Federated Deep Reinforcement Learning for Task Offloading in MEC-Enabled Heterogeneous Networks

Hui Xiao, Zhigang Hu, Xinyu Zhang, Aikun Xu, Meiguang Zheng, and Keqin Li, *Fellow, IEEE*

***Abstract*—The integration of mobile edge computing (MEC) and heterogeneous networks enables network operators to provide task offloading services to a large number of user devices (UDs) for low-latency task processing by equipping macro base stations and densely deployed small base stations with edge servers. Federated deep reinforcement learning allows each UD to collaboratively learn useful knowledge from the interaction with the environment in a privacy-preserving and high-efficiency way and thus has been applied to solve the task offloading problem in recent studies. However, very few of these studies have considered the energy and time costs incurred by the federated learning process. In this article, the goal is to minimize the total UDs' energy consumption while guaranteeing deadline constraints considering both the task offloading process and the federated learning process in MEC-enabled heterogeneous networks. Toward this end, we propose a federated deep *Q*-network (DQN) method where each UD optimizes the offloading decision for the offloading process and the participation decision and training volume for the learning process based on its local DQN model. The simulation results demonstrate the proposed method is superior to several existing methods in terms of energy efficiency and Quality of Service (QoS).**

***Index Terms*—Deep reinforcement learning (DRL), federated learning, mobile edge computing (MEC), task offloading.**

I. Introduction

With the rapid growth in intelligent applications and the exponential increase in data, user devices (UDs) must handle numerous latency-critical and computation-intensive tasks, such as real-time gaming and face recognition [1]. Despite advancements in UDs, it remains challenging for them to complete all tasks locally under strict delay requirements due to their limited battery life and computational capacity. One viable solution to enable efficient task execution is to leverage the mobile edge computing (MEC) paradigm, which brings communication and computation closer to UDs [2]. In current fifth-generation (5G) networks, the dense deployment of small base stations (SBSs) facilitates the integration of MEC into the radio access network (RAN) [3]. MEC offers a highly virtualized platform, providing edge servers with a flexible application environment capable of meeting the real-time interaction demands of multiple UDs. Benefiting from MEC, each UD can decide whether to transfer its computational tasks to edge servers to either accelerate task execution or conserve energy [4], [5]. However, the limited radio spectrum cannot accommodate a large number of UDs simultaneously uploading task data to a single SBS [6]. Moreover, excessive task offloading may overload edge servers, creating computational bottlenecks [7]. Consequently, blind offloading decision making can lead to suboptimal Quality of Service (QoS) due to high transmission and processing delays. Therefore, it is crucial to consider various factors, such as resource capacity and task requirements, in task offloading decision making to ensure optimal energy consumption and latency for UDs.

Several existing works have focused on the aforementioned factors and proposed various computation offloading methods based on mathematical programming [8] (e.g., dynamic programming and convex programming) and heuristic rules [9] (e.g., genetic algorithms and ant colony optimization). Compared with mathematical programming, which can handle multiple variables and constraints to capture the problem and achieve optimal solutions, heuristic methods offer the advantage of lower computational complexity, particularly in large-scale systems [10]. However, both approaches require accurate prior resource information and incur significant running time, making them inadequate for real-time decision making under uncertain resource availability, high UD mobility, and dynamic task requirements. To address this limitation, recent research on computation offloading has adopted reinforcement learning (RL) techniques, which can make real-time optimal decisions through knowledge gained from interactions with the environment, without requiring prior information on the system dynamics and modeling [11]. To overcome the curse of dimensionality inherent in RL, an alternative technique, deep RL (DRL) [12], has been introduced to the field of task offloading. DRL utilizes deep neural networks to assist agents in learning complex policies, thereby accelerating the training process. Traditional centralized RL/DRL methods require a central server to collect data from agents for training [13]. While this approach results in considerable network overhead and raises privacy concerns for UDs, insufficient data

Received 18 February 2024; revised 21 October 2024; accepted 27 November 2024. Date of publication 2 December 2024; date of current version 9 April 2025. This work was supported in part by the Natural Science Foundation of China under Grant 62172442, and in part by the Youth Science Foundation of Natural Science Foundation of Hunan Province under Grant 2020JJ5775. *(Corresponding author: Zhigang Hu.)*

Hui Xiao, Zhigang Hu, Xinyu Zhang, Aikun Xu, and Meiguang Zheng are with the School of Computer Science and Engineering, Central South University, Changsha 410083, China (e-mail: huixiao@csu.edu.cn; zghu@csu.edu.cn; zhangxinyu11014@163.com; aikunxu@csu.edu.cn; zhengmeiguang@csu.edu.cn).

Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

Digital Object Identifier 10.1109/JIOT.2024.3509893

sharing among UDs, such as independent training on each UD, can lead to isolated information islands and poor learning performance.

To balance personal privacy protection and learning performance, researchers have adopted the federated learning framework, which enables collaborative training without sharing personal data [14]. Specifically, each UD can selectively participate in training rounds, using its local dataset to train the model independently and transmitting the model updates to a centralized entity [15]. The centralized node then aggregates these updates to generate new global model parameters, which are subsequently distributed to UDs. Consequently, each training round requires computation and communication resources for both local model training and model update transmission [16]. When applying federated learning to address task offloading in resource-constrained MEC scenarios, there exists a complex interaction between the offloading and learning processes, which has often been overlooked in previous studies. On one hand, model training may compete with task processing for limited resources and time. On the other hand, efficient model training allows UDs to derive valuable information from their local datasets, leading to high-quality offloading solutions. Thus, achieving a balanced allocation of time and resources between task offloading and model training is critical [17], as it reduces energy consumption incurred by task processing and model training under specified latency constraints. Moreover, studies [18], [19] have shown that the number of UD agents participating in each federated learning round significantly impacts the cost, accuracy, and convergence of model training. Consequently, designating the edge server co-located with the SBS as the central entity may limit federated learning performance due to the restricted coverage of the SBS. To address this limitation, heterogeneous networks that employ a macro base station (MBS) to overlay multiple SBSs can be utilized to enhance network capacity [20]. In such a heterogeneous network, the MBS offers broader coverage and more extensive communication and computation resources, making it an ideal central entity for the federated learning process.

In this work, we propose a federated learning-inspired offloading mechanism for an MEC-aided heterogeneous network, which consists of three layers: 1) UDs; 2) SBSs; and 3) MBSs. SBSs provide task offloading services to UDs within their coverage areas, while each UD learns from its experiences to make informed offloading decisions in a distributed manner. MBSs act as centralized nodes, aggregating information from UDs. In task offloading, UDs transmit raw data to the edge server, with the uplink data volume significantly larger than the processed results returned via downlink, as indicated in [21], [30], and [38]. Additionally, uplink transmission rates are typically slower than downlink rates in MEC environments, making uplink transmissions dominant in terms of energy and time consumption, while the impact of downlink transmission is negligible [21], [30], [38]. In federated learning, UDs upload model updates that are similar in size to the global model returned after aggregation. However, despite comparable data sizes, the slower uplink rates and concurrent transmissions from multiple UDs increase energy and time consumption, as noted in [39], [40], and [41]. Conversely, the faster downlink rates and lower energy requirements make the impact of downlink transmission negligible [39], [40], [41]. Therefore, we disregard the time and energy consumed by downlink transmission in both task offloading and federated learning. In line with [13], [21], and [22], our goal is to minimize energy consumption for UDs while ensuring maximum delay constraints, considering the limited battery capacity and desired QoS levels for UDs. We exclude the energy cost of edge nodes (SBSs and MBSs), which benefit from more stable and abundant energy resources. We focus on the energy consumed by the task execution process and the federated learning process, formulating an energy cost minimization problem where offloading decisions, training participation decisions, and training dataset sizes are jointly optimized. This problem is addressed by each UD using federated DRL models. The main contributions of this work include as follows.

1) We propose a federated learning-based framework in MEC-aided heterogeneous networks to achieve real-time, privacy-preserving offloading decisions for delay-sensitive and nonpartitionable tasks. In this framework, each UD functions as a distributed client that learns to determine an optimal location for task processing. MBSs serve as central nodes that facilitate the learning process for distributed clients, while SBSs are responsible for processing tasks offloaded by UDs.
2) We formulate a total energy cost minimization problem for UDs, incorporating constraints of latency and training volume by modeling the joint task offloading and federated learning processes. The objective is to optimize participation decisions and training dataset sizes in each training round to balance training costs and model accuracy, given that the accuracy of federated models directly influences the performance of the offloading decisions they generate.
3) To address the joint offloading and learning optimization problem, we develop a federated deep Q-network (DQN)-based approach that enables each UD to independently make offloading and learning-related decisions based on its local DQN model. Each UD estimates the long-term cost of its actions based on locally observed state information, including task requirements and historical training volume, to guide decision making.
4) Simulation results demonstrate that the proposed method significantly reduces the energy consumption of UDs and the ratio of tasks that violate deadlines by selecting appropriate offloading destinations and adjusting model training intensity.

II. Related Work

In the recent years, researchers have investigated the MEC-enabled computation offloading problem with focus on various optimization objectives, such as energy cost, resource utilization and task completion time, and proposed various task offloading methods based on different optimization techniques.

Zhang et al. [8], Tang and Hu [23], Li et al. [24], Van Huynh et al. [25], and Su et al. [26] proposed computation offloading methods based on mathematical programming, e.g., convex relaxation and dynamic programming. Zhang et al. [8] studied the QoS-aware computation offloading by minimizing the total task execution time, and designed a branch and bound-based method and a generalized benders decomposition (GBD)-based method, respectively. Tang and Hu [23] aimed to minimize the total weighted task delay under battery capacity constraints and utilized successive convex approximation (SCA) to achieve a near-optimal offloading policy and resource distribution. Li et al. [24] developed a task offloading approach enabled by the difference of convex function to minimize the energy expended by UDs while satisfying deadline constraints. Van Huynh et al. [25] integrated multi-UDs with a digital twin MEC network and presented a computation offloading method combining the inner approximation and alternating optimization for minimizing latency. Su et al. [26] studied the sum energy minimization problem for the collaborative cloud–edge computation offloading by exploiting integer linear programming (ILP) and designed a primal-dual-based computation offloading method to tackle the formulated ILP problem.

Another extensively adopted optimization technique is the heuristic rule, e.g., the greedy algorithm and the meta-heuristic algorithm. Guo et al. [27] presented a two-tier task offloading framework in ultradense networks considering different task types and designed a greedy task offloading method which determines the offloading position based on task processing delay. Patel et al. [9] designed a distributed stable matching-inspired method for balancing the energy consumption and monetary cost caused by task offloading. Chen et al. [28] considered dependent tasks under two collaboration scenarios and proposed two heuristic greedy offloading methods to minimize the application completion time, respectively. Chen et al. [29] targeted at the minimization of energy consumption incurred from executing deep neural network-enabled applications for all devices and servers and developed an energy-efficient offloading approach by adopting self-adaptive particle swarm optimization. Focusing on the vehicular scenarios, de Souza et al. [30] modeled the task offloading process as a multiconstraint NP-hard optimization problem with the aim of minimizing application latency and presented a bee colony-based heuristic solution for this problem.

