Resource Management for MEC Assisted Multi-layer Federated Learning Framework

Huibo Li, Yijin Pan, Huiling Zhu, Peng Gong and Jiangzhou Wang, *Fellow, IEEE*

***Abstract*—In this paper, a mobile edge computing (MEC) assisted multi-layer architecture is proposed to support the implementation of federated learning in Internet of Things (IoT) networks. In this architecture, when performing a federated learning based task, data samples can be partially offloaded to MEC servers and cloud server rather than only processing the task at the IoT devices. After collecting local model parameters from devices and MEC servers, cloud server makes an aggregation and broadcasts it back to all devices. An optimization problem is presented to minimize the total federated training latency by jointly optimizing decisions on data offloading ratio, computation resource allocation and bandwidth allocation. To solve the formulated NP hard problem, the optimization problem is converted into quadratically constrained quadratic program (QCQP) and an efficient algorithm is proposed based on semidefinite relaxation (SDR) method. Furthermore, the scenario with the constraint of indivisible tasks in devices is considered and an applicable algorithm is proposed to get effective offloading decisions. Simulation results show that the proposed solutions can get effective resource allocation strategy and the proposed multi-layer federated learning architecture outperforms the conventional federated learning scheme in terms of the learning latency performance.**

***Index Terms*—Federated learning, mobile edge computing, cloud radio access network, resource allocation, SDR method.**

I. Introduction

With a rapid increase of Internet of Things (IoT) devices, artificial intelligence (AI) can be enabled to provide intelligent IoT applications, such as smart city and smart home [1]. Many machine learning (ML) techniques are applied to exploit the significant data generated by IoT devices to make IoT applications more effective [2]. Conventionally, ML schemes require distributed devices to send collected data to a central data center for processing [3]. However, the transmission of massive data results in unacceptable latency for some real-time applications. To overcome these concerns, federated learning framework which is one of the most promising distributed learning algorithms has been introduced [4]. In federated learning, IoT devices cooperatively train a shared common ML model and conduct the training process simultaneously with local data samples. Then devices send trained model parameters, e.g, the gradients of model, to the central data center for aggregation. For example, the traffic flow prediction is based on the data samples collected from sensors scattered in a given area, e.g, from mobile phones, cameras, radars, etc. Instead of sending all data samples to the central data center for processing, IoT devices only send the updated the gradients of model for aggregation [5]–[7]. As a result, federated learning leads to a more powerful AI model by combining training model parameters from different kinds of devices.

To get high-precision model parameters, federated learning still requires a large number of global iterations between local devices and the central data center. During the global iterations, the transmission of local training results inevitably generates communication latency. Therefore, small latency is the key performance measure and challenge for federated learning. So far, research efforts have been focused on improving the learning efficiency of federated learning [8]–[10]. However, all these methods suffer from low training accuracy under training time budget. Because the computing ability of IoT devices is limited and local training results have to experience long transmission distance. To address these issues, mobile edge computing (MEC) has been regarded as a promising solution. Many computation-intensive and delay-sensitive tasks can be offloaded from devices to nearby MEC server [11]–[14]. By applying MEC server at edge to assist federated learning, the computing and storage capabilities [15] of MEC server are leveraged to boost model training process [16].

When considering MEC assisted federated learning, most researches utilized MEC server as a central node to aggregate model parameters and solved the resource allocation problem. In [17] and [18], the authors aimed to achieve low-latency federated learning by optimizing communication resource allocation and training parameters in an MEC server assisted aggregation architecture. [19] explored a bandwidth allocation strategy between IoT devices and the associated MEC edge server to minimize the total energy consumption of devices in federated learning. [20] proposed user selection and uplink resource block allocation scheme to minimize the federated learning training loss by considering packet errors and the availability of wireless resources.

However, local training is only conducted when the participants are in charge and with good WiFi connection in order to reduce the impact on battery lifetime of mobile devices.

Manuscript received xx, xx; revised xx, xx. This work was supported in part by the National Natural Science Foundation of China under Grants No.62001107 and No.62073039, and the Research Fund of National Mobile Communications Research Laboratory, Southeast University (No.2023A03). This research work is also supported by the Big Data Computing Center of Southeast University. *(Corresponding author: Yijin Pan, Peng Gong.)*

H. Li and P. Gong are with the School of Mechatronical Engineering, Beijing Institute of Technology, Beijing 100081, China (e-mail:lijanebit@gmail.com, penggong@bit.edu.cn).

Y. Pan is with the National Mobile Communications Research Laboratory, Southeast University, Nanjing 211111, China (email:panyj@seu.edu.cn).

