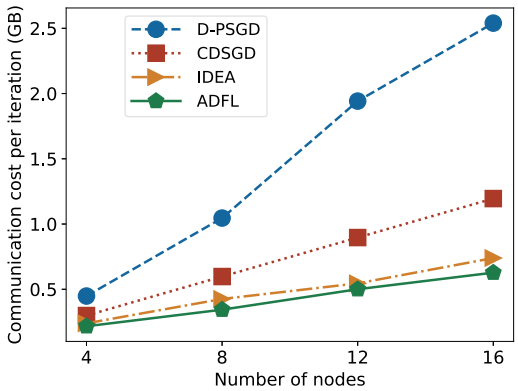


Fig. 7. Average communication cost comparison among different algorithms under CIFAR-10 dataset.

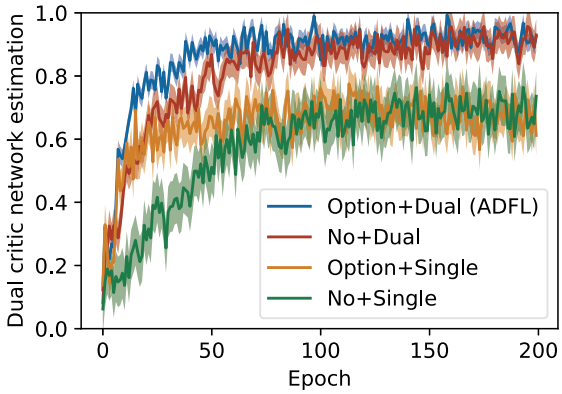


Fig. 8. Convergence performance of ADFL framework.

two interacting networks, enhancing the robustness of the main network through learning from the target network.

VI. Conclusion and Future Work

In this paper, we have proposed an ADFL framework that empowers diverse nodes to learn customized collaboration strategies, optimizing the efficiency of DFL training in MEC systems. To address heterogeneity, our framework incorporates a general collaboration strategy model, allowing participating nodes with varying conditions to engage in tailored cooperative actions. To handle the dynamic nature of MEC systems, we have formulated collaboration strategy learning as a stochastic game, employing a MARL approach. This enables nodes to adaptively learn collaboration strategies by capturing implicit features of dynamic MEC systems. Additionally, we have proposed an effective OCSL mechanism, ensuring nodes learn objective-specific adaptive strategies. We have theoretically proved that the learned collaboration strategy achieves the Nash equilibrium. Extensive experiments on real-world datasets have demonstrated that ADFL outperforms state-of-the-art baselines in model accuracy, efficiency, and communication cost.

In future work, to further enhance the learning efficiency, we will investigate the adaptive model quantization and distillation mechanisms of DFL for MEC systems, which can potentially reduce communication overhead and improve the flexibility of the learning process in heterogeneous and dynamic network environments.

References

[1] M. Zwolenski and L. Weatherill, "The digital universe: Rich data and the increasing value of the Internet of Things," *J. Telecommun. Digit. Economy*, vol. 2, no. 3, pp. 47.1–47.9, 2014.

[2] T. Zhang, K.-Y. Lam, J. Zhao, and J. Feng, "Joint device scheduling and bandwidth allocation for federated learning over wireless networks," *IEEE Trans. Wireless Commun.*, early access, pp. 1–13, Jul. 12, 2023, doi: 10.1109/TWC.2023.3291701.

[3] H. Chen, S. Huang, D. Zhang, M. Xiao, M. Skoglund, and H. V. Poor, "Federated learning over wireless IoT networks with optimized communication and resources," *IEEE Internet Things J.*, vol. 9, no. 17, pp. 16 592–16 605, Sep. 2022.

[4] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in IoT," *IEEE Internet of Things J.*, vol. 7, no. 7, pp. 5986–5994, Jul. 2020.

[5] V.-D. Nguyen, S. K. Sharma, T. X. Vu, S. Chatzinotas, and B. Ottersten, "Efficient federated learning algorithm for resource allocation in wireless IoT networks," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3394–3409, Mar. 2021.