As the task requirements and network conditions are increasingly complex and dynamic, both mathematical programming techniques and heuristic rules need a large amount of iterations to converge to a satisfactory local optimum, and therefore fail to make real-time offloading decisions. Recently, there has been interesting research demonstrating the superior performance of RL and DRL in solving various complex RAN-related decision problems (e.g., task offloading and resource provision) under sophisticated and volatile task demands [31]. Dai et al. [32] developed an cloud-edge-end orchestrated offloading framework where the joint task offloading and resource assignment problem aiming at minimizing system energy cost is expressed as a Markov decision process (MDP) model and tackled by a DRL-based algorithm. Considering the uncertain workload dynamics at the edge servers, Li et al. [33] developed a DQN-enabled offloading method to minimize the cost integrating energy consumption and delay. Tang and Wong [34] designed a distributed offloading method by incorporating DQN and long short-term memory for enabling each UD to make offloading decisions without requiring the information of other UDs. Zhou et al. [35] investigated the system energy minimization problem with delay constraints and proposed two energy-efficient offloading approaches by adopting Q-Learning and double DQN, respectively. To enhance the offloading performance measured by the latency and energy cost, Zhang et al. [36] developed an intelligent offloading method which adopts meta RL for improving the model training speed and DRL for decision making.

However, these works are based on centralized or distributed computation models which ignore the tradeoff between model training performance and personal privacy protection in dynamic network environments. Motivated by this, researchers have proposed federated learning-enabled offloading approaches to enhance the model accuracy performance on the premise of guarding data privacy. Guo et al. [22] investigated the minimization of system cost and designed a federated DRL-based method where each UD acts as a DRL agent to tackle the problem. Prathiba et al. [37] designed a task scheduling and resource management framework enabled by federated Q-Learning to achieve the minimum offloading latency and resource cost. Pan et al. [38] presented a federated DQN-based computation offloading approach for maximizing the system throughput and designed an asynchronous learning policy to improve the learning efficiency. Generally, the performance of federated learning can be enhanced by increasing the update rounds and engaging more UDs in the updating process, which however leads to significant computation and communication costs. Luo et al. [39] modeled the accuracy performance of federated learning deployed in MEC systems and deduce an upper bound of accuracy loss to guide the joint minimization of learning costs and accuracy loss. To enhance the performance of federated learning at a limited cost, Zhang et al. [40] proposed a relay-assisted federated scheduling strategy where each federated node can choose to assist other nodes in data transmission based on the evaluation of revenue and energy consumption. Salh et al. [41] investigated the joint resource scheduling and user selection to minimize the energy cost incurred from the federated learning process while guaranteeing a desired level of training delay.

The significant cost of the federated learning process underscores the importance of balancing model accuracy and learning overhead when designing federated learning-based solutions for resource-limited UDs and wireless networks. However, this aspect has often been overlooked by existing federated offloading mechanisms. Shinde et al. [17] investigated the joint optimization of federated learning and offloading policies, aiming to reduce energy consumption under delay constraints, and proposed a genetic algorithm-inspired method to determine the optimal number of federated learning iterations for improved task offloading performance. Nonetheless, the meta-heuristic approach employed in their

TABLE I
DIFFERENCE BETWEEN OUR APPROACH AND THE MOST RELEVANT APPROACHES

Reference	Optimization Method		Privacy Concerns	Considered Factors			
	Static	Dynamic		Offloading Energy Cost	Offloading Delay	Learning Energy Cost	Learning Delay
Tang et al. [34]		DRL	None		✓		
Zhou et al. [35]		DRL	None	✓	✓		
Zhang et al. [36]		Meta-DRL	None	✓	✓		
Guo et al. [22]		DRL	FL	✓	✓		
Prathiba et al. [37]		Q-Learning	FL	✓	✓		
Pan et al. [38]		DRL	FL		✓		✓
Shinde et al. [17]	GA		FL	✓	✓	✓	✓
Our Approach		DRL	FL	✓	✓	✓	✓

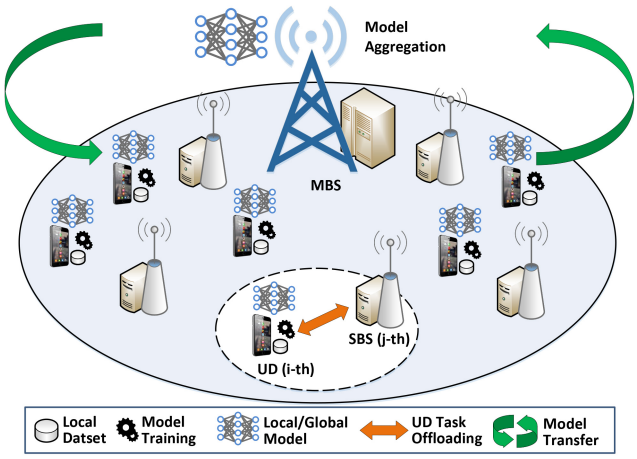

Fig. 1. Architecture of MEC-enabled heterogeneous network.

work is not adaptable to workload dynamics and can only achieve a suboptimal solution. Therefore, the objective of our work is to jointly optimize offloading and learning decisions within a federated MEC-enabled task offloading framework in real-time, aiming to minimize total UD energy consumption while meeting desired latency levels.

To highlight the key features and distinctions of our contributions, we present a qualitative comparison between our proposed approach and the most relevant methods, including [17], [22], [34], [35], [36], [37], [38] in Table I. In the table, "GA" denotes the genetic algorithm and "FL" denotes federated learning. As shown in Table I, our approach advances the field by introducing a joint optimization framework for task offloading and federated learning, addressing privacy concerns while ensuring energy efficiency and QoS guarantees. By employing a DRL-based solution mechanism, our approach adapts effectively to dynamic workloads in MEC-enabled heterogeneous networks.

III. System Model

Fig. 1 illustrates an MEC-enabled heterogeneous network architecture composed of one MBS, M SBSs represented by $\mathcal{M} = \{1, \ldots, M\}$, and N UDs represented by $\mathcal{N} = \{1, \ldots, N\}$. The MBS and SBSs are all associated with an edge server with computational and storage capabilities, where the MBS server is designated as the federated learning platform and the SBS servers provide computation offloading services to UDs. We focus on one time period that is divided into a set of discrete time slots $\mathcal{T} = \{1, \ldots, T\}$, each of which has a time interval of Ω seconds. Without loss of generality, each UD is assumed to generate a computational task at the beginning of each time slot according to [9], [21], and [34]. The computational task of UD i at time slot t can be characterized by the tuple $Task_i^{(t)} = (D_i^{(t)}, C_i^{(t)}, T_{i,\max}^{(t)})$, where $D_i^{(t)}$ represents the size of task input, $C_i^{(t)}$ is the amount of CPU cycles needed to complete the task, and $T_{i,\max}^{(t)}$ denotes the task completion deadline.

The processing capability can vary enormously among UDs, SBSs, and the MBS. We accordingly denote c_i^u, c_j^s, and c^m as the CPU frequency of UD $i \in \mathcal{N}$, SBS $j \in \mathcal{M}$, and the MBS, respectively. Each UD can establish connection with SBS j on a bandwidth B_{ij}^s when entering SBS j's coverage area with radius O_j^s. All UDs are covered by the MBS and can communicate with the MBS on a bandwidth B_i^m. In the following, we focus on making time-slot decisions of task offloading and federated learning in each time slot $t \in \mathcal{T}$.

A. Task Offloading Model

The UDs can determine to transfer their newly arrived tasks to their associated SBSs or process the tasks locally. Here, we consider the practical scenario where tasks are nondivisible according to [9] and [13]. Represent the potential task execution locations by a new set $\mathcal{M}' = \{0, 1, \ldots, M\}$, where index 0 means that the task is performed locally without offloading. Then, a binary offloading variable $u_{ij}^{(t)} \in \{0, 1\}, j \in \mathcal{M}'$ is defined for the task produced by UD i at time slot t, where $u_{ij}^{(t)} = 1, j \neq 0$ denotes that UD i selects SBS j for task execution, and $u_{i0}^{(t)} = 1$ indicates that UD i performs its task locally.

1) Task Uploading Process: When UD i selects SBS j as the offloading destination, the task input data needs to be uploaded to the selected SBS. Here, the result return is neglected due to the high downlink transmission rate and small result data size according to [21], [30], and [38]. For the communication between the UD and the SBS, we calculate the transmission rate from UD i to SBS j based on the Shannon capacity formula [42] as

$$r_{ij}^s(t) = B_{ij}^s \log_2\left(1 + \frac{P_i^u \varrho_{ij}^s(t)}{B_{ij}^s N_0}\right) \forall i \in \mathcal{N}, j \in \mathcal{M}, t \in \mathcal{T} \quad (1)$$

where P_i^u denotes the transmission power of UD i, N_0 denotes the power spectral density of noise, and $\varrho_{ij}^s(t)$ denotes the channel gain between UD i and SBS j.

Then, the transmission time and energy consumed by uploading $Task_i^{(t)}$ of UD i to SBS j is given by

$$T_{ij}^{\mathrm{ou}}(t) = \frac{D_i^{(t)}}{r_{ij}^s(t)} \ \forall i \in \mathcal{N}, j \in \mathcal{M}, t \in \mathcal{T} \tag{2}$$

$$E_{ij}^{\mathrm{ou}}(t) = P_i^u T_{ij}^{\mathrm{ou}}(t) \ \forall i \in \mathcal{N}, j \in \mathcal{M}, t \in \mathcal{T}. \tag{3}$$

2) Task Computation Process: Depending on different offloading decisions, task computation incurs varying degrees of task processing delay and energy cost. For local execution, the computation delay and energy consumption at UD i can be calculated as

$$T_{i0}^{\mathrm{oc}}(t) = \frac{C_i^{(t)}}{c_i^u} \ \forall i \in \mathcal{N}, t \in \mathcal{T}, \tag{4}$$

$$E_{i0}^{\mathrm{oc}}(t) = \kappa \left(c_i^u\right)^2 C_i^{(t)} \ \forall i \in \mathcal{N}, t \in \mathcal{T} \tag{5}$$

respectively, where κ represents the effective switched capacitance [9].

The computation delay for UD i to process its task on SBS j can be given as

$$T_{ij}^{\mathrm{oc}}(t) = \frac{C_i^{(t)}}{c_j^s} \ \forall i \in \mathcal{N}, j \in \mathcal{M}, t \in \mathcal{T}. \tag{6}$$

Here, we ignore the energy consumed by task execution on SBS servers due to the sufficient power supply to the SBSs from grids [13], [21], [22].

3) Task Offloading Problem Formulation: Owing to the constrained battery capacity of UDs and the significance of service quality, we aim to optimize the offloading decision variables $u_{ij}^{(t)}$ to minimize the energy consumed by the UDs while respecting the delay deadlines. The total delay and energy consumption for UD i to complete the task in time slot $t \in \mathcal{T}$ can be detailed as

$$T_i^{\mathrm{o}}(t) = u_{i0}^{(t)} T_{i0}^{\mathrm{oc}}(t) + \sum_{j \in \mathcal{M}} u_{ij}^{(t)} \left(T_{ij}^{\mathrm{ou}}(t) + T_{ij}^{\mathrm{oc}}(t)\right) \tag{7}$$

$$E_i^{\mathrm{o}}(t) = u_{i0}^{(t)} E_{i0}^{\mathrm{oc}}(t) + \sum_{j \in \mathcal{M}} u_{ij}^{(t)} E_{ij}^{\mathrm{ou}}(t) \tag{8}$$

respectively. Then, we have the energy-aware and latency-constrained task offloading optimization problem as follows:

$$\min_{\boldsymbol{u}} \quad \sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{N}} E_i^{\mathrm{o}}(t) \tag{9a}$$

$$\text{s.t.} \quad T_i^{\mathrm{o}}(t) \le T_{i,\max}^{(t)} \ \forall i \in \mathcal{N}, t \in \mathcal{T} \tag{9b}$$

$$u_{ij}^{(t)} d_{ij}^{(t)} \le O_j^s \ \forall i \in \mathcal{N}, t \in \mathcal{T} \tag{9c}$$

$$\sum_{j \in \mathcal{M}'} u_{ij}^{(t)} = 1 \ \forall i \in \mathcal{N}, t \in \mathcal{T} \tag{9d}$$

$$u_{ij}^{(t)} \in \{0, 1\} \ \forall i \in \mathcal{N}, j \in \mathcal{M}', t \in \mathcal{T} \tag{9e}$$

where $\boldsymbol{u} = \{u_{ij}^{(t)} \ \forall i \in \mathcal{N}, j \in \mathcal{M}', t \in \mathcal{T}\}$, (9b) requires the tasks should be completed within the deadline, (9c) states that each SBS can only provide offloading services to UDs in its coverage area, and (9d) and (9e) ensure that each task is processed on either the local device or one unique SBS.

To remove the effect of stochastic environment behaviors on offloading performance, we introduce the DRL algorithm for making the offloading decision according to the observation of current system state. Furthermore, we propose to adopt an federated learning-inspired framework to enhance the privacy of UDs and reduce the communication overhead during model training. The federated learning process consumes a certain amount of energy and time and hence may compete with task processing for time and resource.

B. Federated Learning Model

In time slot t, the learning process can be divided into three phases, including the distribution of global model from the MBS to UDs, the local model training at UDs, and the federated averaging at the MBS. Here, the global model distribution is neglected due to the high rate of downlink communication [39].

1) Local Model Training: Each UD $i \in \mathcal{N}$ is able to update their local model based on its own dataset Ψ_i. Specifically, we define $z_i^{(t)}$ as a training participation indicator, where $z_i^{(t)} = 1$ if UD i is involved in the training process in time slot t and $z_i^{(t)} = 0$ otherwise. For the model training at UD $i \in \mathcal{N}$ in each time slot $t \in \mathcal{T}$, we randomly sample a mini-batch $\tilde{\Psi}_i^{(t)}$ with $h_i^{(t)}$ data elements from the local dataset Ψ_i of UD i. Generally, the computational complexity of a DRL model is measured based on its requirement of CPU cycles [17]. Accordingly, the delay and energy consumed by the local model training at UD i in time slot t are calculated as

$$T_i^{\mathrm{fc}}(t) = \frac{z_i^{(t)} \varphi_i h_i^{(t)}}{c_i^u} \ \forall i \in \mathcal{N}, t \in \mathcal{T} \tag{10}$$

$$E_i^{\mathrm{fc}}(t) = z_i^{(t)} \kappa \left(c_i^u\right)^2 \varphi_i h_i^{(t)} \ \forall i \in \mathcal{N}, t \in \mathcal{T} \tag{11}$$

respectively, where φ_i is the total amount of CPU cycles needed for training a data sample at UD i.

2) Federated Averaging: After completing the local model training on mini-batch $\tilde{\Psi}_i^{(t)}$, UD i needs to transmit its updated model $\boldsymbol{\theta}_i^{(t)}$ to the centralized server associated with the MBS for federated averaging. The uplink transmission rate from UD i to the MBS is defined as

$$r_i^m = B_i^m \log_2 \left(1 + \frac{P_i^u \varrho_i^m(t)}{B_i^m N_0}\right) \ \forall i \in \mathcal{N}, t \in \mathcal{T} \tag{12}$$

where $\varrho_i^m(t)$ denotes the channel gain between UD i and the MBS. Then, the time and energy consumed by the uplink transmission of UD i's model parameters in time slot t can be computed as

$$T_i^{\mathrm{fu}}(t) = \frac{z_i^{(t)} |\boldsymbol{\theta}_i^{(t)}|}{r_i^m} \ \forall i \in \mathcal{N}, t \in \mathcal{T} \tag{13}$$

$$E_i^{\mathrm{fu}}(t) = z_i^{(t)} P_i^u T_i^{\mathrm{fu}}(t) \ \forall i \in \mathcal{N}, t \in \mathcal{T} \tag{14}$$

where $|\boldsymbol{\theta}_i^{(t)}|$ denotes the data size of the model parameters. Here, the delay resulting from aggregating the received model parameters on the MBS server is omitted due to the superior computation capabilities of the MBS server. Accordingly, the

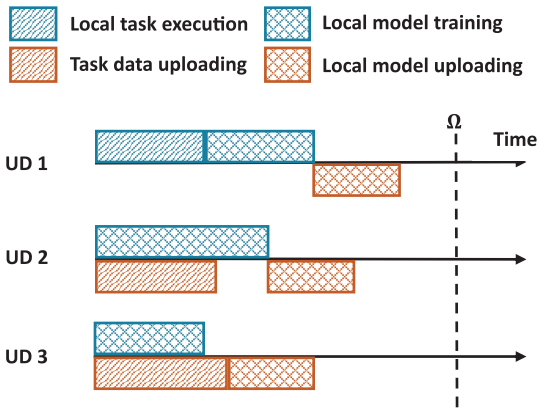

Fig. 2. Temporal relations between task offloading and federated learning on the UDs.

overall time and energy consumed by UD i for model training in time slot t can be given by

$$T_i^{\mathrm{f}}(t) = T_i^{\mathrm{fc}}(t) + T_i^{\mathrm{fu}}(t) \ \forall i \in \mathcal{N}, t \in \mathcal{T} \tag{15}$$

$$E_i^{\mathrm{f}}(t) = E_i^{\mathrm{fc}}(t) + E_i^{\mathrm{fu}}(t) \ \forall i \in \mathcal{N}, t \in \mathcal{T}. \tag{16}$$

C. Joint Task Offloading and Federated Learning Optimization

In this article, we assume a sequential order of the task offloading process and the federated learning process in each time slot; that is, each UD first executes its task based on the offloading decision and then proceeds with model training if it is identified for the learning process. In this scenario, computation or communication resources on UDs are allocated to the learning process only when they are not required for task processing. Fig. 2 illustrates three different cases of temporal relations between task offloading and federated learning. For instance, for UD 1, which opts for local task execution, local model training can only commence after task completion. In contrast, for UDs 2 and 3, which choose to offload tasks, local model training is not restricted by task computation; however, the updated local model can only be uploaded after the task data has been fully uploaded. Depending on the offloading decisions, the total time required to complete both task execution and model training is calculated as

$$T_i^{\mathrm{total}}(t) = u_{i0}^{(t)}\left(T_{i0}^{\mathrm{oc}}(t) + T_i^{\mathrm{f}}(t)\right) + \sum_{j\in\mathcal{M}} u_{ij}^{(t)}\left(\max\left(T_{ij}^{\mathrm{ou}}(t), T_i^{\mathrm{fc}}(t)\right) + T_i^{\mathrm{fu}}(t)\right). \tag{17}$$

The energy consumed for accomplishing both task execution and model training is

$$E_i^{\mathrm{total}}(t) = E_i^{\mathrm{o}}(t) + E_i^{\mathrm{f}}(t). \tag{18}$$

Research [19] has demonstrated the great relationship between the training dataset size and the model performance, that is, the model performance increases with the training data volume until arriving at a specific point. Besides, some clients' long absence from model aggregation will result in incomplete information collection and cause a degraded performance of the aggregated global model. Hence, we specify the minimum training data amount $h^{\min}$ and $H^{\min}$ from the local and global perspectives, respectively, to optimize the model accuracy more effectively and efficiently. Accordingly, we build the joint task offloading and federated learning optimization problem as follows:

$$\min_{\boldsymbol{u},\boldsymbol{z},\boldsymbol{h}} \ \sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{N}} E_i^{\mathrm{total}}(t) \tag{19a}$$

$$\text{s.t.} \ \ T_i^{\mathrm{total}}(t) \le \Omega \ \forall i \in \mathcal{N}, t \in \mathcal{T} \tag{19b}$$

$$\sum_{k=t-\phi+1}^{t} z_i^{(k)} h_i^{(k)} \ge h^{\min} \ \forall i \in \mathcal{N}, t \in \mathcal{T} \tag{19c}$$

$$\sum_{i\in\mathcal{N}} z_i^{(t)} h_i^{(t)} \ge H^{\min} \ \forall i \in \mathcal{N}, t \in \mathcal{T} \tag{19d}$$

$$z_i^{(t)} \in \{0, 1\} \ \forall i \in \mathcal{N}, t \in \mathcal{T} \tag{19e}$$

$$(9b)-(9e) \tag{19f}$$

where $\boldsymbol{z} = \{z_i^{(t)} \ \forall i \in \mathcal{N}, t \in \mathcal{T}\}$ and $\boldsymbol{h} = \{h_i^{(t)} \ \forall i \in \mathcal{N}, t \in \mathcal{T}\}$, (19b) requires both offloading process and learning process should be accomplished before the end of the current time slot, (19c) indicates the minimum amount of local training data during the final ϕ time slots for each UD, and (19d) denotes the minimum amount of global training data in each time slot to guarantee the global model accuracy. With (19c) and (19d), the training volume of each UD can be adjusted according to the change of task requirements in a real-time manner, which in turn facilitates high model accuracy and low energy cost.

Complexity Analysis: The formulated (19) is a mixed-integer nonlinear programming (MINLP) problem involving decision variables $\boldsymbol{u}, \boldsymbol{z}$, and $\boldsymbol{h}$. Solving this problem is challenging due to its exponentially large combinatorial search space. Furthermore, the optimization variables are interdependent: task offloading decisions $\boldsymbol{u}$ influence both the federated learning participation $\boldsymbol{z}$ and local training data amounts $\boldsymbol{h}$, while the outcomes of federated learning, in turn, impact the optimal task offloading strategy. This mutual dependence among the variables significantly increases the complexity of the problem.

Theorem 1: The joint task offloading and federated learning optimization problem in (19) is NP-hard.

Proof: Problem (19) can be shown to be NP-hard by extending the classical 0-1 Knapsack Problem with additional constraints. A formal proof is provided in Appendix A, demonstrating the reduction from the Knapsack Problem and establishing the NP-hardness of (19). ■

Based on Theorem 1, (19) is NP-hard, meaning that finding an optimal solution in polynomial time is infeasible unless P equals NP. To efficiently achieve near-optimal solutions within a reasonable time frame, we propose a DRL-based approach.

IV. Proposed Solutions

Leveraging the federated framework, each UD functions as a decision-making agent that interacts with the MEC environment. By defining the state space, action space, and cost function for each UD, we can formulate the joint offloading and federated learning problem for each UD as an MDP.

To handle the scalability problem of large-scale systems and the lack of prior information, we adopt DQN, a DRL-based method, which uses a deep neural network to explore the optimal state-action value (i.e., Q-value) for the formulated MDP problem [13]. Each UD trains its own DQN model parameters individually on its local data set and learns the data characteristics of other UDs via the parameter aggregation on the MBS server.