[6] J. Kang et al., "Communication-efficient and cross-chain empowered federated learning for Artificial Intelligence of Things," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 5, pp. 2966–2977, Sep./Oct. 2022.

[7] D. Yang et al., "DetFed: Dynamic resource scheduling for deterministic federated learning over time-sensitive networks," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 5162–5178, May 2024.

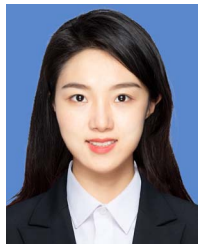
[8] P. Li, Y. Zhong, C. Zhang, Y. Wu, and R. Yu, "FedRelay: Federated relay learning for 6G mobile edge intelligence," *IEEE Trans. Veh. Technol*, vol. 72, no. 4, pp. 5125–5138, Apr. 2023.

[9] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5330–5340.

[10] X. Jia et al., "Highly scalable deep learning training system with mixed-precision: Training ImageNet in four minutes," 2018, *arXiv: 1807.11205*.

[11] Z. Jiang, A. Balu, C. Hegde, and S. Sarkar, "Collaborative deep learning in fixed topology networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5904–5914.

[12] A. Salama, A. Stergioulis, S. A. Zaidi, and D. McLernon, "Decentralized federated learning on the edge over wireless mesh networks," *IEEE Access*, vol. 11, pp. 124 709–124 724, 2023.
[13] H. Xu, M. Chen, Z. Meng, Y. Xu, L. Wang, and C. Qiao, "Decentralized machine learning through experience-driven method in edge networks," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 2, pp. 515–531, Feb. 2022.
[14] Z. Wu, X. Wu, and Y. Long, "Joint scheduling and robust aggregation for federated localization over unreliable wireless D2D networks," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 3, pp. 3359–3379, Sep. 2023.
[15] S. Chen, Y. Xu, H. Xu, Z. Jiang, and C. Qiao, "Decentralized federated learning with intermediate results in mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 23, no. 1, pp. 341–358, Jan. 2024.
[16] Y. Qu et al., "Decentralized federated learning for UAV networks: Architecture, challenges, and opportunities," *IEEE Netw.*, vol. 35, no. 6, pp. 156–162, Nov./Dec. 2021.
[17] Y. Wang, Y. Wu, X. Chen, G. Feng, and B. C. Ooi, "Incentive-aware decentralized data collaboration," in *Proc. ACM Manage. Data*, vol. 1, no. 2, pp. 1–27, 2023.
[18] Q. Liu, L. Shi, L. Sun, J. Li, M. Ding, and F. Shu, "Path planning for UAV-mounted mobile edge computing with deep reinforcement learning," *IEEE Trans. Veh. Technol*, vol. 69, no. 5, pp. 5723–5728, May 2020.
[19] S. Batabyal and P. Bhaumik, "Mobility models, traces and impact of mobility on opportunistic routing algorithms: A survey," *IEEE Commun. Surveys Tut.*, vol. 17, no. 3, pp. 1679–1707, Third Quarter, 2015.
[20] T. Cao, Y. Huang, J. Wang, T. Ji, S. Huang, and S. Zhou, "Key competence analysis of non-terrestrial network-based cellular Backhaul," in *Proc. IEEE Veh. Technol. Conf.*, 2021, pp. 1–5.
[21] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.
[22] G. Mitsis, E. E. Tsiropoulou, and S. Papavassiliou, "Price and risk awareness for data offloading decision-making in edge computing systems," *IEEE Syst. J.*, vol. 16, no. 4, pp. 6546–6557, Dec. 2022.
[23] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," in *Proc. AAAI Conf. Artif. Intell.*, 2017, pp. 1726–1734.
[24] L. E. Blume, "The statistical mechanics of strategic interaction," *Games Econ. Behav.*, vol. 5, no. 3, pp. 387–424, 1993.
[25] A.-L. Barabási, R. Albert, and H. Jeong, "Mean-field theory for scale-free random networks," *Physica A: Statist. Mechanics Appl.*, vol. 272, no. 1/2, pp. 173–187, 1999.
[26] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5571–5580.
[27] J. Hu and M. P. Wellman, "Nash Q-learning for general-sum stochastic games," *J. Mach. Learn. Res.*, vol. 4, no. Nov., pp. 1039–1069, 2003.
[28] T. Jaakkola, M. Jordan, and S. Singh, "Convergence of stochastic iterative dynamic programming algorithms," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 1993, pp. 1185–1201.
[29] Z. Meng, H. Xu, M. Chen, Y. Xu, Y. Zhao, and C. Qiao, "Learning-driven decentralized machine learning in resource-constrained wireless edge computing," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.
[30] X. Chen, L. Pu, L. Gao, W. Wu, and D. Wu, "Exploiting massive D2D collaboration for energy-efficient mobile edge computing," *IEEE Wireless Commun.*, vol. 24, no. 4, pp. 64–71, Aug. 2017.
[31] T. Cao, Y. Huang, J. Wang, T. Ji, S. Huang, and S. Zhou, "Key competence analysis of non-terrestrial network-based cellular Backhaul," in *Proc. IEEE 94th Veh. Technol. Conf.*, 2021, pp. 1–5.
[32] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[33] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv: 1708.07747*.
[34] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/dataset+for+sensorless+drive+diagnosis
[35] U. M. L. Repository, "Sensorless drive diagnosis dataset," 2017. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/dataset+for+sensorless+drive+diagnosis
[36] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, N. Hoang, and Y. Khazaeni, "Bayesian nonparametric federated learning of neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7252–7261.
[37] A. E. Durmus, Z. Yue, M. Ramon, M. Matthew, W. Paul, and S. Venkatesh, "Federated learning based on dynamic regularization," in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 1–12.
[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[39] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[40] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko, "Revisiting deep learning models for tabular data," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 18 932–18 943.