A. State, Action, and Cost

1) State Space: Each UD $i \in \mathcal{N}$ can observe its own state information, including the task characteristics and history record of effective training volume at the beginning of each time slot. As required in (19b), each UD i participating in the model training with $z_i^{(t)} = 1$ should complete the predetermined training volume and model uploading within time slot t. Otherwise, the training effort is futile since the local model can not be uploaded in time for aggregation. Accordingly, we denote the effective training sample size of UD i in time slot t as $\bar{h}_i^{(t)} = h_i^{(t)} \mathbb{1}(T_i^{\text{total}}(t) \leq \Omega)$. Define the set of available SBSs for UD i in time slot t as $\mathcal{M}_i^{(t)} = \{j \in \mathcal{M} | d_{ij}^{(t)} \leq O_j^s\}$. Each UD i is able to observe the resource information of SBSs in $\mathcal{M}_i^{(t)}$ and the MBS. Consequently, the state for observation by UD i in time slot t is described as

$$\boldsymbol{s}_i^{(t)} = \left(Task_i^{(t)}, \bar{\boldsymbol{h}}_i^{\phi-1}(t), R_i^{(t)}\right) \tag{20}$$

where $\bar{\boldsymbol{h}}_i^{\phi-1}(t) = \{\bar{h}_i^{(t-\phi+1)}, \ldots, \bar{h}_i^{(t-1)}\}$ denotes the history of UD i's effective training volume over the previous $\phi - 1$ time slots and $R_i^{(t)} = (c_i^u, \{c_j^s, B_{ij}^s\}_{j \in \mathcal{M}_i^{(t)}}, B_i^m)$ encapsulates the available resource information in time slot t.

2) Action Space: Based on the observed state, each UD i chooses where to execute $Task_i^{(t)}$ from the set of available places $\mathcal{M}_i'^{(t)} = 0 \cup \mathcal{M}_i^{(t)}$. For each $j \in \mathcal{M}' \setminus \mathcal{M}_i'^{(t)}$, we have $u_{ij}^{(t)} = 0$. Meanwhile, each UD should also decide whether to participate in the learning process in time slot t, i.e., $z_i^{(t)}$, and further determine the corresponding training batch size, i.e., $h_i^{(t)}$. Here, we set the training batch size $h_i^{(t)}$ to be chosen from a discrete set $\mathcal{H} \triangleq \{0, h_1, h_2, \ldots, h_o\}$ with $|\mathcal{H}|$ optional values to reduce the complexity of action space. Specifically, $h_i^{(t)} = 0$ indicates UD i is pointedly absent from the model training in time slot t, i.e., $z_i^{(t)} = 0$; otherwise, $h_i^{(t)} > 0$. Then, the action vector of UD i in time slot t is given by

$$\boldsymbol{a}_i^{(t)} = \left(\boldsymbol{u}_i^{(t)}, h_i^{(t)}\right). \tag{21}$$

The action space available to UD i can be obtained as $\mathcal{A}_i^{(t)} = \{0, 1\}^{|\mathcal{M}_i'^{(t)}|} \times |\mathcal{H}|$.

3) Cost Function: Recall that our main objective is to minimize the energy cost incurred from task processing and model training, as shown in (19). Furthermore, the constraints in (19) need to be guaranteed. Among these constraints, the satisfaction of (9c)–(9e) and (19e) can be ensured by restricting the action space for each UD. Unfortunately, each UD is not accessible to the states and actions of other UDs whereas the performance of its task processing and model training can be affected by the other UDs' actions.

According to the history value record $\bar{\boldsymbol{h}}_i^{\phi-1}(t)$ and the current value $\bar{h}_i^{(t)}$ in time slot t, Each UD i can calculate the total effective training volume during the past ϕ time slots as $\bar{h}_i^{\phi}(t) = \sum_{k=t-\phi+1}^{t} \bar{h}_i^{(k)}$. Let $\bar{\mathcal{N}}^{(t)} = \{i \in \mathcal{N} | z_i(t) = 1, T_i^{\text{total}}(t) \leq \Omega\}$ denote the UDs participating in the training process and completing the local model training and uploading within time slot t. Each UD in $\bar{\mathcal{N}}^{(t)}$ sends its value of $h_i(t)$ along with its updated model to the MBS.The MBS calculates the effective global training volume in time slot t as $\bar{H}^{(t)} = \sum_{i \in \bar{\mathcal{N}}^{(t)}} h_i^{(t)}$ and signals each UD a failure if the volume fails to meet the requirement. Besides, each UD receives a failure response from its tasks that cannot be completed by the offloading SBS within the deadline. In case of a violation of the constraints with respect to latency or training volume, we define a personalized cost function for each UD i as follows:

$$F_i^{(t)} = E_i^{\text{total}}(t) + E_i^{\text{o}}(t)\Big(\xi_i^d \mathbb{1}\Big(T_i^{\text{o}}(t) > T_{i,\max}^{(t)}\Big) + \xi_i^H \mathbb{1}\Big(\bar{H}^{(t)} < H^{\min}\Big) + \xi_i^h \mathbb{1}\Big(\bar{h}_i^{\phi}(t) < h^{\min}\Big)\Big) \tag{22}$$

where ξ_i^d, ξ_i^H, and $\xi_i^h > 0$ are the three weight factors. Here, we adopt UD i's TO energy consumption $E_i^{\text{o}}(t)$ as the benchmark value of penalty terms since the violation should be punished more severely for causing higher energy consumption. Specifically, ξ_i^d represents the task failure penalty imposed when the task of UD i encounters a timeout; ξ_i^H and ξ_i^h describe the penalties for UD i failing to meet the minimum requirements of global and local training volume.

B. Markovian Decision Process Formulation

The joint task offloading and federated learning optimization problem can be approximated as an individual MDP for each UD. The goal of this formulation is to capture the decision-making process for each UD, where decisions on task offloading and participation in federated learning are made at each time step, and the system evolves based on these decisions. In the MDP, the state at time t, denoted as $\boldsymbol{s}_i^{(t)}$, represents the current system state observed by UD i. The action taken by UD i at time t, denoted as $\boldsymbol{a}_i^{(t)}$, is selected from the action space $\mathcal{A}_i^{(t)}$. The state transition probability $p_{\boldsymbol{s}_i^{(t)}\boldsymbol{s}_i^{(t+1)}}(\boldsymbol{a}_i^{(t)})$ governs the transition to the next state $\boldsymbol{s}_i^{(t+1)}$ after executing action $\boldsymbol{a}_i^{(t)}$. Given the state space, action space, the transition model, and cost function, the goal is to find the optimal policy π_i^* that minimize the cumulative cost. The objective function for each UD is

$$\pi_i^* = \arg\min_{\pi_i} \mathbb{E}\left[\sum_{t \in \mathcal{T}} \gamma F\left(\boldsymbol{s}_i^{(t)}, \boldsymbol{a}_i^{(t)}\right)\right] \tag{23}$$

where $\gamma \in [0, 1)$ is the discount factor that balances immediate and future rewards.

In this study, we adopt the DQN approach to achieve the optimization goals in (23). The decision to use DQN stems from its ability to efficiently handle discrete action spaces, which aligns with the decision-making process for UDs. As demonstrated in [38], DQN performs exceptionally well in high-dimensional environments similar to our study. Additionally, compared to policy-based methods like proximal

policy optimization (PPO) or actor–critic methods, DQN's off-policy training and experience replay allow for efficiently reuse of past experiences, thus reducing the number of required training episodes and, by extension, the computational and energy costs [43]. This makes DQN well-suited to our goal of optimizing both performance and energy efficiency. We recognize DQN's limitation of overestimation bias in Q-value estimation. While Double DQN can mitigate this issue, we prioritize DQN for its computational simplicity and favorable tradeoff between performance and efficiency, particularly with our emphasis on energy reduction. As noted in [43], DQN consistently delivers high performance while consuming less energy compared to more complex methods.

C. DQN-Based Solution

Each UD $i \in \mathcal{N}$ maintains an experience replay buffer Ψ_i with capacity ζ_i to store the observed transition experience $\psi_i^{(t)} = (\boldsymbol{s}_i^{(t)}, \boldsymbol{a}_i^{(t)}, F_i^{(t)}, \boldsymbol{s}_i^{(t+1)})$ in each time slot $t \in \mathcal{T}$. Note the experience replay buffer here is actually equivalent to the local dataset mentioned in Section III-B1. Thus, we use the same notation here to facilitate reading. Each UD $i \in \mathcal{N}$ adopts two neural networks consisting of an evaluation network $EvaNet_i$ and a target network $TarNet_i$. These two networks own the identical network structure but individual network parameter vector which we denote by $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_i'$ for $EvaNet_i$ and $TarNet_i$, respectively. In DQN, $EvaNet_i$ is utilized to choose actions, while $TarNet_i$ is for generating a target Q-value which estimates the expected long-term cost of selecting a state-action pair.

Let $Q_i(\boldsymbol{s}_i^{(t)}, \boldsymbol{a}_i^{(t)}; \boldsymbol{\theta}_i^{(t)})$ and $Q_i(\boldsymbol{s}_i^{(t)}, \boldsymbol{a}_i^{(t)}; \boldsymbol{\theta}_i')$ denote the Q-value of $EvaNet_i$ and $TarNet_i$ under state-action pair $(\boldsymbol{s}_i^{(t)}, \boldsymbol{a}_i^{(t)})$, respectively. According to the observed current state $\boldsymbol{s}_i^{(t)}$, UD i will select its action according to the following ϵ-greedy rule:

$$\boldsymbol{a}_i^{(t)} = \begin{cases} \text{a random chosen action}, & \text{with probab. } \epsilon, \\ \arg\min\limits_{\boldsymbol{a}' \in \mathcal{A}_i} Q_i\left(\boldsymbol{s}_i^{(t)}, \boldsymbol{a}'; \boldsymbol{\theta}_i^{(t)}\right), & \text{with probab. } 1-\epsilon \end{cases} \tag{24}$$

where ϵ indicates the probability to explore a random solution instead of greedy exploitation. At the beginning of the next time slot, UD i can observe the cost $F_i^{(t)}$ and next state $\boldsymbol{s}_i^{(t+1)}$. Then, UD i stores this corresponding experience $(\boldsymbol{s}_i^{(t)}, \boldsymbol{a}_i^{(t)}, F_i^{(t)}, \boldsymbol{s}_i^{(t+1)})$ in its replay buffer.

1) Distributed Training at User Device $i \in \mathcal{N}$: In each time slot t, each UD $i \in \mathcal{N}$ chooses whether or not to participate in the federated training. Given an experience $(\boldsymbol{s}_i^{(t)}, \boldsymbol{a}_i^{(t)}, F_i^{(t)}, \boldsymbol{s}_i^{(t+1)})$ from the sampled mini-batch $\tilde{\Psi}_i^{(t)}$, the parameter vector $\boldsymbol{\theta}_i$ of $EvaNet_i$ are updated to minimize the difference between the Q-value predicted by $EvaNet_i$ and the target Q-value obtained by $TarNet_i$. Precisely, UD i trains its $EvaNet_i$ via iteratively updating the parameter vector $\boldsymbol{\theta}_i$ to minimize the loss function as follows:

$$L_i\left(\boldsymbol{\theta}_i^{(t)}\right) = \mathbb{E}\left[\left(Q_i\left(\boldsymbol{s}_i^{(t)}, \boldsymbol{a}_i^{(t)}; \boldsymbol{\theta}_i^{(t)}\right) - \hat{Q}_{i,t}^{\text{Target}}\right)^2\right] \tag{25}$$

where $\hat{Q}_{i,t}^{\text{Target}} = F_i^{(t)} + \gamma \min_{\boldsymbol{a}' \in \mathcal{A}_i^{(t)}} Q_i(\boldsymbol{s}_i^{(t+1)}, \boldsymbol{a}'; \boldsymbol{\theta}_i')$ denotes the estimated long-term cost of action $\boldsymbol{a}_i^{(t)}$ under state $\boldsymbol{s}_i^{(t)}$, i.e., the target Q-value.

Algorithm 1 Proposed JOFO Algorithm

Require: $\boldsymbol{s}_i(t)$
Ensure: $\boldsymbol{u}_i(t), h_i(t)$
1: Initialize $\Psi_i, \boldsymbol{\theta}_i^{(t)}, \boldsymbol{\theta}_i', \boldsymbol{\theta}^{(t)}, \boldsymbol{\theta}'$;
2: **for** $t \in \mathcal{T}$ **do**
3: Download $\boldsymbol{\theta}^{(t)}$ and $\boldsymbol{\theta}'$ from the MBS server, and set $\boldsymbol{\theta}_i^{(t)} \leftarrow \boldsymbol{\theta}^{(t)}$ and $\boldsymbol{\theta}_i' \leftarrow \boldsymbol{\theta}'$;
4: Choose an action $\boldsymbol{a}_i^{(t)} = \left(\boldsymbol{u}_i^{(t)}, h_i^{(t)}\right)$ according to (24) for the observed state $\boldsymbol{s}_i^{(t)}$;
5: Offload the task according to $\boldsymbol{u}_i^{(t)}$;
6: **if** $h_i^{(t)} > 0$ **then**
7: Sample a mini-batch $\tilde{\Psi}_i^{(t)}$ of $h_i^{(t)}$ data elements from Ψ_i;
8: Calculate $L_i\left(\boldsymbol{\theta}_i^{(t)}\right)$ as (25), and update $\boldsymbol{\theta}_i^{(t+1)}$ as (26).
9: **if** UD i completes the training data amount $h_i(t)$ within time slot t **then**
10: Upload the updated $\boldsymbol{\theta}_i$ to the MBS server;
11: **end if**
12: **end if**
13: Observe the cost $F_i^{(t)}$ and the next state $\boldsymbol{s}_i^{(t+1)}$;
14: Save experience $\left(\boldsymbol{s}_i^{(t)}, \boldsymbol{a}_i^{(t)}, F_i^{(t)}, \boldsymbol{s}_i^{(t+1)}\right)$ in Ψ_i;
15: Update $\boldsymbol{\theta}^{(t+1)}$ via federated averaging as (27).
16: **if** $t \mod \tau = 0$ **then**
17: Update the global target network $\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta}^{(t)}$
18: **end if**
19: **end for**

In this work, we exploit the stochastic gradient descent (SGD) method for local model training aiming at empirical risk minimization [44]. The corresponding update results of $EvaNet_i$ of UD i after the model training on mini-batch $\tilde{\Psi}_i^{(t)}$ can be defined as

$$\boldsymbol{\theta}_i^{(t+1)} = \boldsymbol{\theta}_i^{(t)} - \eta g_i^{(t)} \tag{26}$$

where η is the learning rate and $g_i^{(t)} = \nabla L_i(\boldsymbol{\theta}_i^{(t)}; \tilde{\Psi}_i^{(t)})$ is the stochastic gradient.

2) Model Aggregation and Distribution at the MBS Server: In the end of each time slot t, the MBS server receives the parameter update results from the UDs in set $\tilde{\mathcal{N}}^{(t)}$. Then, the MBS updates the global model based on aggregated update $\Delta^{(t)}$ considering batch size ratios $w_i^{(t)}$ as follows:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta \Delta^{(t)} \tag{27}$$

where $\Delta^{(t)} = \sum_{i \in \tilde{\mathcal{N}}^{(t)}} w_i^{(t)} g_i^{(t)}$ with $w_i^{(t)} = [(h_i^{(t)})/(\sum_{k \in \tilde{\mathcal{N}}^{(t)}} h_k^{(t)})]$. Every τ time slots, the parameter vector $\boldsymbol{\theta}^{(t)}$ of the global evaluation network is duplicated to replace the parameter vector $\boldsymbol{\theta}'$ of the global target network.

The proposed joint offloading decision and federated learning optimization (JOFO) algorithm is depicted in Algorithm 1. Fig. 3 presents an intuitive view of the scheme designed for solving the formulated joint offloading and learning optimization problem. In detail, each UD, acting as a federated optimizer, maintains a local DQN model where a target Q-network is used to assist a main Q-network in learning

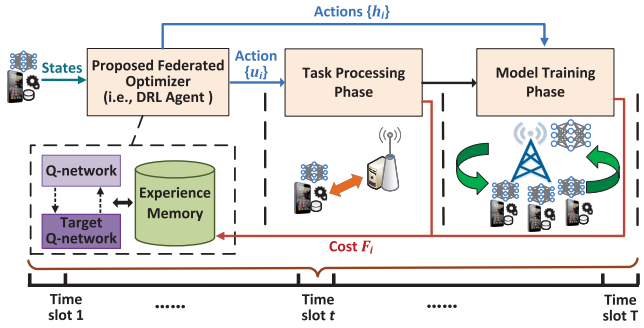

Fig. 3. Scheme for the joint task offloading and federated learning optimization.

from historical data stored in the experience memory. At the beginning of the t-th time slot, each UD acquires the global evaluation network $\boldsymbol{\theta}^{(t)}$ and the global target network $\boldsymbol{\theta}'$ from the MBS server to replace its local corresponding network parameters $\boldsymbol{\theta}_i^{(t)}$ and $\boldsymbol{\theta}_i'$. Based on the collected current state information $\boldsymbol{s}_i^{(t)}$, including task demands and training records, the local model draws an action $\boldsymbol{a}_i^{(t)} = (\boldsymbol{u}_i^{(t)}, h_i^{(t)})$ for the joint offloading and training decision. Then, the task is executed locally or offloaded to the SBS following the offloading decision $\boldsymbol{u}_i^{(t)}$. The local model training later starts if the local computation resources are not occupied by the corresponding task processing process. Only when the model training on the mini-batch with $h_i^{(t)}$ data elements is accomplished within the given time slot can the updated local model be transmitted to the MBS for federated averaging. At the end of each time slot, the MBS server conducts the aggregation on the models which are successfully uploaded in time. At the beginning of time slot $t+1$, each UD can receive the cost $F_i^{(t)}$ and observe the next state information $\boldsymbol{s}_i^{(t+1)}$, which are saved in a new experience $(\boldsymbol{s}_i^{(t)}, \boldsymbol{a}_i^{(t)}, F_i^{(t)}, \boldsymbol{s}_i^{(t+1)})$ for later training.

D. Convergence Analysis

JOFO is a federated DQN-based algorithm that enables UDs to act as federated clients, collaboratively learning optimal policies. We recognize that handling nonindependent and identically distributed (non-IID) data across UDs poses significant challenges in federated DQN, affecting both convergence speed and stability. In this work, we implement strategies to balance client contributions during model aggregation, mitigating the impact of data heterogeneity. Specifically, we enforce constraints on minimum global training data volume as in (19d) and minimum local training data volume as in (19c) to ensure balanced data distribution and sufficient client participation in each round. Additionally, we adjust aggregation weights based on clients' batch sizes during global model updates, as shown in (27). In this section, we provide a rigorous convergence proof of JOFO which explicitly considers data heterogeneity by introducing a deterministic bound on client drift, denoted by δ, as follows.

Proposition 1: Under the assumptions specified in Appendix B, the sequence $\{\theta^{(t)}\}$ generated by JOFO satisfies

$$\limsup_{t\to\infty} \|\nabla L\left(\theta^{(t)}\right)\| \leq \delta$$

where $\nabla L(\theta^{(t)}) = \sum_{i=1}^{N} p_i \nabla L_i(\theta^{(t)})$ is the gradient of the global loss function at iteration t, p_i are the weights (e.g., based on data sizes) satisfying $\sum_{i=1}^{N} p_i = 1$, and $\delta \geq 0$ is the deterministic bound on client drift defined in Assumption 6 in Appendix B.

Proof: See Appendix B. ■

This analysis implies that JOFO converges to a neighborhood of a stationary point, with the size of the neighborhood bounded by δ. The size of this neighborhood is determined by the client drift δ, which encapsulates the effect of partial participation and data heterogeneity among UDs.

TABLE II
SIMULATION PARAMETERS

Parameter	Value
The number of UDs (N)	50
The number of SBSs (M)	5
Task size ($D_i^{(t)}$)	$\{2.0, 2.1, \cdots, 5.0\}$ Mbits
Task computation ($C_i^{(t)}$)	0.4 gigacycles per Mbits
Task latency ($T_{i,max}^{(t)}$)	0.3 second
Training batch size ($h_i^{(t)}$)	$\{0, 10, 20, \cdots, 100\}$
Model parameter size ($\|\boldsymbol{\theta}_i^{(t)}\|$)	1 Mbits
Model training (φ_i)	0.01 gigacycles per data sample
Local minimum batch size $h^{\min}$	100
Global minimum batch size $H^{\min}$	$Nh^{\min}/\phi$
CPU frequency of UDs c_i^u	{4,5,6} gigacycles/s
CPU frequency of SBSs c_j^s	90 gigacycles/s
Transmission power of UDs P_i^u	0.2 W
SBS bandwidth B_{ij}^s	5 MHz
MBS bandwidth B_m	10 MHz
Noise power N_0	-176 dbm/Hz
Coefficient κ	10^{-29}
Time interval Ω	0.5 second

V. PERFORMANCE EVALUATION

In this section, we conduct numerical simulations based on Python-PyTorch simulator to evaluate the performance of our proposed method JOFO for joint offloading and learning optimization. We consider a scenario where 50 UDs are randomly scattered in the coverage region of five SBSs. The parameters are set according to [17], [21], and [34] and are listed in Table II. We consider a fully connected deep neural network with one input layer, one output layer, and three hidden layers, where the three hidden layers consist of 64, 128, and 64 hidden neurons, respectively. Additionally, we set the discount factor $\gamma = 0.9$ and employ the Adam optimizer with a learning rate $\eta = 0.001$. The probability of random exploration ϵ gradually decreases from 1 to 0.01.

Fig. 4 depicts the convergence behavior of JOFO given different training intensity requirements, i.e., the amount of time slots ϕ assigned to each UD for completing the minimum local training volume. Here, the x-axis represents the time slot and the y-axis denotes the average cost among the UDs in each time slot. In Fig. 4, the average cost obtained by JOFO at the beginning is relatively high since we set the first 100 time slots only for collecting experiences without model training. Once the model training is initiated, the cost starts to gradually

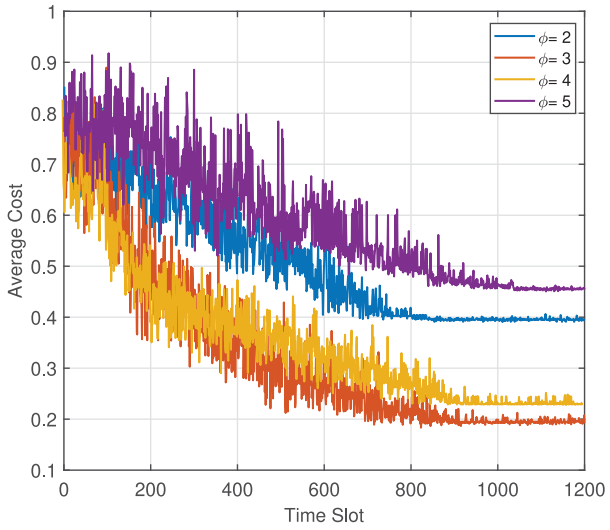

Fig. 4. Convergence behavior of JOFO under different requirements of training intensity.