Yatong Wang (Member, IEEE) received the BE and PhD degrees in communication and information engineering from the University of Electronic Science and Technology of China, in 2018 and 2023, respectively. From 2021 to 2022, she was a visiting student with the Database Laboratory, School of Computing, National University of Singapore, Singapore. She is currently a postdoctoral fellow with the School of Information and Electronics, Beijing Institute of Technology, China. Her research interests include intelligent wireless networking, UAV swarm-based cooperative sensing, and distributed machine learning.

Zhongyi Wen received the BE degree from the University of Electronic Science and Technology of China (UESTC), in 2023. He is currently working toward the PhD degree with the School of Information and Communication Engineering, UESTC. His research interests include machine learning, optimization, and adaptive signal processing in the fields of radio frequency fingerprinting identification, AI communications, and medical images.

Yunjie Li (Senior Member, IEEE) received the PhD degree in signal and information processing from the Beijing Institute of Technology (BIT), China, in 2005. Currently, he is a professor with the School of Information and Electronics, BIT, China. His research interests include intelligent system and signal processing.

Bin Cao (Senior Member, IEEE) is a full professor with the State Key Laboratory of Network and Switching Technology, Beijing University of Posts and Telecommunications. He received the IEEE Outstanding Leadership Award in 2020, the Best Paper Award at IEEE BMSB 2021, and the IEEE TEMS Mid-Career Award in 2021. He is an associate editor of *IEEE Transactions on Mobile Computing* and *Digital Communications and Networks*, serves/served as a (lead) guest editor of the *IEEE Communications Magazine*, *IEEE Internet of Things Journal*, *IEEE Transactions on Industrial Informatics*, and *IEEE Sensors Journal*, as well as the co-chair for IEEE ICNC 2018, IEEE Blockchain 2020, and IEEE Globecom 2022. He is the founding vice chair of Special Interest Group on Wireless Blockchain Networks in IEEE Cognitive Networks Technical Committee, and a Chief Young scientist of the National Key Research and Development Program of China.