TABLE III
PERFORMANCE EVALUATION OF PENALTY WEIGHTS

ξ_i^d	ξ_i^H	ξ_i^h	Total EC (J)	Offloading EC (J)	Learning EC (J)	Task Failure Rate (%)
0	0	0	0.130	0.082	0.048	22.68
1	0	0	0.161	0.097	0.064	12.71
0	1	0	0.175	0.074	0.102	21.89
0	0	1	0.176	0.069	0.107	23.43
1	1	0	0.216	0.105	0.111	7.54
1	0	1	0.219	0.103	0.116	7.21
0	1	1	0.175	0.056	0.119	24.36
1	1	1	0.237	0.109	0.128	7.07
2	1	1	0.238	0.113	0.125	4.55
1	2	1	0.244	0.101	0.143	5.52
1	1	2	0.261	0.106	0.155	5.63

decrease and flattens out over time. Setting an appropriate value of ϕ can help the UDs effectively control the training cost while ensuring the training performance. When $\phi = 5$, the convergence speed of JOFO is the slowest due to the low training intensity for DRL models. Although the energy consumed by model training is saved, the under-trained models may lead to offloading decisions with poor performance in energy saving and QoS guarantee. When $\phi = 2$, the high training intensity enables a fast convergence speed of JOFO, which however consumes excessive energy for model training. The training cost and model performance can be well balanced when $\phi = 3$, since JOFO can converge to the lowest average cost at a fast and stable speed.

To evaluate the effect of the penalty weights, i.e., ξ_i^d, ξ_i^H and ξ_i^h in (22), we evaluate the performance of JOFO given different values of these penalty weights. Table III displays the average value of total energy consumption (Total EC), offloading energy consumption (Offloading EC), learning energy consumption (Learning EC) and task failure rate among the UDs and time slots under different penalty weights ξ_i^d, ξ_i^H, and ξ_i^h. From the results, we can observe that the enforcement of the task latency constraints by setting $\xi_i^d > 0$ leads to a reduced ratio of overdue tasks but an increased offloading energy cost. Without the limitation of latency constraints, all

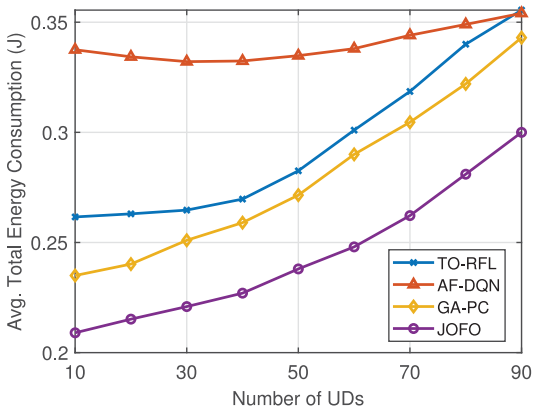

Fig. 5. Average total EC of different methods under different number of UDs.

UDs tend to transfer the task to the SBS instead of performing the task locally to save energy. In such case, the limited computation capacity of SBS servers is unable to complete such huge amount of tasks simultaneously and therefore causes lots of failed tasks. Besides, imposing the training volume constraints by setting $\xi_i^H > 0$ or $\xi_i^h > 0$ ensures the models get adequate training for achieving a reduced level of Offloading EC at the expense of an increased learning energy cost, as demonstrated in Table III. Although the average Total EC is the lowest when $\xi_i^d, \xi_i^H, \xi_i^h = 0$, the task failure rate is too high to guarantee a desired QoS level. It can be seen that JOFO can explore a reasonable tradeoff between the energy consumption and the task failure rate when $\xi_i^d, \xi_i^H, \xi_i^h = 2, 1, 1$. Therefore, we follow this ratio to conduct the subsequent evaluation.

We compare the performance of our presented method JOFO with several methods, including task offloading based on random federated learning scheduling (denoted by TO-RFL), AF-DQN in [38], and GA-PC in [17]. TO-RFL is a benchmark method where each UD makes the offloading decision based on the local DQN model and samples a random amount of data elements to train its local model. AF-DQN is a federated DQN-based task offloading algorithm that sets a fixed time interval for federated clients to asynchronously upload local models to the centralized server. GA-PC is a genetic algorithm-based method to optimize the energy consumption and latency of the joint learning and offloading process. Both AF-DQN and GA-PC employ a federated mechanism to make task offloading decision and consider the optimization of the model training process. However, AF-DQN ignored the training cost while GA-PC neglected the dynamic interaction between offloading decision and model training. We evaluate the impact of a variable number of UDs (ranging from 10 to 90) on the energy consumption, the ratio of failed tasks, and the ratio of under-trained local and global training times.

Fig. 5 demonstrates the average Total EC among the UDs and the time slots given different number of UDs. As shown in Fig. 5, as the amount of UDs increases, the average total energy consumed by all the methods except AF-DQN gradually grows. In contrast, the average Total EC of AF-DQN decreases as the number of UDs increases from 10 to 30 and then ascends slightly with a rising number of UDs. Despite

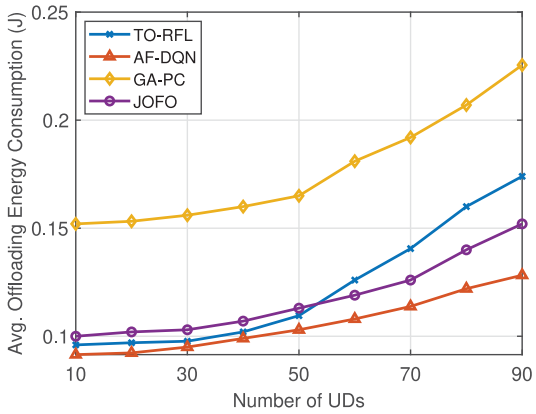

Fig. 6. Average task offloading EC of different methods under different number of UDs.

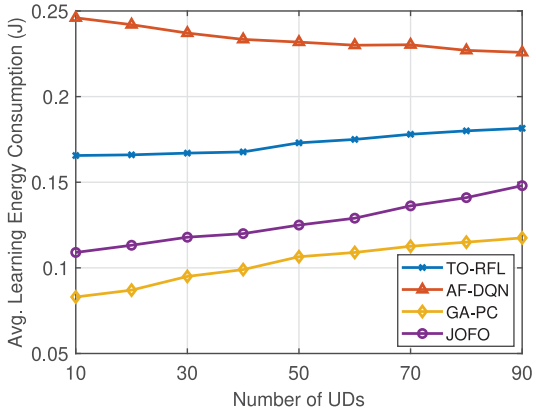

Fig. 7. Average federated learning EC of different methods under different number of UDs.

the minimum increasing rate among all the methods, AF-DQN always consumes the largest amount of energy except when the number of UDs increases to 90. On the contrary, JOFO keeps yielding the lowest Total EC, which proves JOFO's remarkable efficiency at conserving energy. When the number of UDs is 50, JOFO reduces the average Total EC by 12.34%–28.92% compared with the other methods. As stated in (18), the Total EC results from two processes, i.e., the task processing and the federated learning. To further prove that JOFO can effectively juggle the energy-saving improvements of the offloading and learning processes, we further evaluate the energy performance of all the methods in these two processes, respectively, as shown in Figs. 6 and 7.

From Fig. 6, it can be noticed that the average Offloading EC of all the methods rises with a growing number of UDs. This is because heavier task workload strains the limited bandwidth resources, resulting in an increase in data transmission time and task uploading energy. Moreover, confronted with the expanding strain on SBSs' bandwidth and computation resources, more UDs have to choose to conduct the task locally to satisfy their task delay requirements. Among all the methods, AF-DQN consumes the least amount of energy for task offloading, while GA-PC uses the greatest. Besides, the average Offloading EC of JOFO approaches that of AF-DQN. The offloading energy cost depends on the quality of the offloading decision made by the local DRL models and thus reflects the accuracy performance of DRL models. Therefore, it can be derived that both AF-DQN and JOFO can effectively train accurate DRL models for making high-quality offloading solutions.

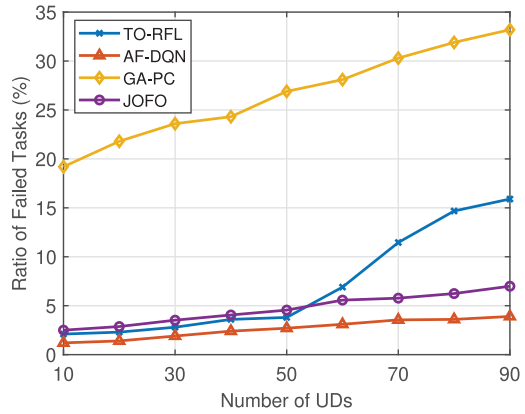

Fig. 8. Ratio of failed tasks of different methods under different number of UDs.

Fig. 7 depicts the average Learning EC of all the methods under a variable number of UDs. AF-DQN adopts a greedy training mechanism without the capacity of training volume regulation and sets each UD to train its local DRL model when the local computing resources are available. Over-committing local computing resources can result in high computation energy cost, which accounts for the high Learning EC of AF-DQN. Conversely, GA-PC generates the lowest Learning EC, which implies GA-PC's low-volume training manner. However, combining the high Offloading EC of GA-PC in Fig. 6, we can infer that the GA-based solution to learning process time leads to under-trained DRL models in GA-PC which result in low-quality offloading decisions. The proposed JOFO achieves a reasonable tradeoff between the model accuracy and the training cost, and thus obtains a low-energy solution of offloading decision at a low learning energy cost.

Fig. 8 demonstrates the ratio of tasks which are not completed within the required task latency under different amount of UDs. As shown in Fig. 8, both AF-DQN and JOFO obtain a slightly increased ratio of failed tasks with a growing number of UDs. This is primarily due to the informed offloading decision making enabled by well-trained models in AF-DQN and JOFO. As the number of UDs increases from 10 to 90, the ratio of failed tasks of JOFO keeps less than 7%, while that of TO-RFL rises to more than 15%. This emphasizes the significance of training volume adjustment for guaranteeing the QoS requirements of UDs especially under a heavy and unknown dynamic system workload.

Figs. 9 and 10 plot the ratio of time slots in which the local and global training volume constraints are violated, respectively. Among all the methods, AF-DQN and JOFO maintain a low ratio of under-trained local and global training times as shown in Figs. 9 and 10 respectively. This demonstrates that the models in AF-DQN and JOFO get adequate training and thus are well equipped to make wise joint offloading and

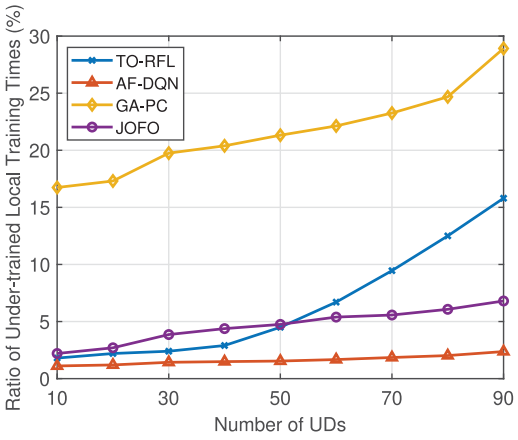

Fig. 9. Ratio of under-trained local training times of different methods under different number of UDs.

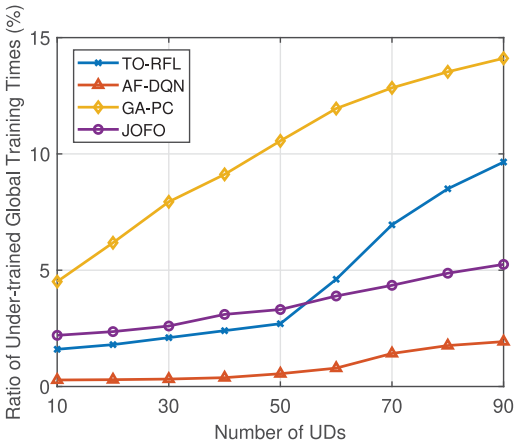

Fig. 10. Ratio of under-trained global training times of different methods under different number of UDs.

learning decisions. While AF-DQN ensures the effective training volume by its greedy training principle, JOFO adaptively adjusts the size of training dataset according to the dynamic workload to achieve effective and efficient model training. Notably, the ratio of under-trained times obtained in TO-RFL increases dramatically when the number of UDs exceeds 50. This is because the limited MBS bandwidth cannot support a large number of UDs to upload the trained model to the MBS in time. The ineffective local training not only wastes a lot of energy but also degrades the training performance, leading the DRL model on a downward spiral with more irrational joint offloading and learning decisions being made. In conjunction with the performance results given above, we can infer that rational decision making for the joint offloading and learning processes helps the DRL models reach a greater energy-related and QoS-related performance, which is exactly the key strength of our proposed method JOFO. Considering the cost of task processing and model training, JOFO leverages DRL models to adaptively make high-quality joint offloading and learning decisions based on the task information and training history record, which in turn facilitate efficient model training and improve the quality of the DRL models' decision making.

VI. Conclusion

In this work, we studied the energy-aware federated task offloading mechanism, where the UDs' energy consumption incurred by task processing and federated learning is taken into consideration. We focused on the resource sharing on time and energy between the task offloading process and the federated learning process as well as the significant effect of model training on the performance of offloading decision. We adopted the DQN algorithm in a federated framework to enable each UD to make a high-quality joint offloading and model training decision without requiring the prior information of task requirements and network conditions. Simulation results demonstrated that the proposed method outperforms other existing methods in terms of reducing the UDs' energy consumption by balancing the model performance and training cost. For future work, we will incorporate the energy consumed by edge servers into the energy-efficiency optimization to further improve the energy-saving performance of the whole MEC-based network architecture. In addition, future work will explore communication-efficient strategies, like model update compression, to reduce communication costs and enhance JOFO's adaptability in resource-constrained MEC networks. We will also address non-IID data distribution by integrating advanced model aggregation methods, such as personalized federated learning, to improve convergence stability in dynamic MEC environments.

Appendix A
Proof of Theorem 1

To demonstrate the NP-hardness of Problem (19), we consider a simplified version that focuses solely on federated learning optimization in a single time slot ($T = 1$). We assume that time constraints are satisfied for all UDs and set $h^{\min} = 0$ for simplification. Let v_i represent the training data amount available at UD i. We set the training amount for each UD as $h_i^{(1)} = v_i$. Let α and β denote the energy consumed per data sample during local training and the energy consumed for uploading the model, respectively. The simplified federated learning optimization problem can be formulated as

$$\min_{z^{(1)}} \quad \sum_{i\in\mathcal{N}} z_i^{(1)}(\alpha v_i + \beta) \tag{28a}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{N}} z_i^{(1)} v_i \geq H^{\min}, \tag{28b}$$

$$z_i^{(1)} \in \{0,1\} \quad \forall i \in \mathcal{N}. \tag{28c}$$

Let W' be the minimal Total EC achievable while satisfying the training data requirement $H^{\min}$. That is

$$W' = \min_{z^{(1)}} \left\{ \sum_{i\in\mathcal{N}} z_i^{(1)}(\alpha v_i + \beta) \;\middle|\; \sum_{i\in\mathcal{N}} z_i^{(1)} v_i \geq H^{\min} \right. \\ \left. z_i^{(1)} \in \{0,1\} \right\}. \tag{29}$$

Problem (28) can be transformed into the following equivalent training data maximization problem under the energy constraint W' [45]:

$$\max_{z^{(1)}} \quad \sum_{i\in\mathcal{N}} z_i^{(1)} v_i \tag{30a}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{N}} z_i^{(1)} v_i \geq H^{\min} \tag{30b}$$

$$\sum_{i\in\mathcal{N}} z_i^{(1)} (\alpha v_i + \beta) \leq W' \tag{30c}$$

$$z_i^{(1)} \in \{0, 1\}. \tag{30d}$$

Problem (30) is equivalent to the 0-1 Knapsack Problem with an additional minimum value requirement, which remains NP-hard [46]. Specifically, we can regard the N UDs as the items, training data amounts v_i as the values, energy consumptions $\alpha v_i + \beta$ as the weights, and total energy budget W' as the knapsack capacity. Therefore, (30) is NP-hard, and consequently, (28) is NP-hard. Since (28) is a simplified case of (19), we conclude that (19) is NP-hard.

Appendix B
Proof of Proposition 1

In JOFO, SGD is employed to update the local models as in (26), while the global model is updated based on batch size ratios as in (27). Following the frameworks in [47] and [48], we make the following assumptions.

1) Each local loss function $L_i(\theta)$ is L-smooth and lower bounded
$$\|\nabla L_i(\theta) - \nabla L_i(\theta')\| \leq L\|\theta - \theta'\|, \ L_i(\theta) \geq L_{\inf}.$$
2) The stochastic gradients $g_i^{(t)}$ are unbiased estimators of the true gradients with bounded variance
$$\mathbb{E}\left[g_i^{(t)}\right] = \nabla L_i\left(\theta^{(t)}\right), \ \mathbb{E}\left[\|g_i^{(t)} - \nabla L_i\left(\theta^{(t)}\right)\|^2\right] \leq \frac{\sigma_i^2}{h_i^{(t)}}.$$
3) The gradient norms are bounded
$$\|\nabla L_i(\theta)\| \leq G.$$
4) There exist lower bounds on batch sizes for clients participating in the training process
$$h_i^{(t)} \geq \chi > 0 \ \forall i \in \bar{\mathcal{N}}^{(t)}.$$
5) The learning rate η is sufficiently small and may diminish over time.
6) There exists a deterministic bound on client drift
$$\left\|\nabla L\left(\theta^{(t)}\right) - \sum_{i\in\bar{\mathcal{N}}^{(t)}} w_i^{(t)} \nabla L_i\left(\theta^{(t)}\right)\right\| \leq \delta, \ \delta \geq 0.$$

Using Assumption 1 and the standard result for smooth functions from [49], we derive

$$L\left(\theta^{(t+1)}\right) \leq L\left(\theta^{(t)}\right) - \eta \nabla L\left(\theta^{(t)}\right)^\top \Delta^{(t)} + \frac{L\eta^2}{2}\|\Delta^{(t)}\|^2.$$

Taking expectations over the stochastic gradients, we get

$$\mathbb{E}_{\tilde{\Psi}^{(t)}}\left[L\left(\theta^{(t+1)}\right)\right] \leq L\left(\theta^{(t)}\right) - \eta \nabla L\left(\theta^{(t)}\right)^\top \bar{g}^{(t)} + \frac{L\eta^2}{2}\mathbb{E}_{\tilde{\Psi}^{(t)}}\left[\|\Delta^{(t)}\|^2\right]$$

where $\tilde{\Psi}^{(t)} = \{\tilde{\Psi}_i^{(t)}\}_{i\in\bar{\mathcal{N}}^{(t)}}$ is the collection of random variables representing the stochasticity in the gradients at iteration t, and $\bar{g}^{(t)} = \sum_{i\in\bar{\mathcal{N}}^{(t)}} w_i^{(t)} \nabla L_i(\theta^{(t)})$ is the aggregated true gradients.

To bound the term $\nabla L(\theta^{(t)})^\top \bar{g}^{(t)}$, we have

$$\nabla L\left(\theta^{(t)}\right)^\top \bar{g}^{(t)} = \|\nabla L\left(\theta^{(t)}\right)\|^2 - \nabla L\left(\theta^{(t)}\right)^\top \left(\nabla L\left(\theta^{(t)}\right) - \bar{g}^{(t)}\right).$$

By using Assumption 6

$$\left\|\nabla L\left(\theta^{(t)}\right) - \bar{g}^{(t)}\right\| \leq \delta$$

and applying the Cauchy–Schwarz inequality

$$\left|\nabla L\left(\theta^{(t)}\right)^\top \left(\nabla L\left(\theta^{(t)}\right) - \bar{g}^{(t)}\right)\right| \leq \|\nabla L\left(\theta^{(t)}\right)\|\delta$$

we obtain

$$\nabla L\left(\theta^{(t)}\right)^\top \bar{g}^{(t)} \geq \|\nabla L\left(\theta^{(t)}\right)\|^2 - \|\nabla L\left(\theta^{(t)}\right)\|\delta.$$

Using the bounded variance of stochastic gradients (Assumption 2) and the fact that batch sizes are lower bounded (Assumption 4), we can bound the expected squared norm $\mathbb{E}_{\tilde{\Psi}^{(t)}}[\|\Delta^{(t)}\|^2]$ according to standard results in [47] as follows:

$$\mathbb{E}_{\tilde{\Psi}^{(t)}}\left[\|\Delta^{(t)}\|^2\right] \leq (G+\delta)^2 + \frac{\sigma_{\max}^2}{\chi}$$

where $\sigma_{\max}^2 = \max_i \sigma_i^2$ and G is the bound on gradient norms (Assumption 3).

Combining the bounds, we obtain

$$\mathbb{E}_{\tilde{\Psi}^{(t)}}\left[L\left(\theta^{(t+1)}\right)\right] \leq L\left(\theta^{(t)}\right) - \eta\left(\|\nabla L\left(\theta^{(t)}\right)\|^2 - \|\nabla L\left(\theta^{(t)}\right)\|\delta\right) + \frac{L\eta^2}{2}\left((G+\delta)^2 + \frac{\sigma_{\max}^2}{\chi}\right).$$

Summing both sides over $t = 0$ to $T-1$ and using the telescoping sum, we have

$$\eta \sum_{t=0}^{T-1}\left(\|\nabla L\left(\theta^{(t)}\right)\|^2 - \|\nabla L\left(\theta^{(t)}\right)\|\delta\right) \leq L\left(\theta^{(0)}\right) - \mathbb{E}\left[L\left(\theta^{(T)}\right)\right] + T\frac{L\eta^2}{2}\left((G+\delta)^2 + \frac{\sigma_{\max}^2}{\chi}\right).$$

Since $L(\theta^{(T)}) \geq L_{\inf}$, the term $L(\theta^{(0)}) - \mathbb{E}[L(\theta^{(T)})]$ is bounded by $\Delta L = L(\theta^{(0)}) - L_{\inf}$. Dividing both sides by $T\eta$ and taking $T \to \infty$, we find

$$\limsup_{t\to\infty} \|\nabla L\left(\theta^{(t)}\right)\| \leq \delta + \frac{L\eta}{2}\left((G+\delta)^2 + \frac{\sigma_{\max}^2}{\chi}\right).$$

As η is sufficiently small (Assumption 5), the second term becomes negligible, leading to

$$\limsup_{t\to\infty} \|\nabla L\left(\theta^{(t)}\right)\| \leq \delta.$$

REFERENCES

[1] R. Luo, H. Jin, Q. He, S. Wu, and X. Xia, "Cost-effective edge server network design in mobile edge computing environment," *IEEE Trans. Sustain. Comput.*, vol. 7, no. 4, pp. 839–850, Oct.–Dec. 2022.

[2] C. Shang, Y. Sun, H. Luo, and M. Guizani, "Computation offloading and resource allocation in NOMA–MEC: A deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 10, no. 17, pp. 15464–15476, Sep. 2023.

[3] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multiaccess edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1657–1681, 3rd Quart., 2017.

[4] K. Li, X. Wang, Q. He, Q. Ni, M. Yang, and S. Dustdar, "Computation offloading for tasks with bound constraints in multiaccess edge computing," *IEEE Internet Things J.*, vol. 10, no. 17, pp. 15526–15536, Sep. 2023.

[5] C. R. Panigrahi, J. L. Sarkar, B. Pati, R. Buyya, P. Mohapatra, and A. Majumder, "Mobile cloud computing and wireless sensor networks: A review integration architecture and future directions," *Inst. Eng. Technol. Netw.*, vol. 10, no. 4, pp. 141–161, 2021.

[6] L. Gu, W. Zhang, Z. Wang, D. Zeng, and H. Jin, "Service management and energy scheduling toward low-carbon edge computing," *IEEE Trans. Sustain. Comput.*, vol. 8, no. 1, pp. 109–119, Jan.–Mar. 2023.

[7] C. Liu, F. Tang, Y. Hu, K. Li, Z. Tang, and K. Li, "Distributed task migration optimization in MEC by extending multiagent deep reinforcement learning approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1603–1614, Jul. 2021.

[8] J. Zhang et al., "Joint resource allocation for latency-sensitive services over mobile edge computing networks with caching," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4283–4294, Jun. 2019.

[9] Y. S. Patel, M. Reddy, and R. Misra, "Energy and cost trade-off for computational tasks offloading in mobile multi-tenant clouds," *Cluster Comput.,* vol. 24, pp. 1793–1824, Jan. 2021.

[10] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[11] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 2nd ed. Cambridge, MA, USA: MIT Press, 1998.

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

[13] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2581–2593, Nov. 2020.

[14] Z. Du, C. Wu, T. Yoshinaga, K.-L. A. Yau, Y. Ji, and J. Li, "Federated learning for vehicular Internet of Things: Recent advances and open issues," *IEEE Open J. Comput. Soc.*, vol. 1, pp. 45–61, 2020.

[15] J. Konecný, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon, "Federated Learning: Strategies for Improving Communication Efficiency," in *Proc. NIPS Workshop Private MultiParty Mach. Learn.*, 2016, pp. 1–10.

[16] Z. Lian, W. Wang, Z. Han, and C. Su, "Blockchain-based personalized federated learning for Internet of Medical Things," *IEEE Trans. Sustain. Comput.*, vol. 8, no. 4, pp. 694–702, Oct.–Dec. 2023, doi: 10.1109/TSUSC.2023.3279111.

[17] S. S. Shinde, A. Bozorgchenani, D. Tarchi, and Q. Ni, "On the design of federated learning in latency and energy constrained computation offloading operations in vehicular edge computing systems," *IEEE Trans. Veh. Technol.*, vol. 71, no. 2, pp. 2041–2057, Feb. 2022.

[18] D. Ye, R. Yu, M. Pan, and Z. Han, "Federated learning in vehicular edge computing: A selective model aggregation approach," *IEEE Access*, vol. 8, pp. 23920–23935, 2020.

[19] C. T. Dinh et al., "Federated learning over wireless networks: Convergence analysis and resource allocation," *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 398–409, Feb. 2021.

[20] E. El Haber, T. M. Nguyen, C. Assi, and W. Ajib, "Macro-cell assisted task offloading in MEC-based heterogeneous networks with wireless backhaul," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 4, pp. 1754–1767, Dec. 2019.

[21] C. Xu, G. Zheng, and X. Zhao, "Energy-minimization task offloading and resource allocation for mobile edge computing in NOMA heterogeneous networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 16001–16016, Dec. 2020.

[22] Y. Guo, Z. Zhao, K. He, S. Lai, J. Xia, and L. Fan, "Efficient and flexible management for Industrial Internet of Things: A federated learning approach," *Comput. Netw.*, vol. 192, Jun. 2021, Art. no. 108122.

[23] L. Tang and H. Hu, "Computation offloading and resource allocation for the Internet of Things in energy-constrained MEC-enabled HetNets," *IEEE Access*, vol. 8, pp. 47509–47521, 2020.

[24] S. Li, W. Sun, Y. Sun, and Y. Huo, "Energy-efficient task offloading using dynamic voltage scaling in mobile edge computing," *IEEE Trans. Netw. Sci. Eng.* , vol. 8, no. 1, pp. 588–598, Jan.–Mar. 2021.

[25] D. Van Huynh et al., "URLLC edge networks with joint optimal user association, task offloading and resource allocation: A digital twin approach," *IEEE Trans. Commun.*, vol. 70, no. 11, pp. 7669–7682, Nov. 2022.

[26] Q. Su, Q. Zhang, W. Li, and X. Zhang, "Primal-dual-based computation offloading method for energy-aware cloud-edge collaboration," *IEEE Trans. Mobile Comput.*, vol. 23, no. 2, pp. 1534–1549, Feb. 2024, doi: 10.1109/TMC.2023.3237938.

[27] H. Guo, J. Liu, and J. Zhang, "Computation offloading for multiaccess mobile edge computing in ultradense networks," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 14–19, Aug. 2018.

[28] L. Chen, J. Wu, J. Zhang, H.-N. Dai, X. Long, and M. Yao, "Dependency-aware computation offloading for mobile edge computing with edge-cloud cooperation," *IEEE Trans. Cloud Comput.*, vol. 10, no. 4, pp. 2451–2468, Oct.–Dec. 2022.

[29] X. Chen, J. Zhang, B. Lin, Z. Chen, K. Wolter, and G. Min, "Energy-efficient offloading for DNN-based smart IoT systems in cloud-edge environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 683–697, Mar. 2022.

[30] A. B. de Souza, P. A. L. Rego, V. Chamola, T. Carneiro, P. H. G. Rocha, and J. N. de Souza, "A bee colony-based algorithm for task offloading in vehicular edge computing," *IEEE Syst. J.*, vol. 17, no. 3, pp. 4165–4176, Sep. 2023.

[31] C. Shang, Y. Sun, and H. Luo, "A hybrid deep reinforcement learning approach for dynamic task offloading in NOMA-MEC system," in *Proc. 19th Annu. IEEE Int. Conf. Sens., Commun., Netw. (SECON)*, Stockholm, Sweden, 2022, pp. 434–442.

[32] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang, "Edge intelligence for energy-efficient computation offloading and resource allocation in 5G beyond," *IEEE Trans. Veh. Technol*, vol. 69, no. 10, pp. 12175–12186, Oct. 2020.

[33] C. Li et al., "Dynamic offloading for multiuser muti-CAP MEC networks: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 70, no. 3, pp. 2922–2927, Mar. 2021.

[34] M. Tang and V. W. S. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Trans. Mobile Comput.*, vol. 21, no. 6, pp. 1985–1997, Jun. 2022.

[35] H. Zhou, K. Jiang, X. Liu, X. Li, and V. C. M. Leung, "Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing," *IEEE Internet Things J.*, vol. 9, no. 2, pp. 1517–1530, Jan. 2022.

[36] Z. Zhang, N. Wang, H. Wu, C. Tang, and R. Li, "MR-DRO: A fast and efficient task offloading algorithm in heterogeneous edge/cloud computing environments," *IEEE Internet Things J.*, vol. 10, no. 4, pp. 3165–3178, Feb. 2023.

[37] S. B. Prathiba, G. Raja, S. Anbalagan, K. Dev, S. Gurumoorthy, and A. P. Sankaran, "Federated learning empowered computation offloading and resource management in 6G-V2X," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 5, pp. 3234–3243, Sep./Oct. 2022.

[38] C. Pan et al., "Asynchronous federated deep reinforcement learning-based URLLC-aware computation offloading in space-assisted vehicular networks," *IEEE Trans. Intell. Transp. Syst.* , vol. 24, no. 7, pp. 7377–7389, Jul. 2023.

[39] X. Luo, Z. Zhao, and M. Peng, "Tradeoff between model accuracy and cost for federated learning in the mobile edge computing systems," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, Nanjing, China, 2021, pp. 1–6.

[40] X. Zhang, R. Chen, J. Wang, H. Zhang, and M. Pan, "Energy efficient federated learning over cooperative relay-assisted wireless networks," in *Proc. GLOBECOM IEEE Global Commun. Conf.*, Rio de Janeiro, Brazil, 2022, pp. 179–184.

[41] A. Salh et al., "Energy-efficient federated learning with resource allocation for green IoT edge intelligence in B5G," *IEEE Access*, vol. 11, pp. 16353–16367, 2023.

[42] C. Kai, H. Li, L. Xu, Y. Li, and T. Jiang, "Energy-efficient device-to-device communications for green smart cities," *IEEE Trans. Ind. Informat.*, vol. 14, no. 4, pp. 1542–1551, Apr. 2018.

[43] X. Wang et al., "Deep reinforcement learning: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 4, pp. 5064–5078, Apr. 2024.

[44] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, Cambridge, MA, USA: MIT Press, 2016.

[45] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[46] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman, 1979.
[47] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proc. 37th Int. Conf. Mach. Learn. (ICML)*, 2020, pp. 5132–5143.
[48] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, 2020, pp. 429–450.
[49] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Rev.*, vol. 60, no. 2, pp. 223–311, 2018.

Hui Xiao received the B.E. degree from Shandong University, Jinan, China, in 2017, and the M.E. degree from Central South University, Changsha, China, in 2020, where she is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering.

Her main research interests include mobile edge computing and cloud computing.

Zhigang Hu received the B.S., M.S., and Ph.D. degrees from Central South University (CSU), Changsha, China, in 1985, 1988, and 2002, respectively.

In 2002, he joined CSU, where he is a Professor with the School of Computer Science and Engineering. He has published over 200 research papers. His research interests include radar signal processing and classification/recognition, high performance computing, and cloud computing.

Xinyu Zhang received the master's degree from Central South University, Changsha, China, in 2017, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering.

His main research interests include edge computing, cloud computing, and federated deep reinforcement learning.

Aikun Xu received the M.S. degree from Central South University, Changsha, China, in 2022, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering.

His research interests include deep learning, graph neural network, deep reinforcement learning, scheduling, electric vehicles, and edge computing.

Meiguang Zheng received the B.S. and Ph.D. degrees in computer science from Central South University, Changsha, China, in 2005 and 2011 respectively.

She is currently an Associate Professor with the School of Computer Science and Engineering, Central South University. She is currently leading some research projects supported by the National Natural Science Foundation of China. Her research interests include federated learning, distributed machine learning, computer vision, and edge computing.

Keqin Li (Fellow, IEEE) received the B.S. degree in computer science from Tsinghua University, Beijing, China, in 1985, and the Ph.D. degree in computer science from the University of Houston , Houston, TX, USA, in 1990.

He is currently a SUNY Distinguished Professor of Computer Science with the State University of New York, New Paltz, NY, USA. He is also a National Distinguished Professor with Hunan University, Changsha, China. He has authored or coauthored over 900 journal articles, book chapters, and refereed conference papers. He holds nearly 70 patents announced or authorized by the Chinese National Intellectual Property Administration. He is among the world's top five most influential scientists in parallel and distributed computing in terms of both single-year impact and career-long impact based on a composite indicator of Scopus citation database. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyber-physical systems, heterogeneous computing systems, big data computing, high-performance computing, CPU-GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, intelligent, and soft computing.

Dr. Li has received several best paper awards and chaired many international conferences. He is currently an Associate Editor of the *ACM Computing Surveys* and the *CCF Transactions on High Performance Computing*. He has served on the editorial boards of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON CLOUD COMPUTING, the IEEE TRANSACTIONS ON SERVICES COMPUTING, and the IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING. He is an AAAS Fellow and an AAIA Fellow and also a member of Academia Europaea (Academician of the Academy of Europe).