H. Zhu and J. Wang are with the School of Engineering, University of Kent, Canterbury CT2 7NT, U.K. (e-mail: h.zhu@kent.ac.uk, j.z.wang@kent.ac.uk).

To solve this problem, MEC can be used not only as a central aggregation node but also to train data samples with its powerful computing ability. Although federated learning is intended for the privacy preserving application, partial local datasets which are not privacy-sensitive can be offloaded to the MEC for further computation [21], such as autonomous driving and mobile surveillance. Local devices can decide the offoaded data samples they want to share and further improve the model performance [22], [23].

Therefore, leveraging MEC servers for training is envisioned to improve the model performance for the federated learning in the IoT networks. More training iterations can be achieved and a higher precise training results can be obtained under limited time budget. Nevertheless, one disadvantage of MEC-based FL is the limited number of clients each server can access, leading to inevitable training performance loss. While considering multiple MEC servers in federated learning scheme also needs to experience new challenges to tackle. When MEC servers are not responsible for aggregating model parameters but training data, reasonable allocation of computation resources becomes a key issue affecting learning efficiency and accuracy. In addition, when MEC servers play roles as distributed nodes to assist training model, aggregating these distributed model parameters needs to be considered.

Inline with this idea, we consider a multi-layer MEC-assisted federated learning architecture including devices, MEC servers and cloud server. Due to the centralized ability and powerful computing capability, cloud server has abilities to serve as the central node for training data samples and aggregation. This kind of multi-layer can divide a global model into some subgroups to train multiple features. In [24], the authors proposed a multi-layer personality federated learning model to learn students' habit within the online classes. However, the distribution of data samples in each layer will affect the training and transmission delay. Another key issue affecting the learning delay is that training results from different devices cannot reach the cloud server at the same time. Distributed training process in multi-layer scheme needs to be analyzed. These challenges have not been addressed in MEC-assisted multi-layer federated learning architecture.

In this paper, a multi-layer architecture is proposed for performing federated learning based tasks. In this multi-layer federated learning architecture, the mobile devices, MEC servers and cloud server are designed as the base layer, the middle layer and the top layer, respectively. Data samples can be offloaded to each layer to be trained distributely under a strict encryption rule to protect data privacy. After training, IoT devices and MEC servers need to transmit the model parameters to cloud server for aggregation. Then, based on the architecture, the data offloading procedure for one global training iteration is analyzed. Furthermore, when collected data samples are trained as a whole data package, a data offloading procedure is also considered in this multi-layer architecture. Our target is to minimize the total federated learning delay by optimizing the computation and communication resource allocation.

The contributions of this paper are summarized as follows,

- We propose an MEC assisted multi-layer architecture to achieve data training of IoT devices in federated learning. Data samples collected from IoT devices can be partially offloaded to MEC servers and cloud server for joint training. The training results will be uploaded and aggregated in the cloud server. Based on the synchronous training method, we also construct offloading and joint training procedure of each layer within one global training iteration. Then, an optimization problem is formulated to minimize the total federated learning latency by jointly optimizing data offloading ratio, computation resource allocation in MEC servers and bandwidth allocation in wireless transmissions.
- To solve the optimization problem which is NP hard, we reformulate and transform it into quadratically constrained quadratic program (QCQP) format. We then propose an efficient algorithm based on semidefinite relaxation (SDR) method to solve this challenging problem. We recover feasible offloading decisions from the relaxed solution and get the approximate solution by applying Gaussian randomization way.
- We extend the original problem to the total offloading scenario. A solution to get an efficient data offloading scheme with the constraint of indivisible tasks is proposed. We perform numerical simulations to evaluate the proposed algorithms. The simulation results show that applying the proposed MEC assisted multi-layer architecture in federated learning can significantly reduce data training latency compared to the conventional federated learning scheme.

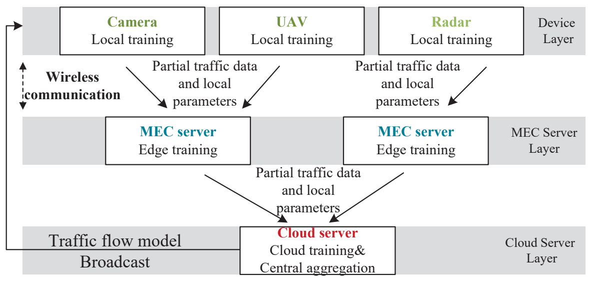


Fig. 1. MEC assisted federated learning in traffic flow prediction

The rest of the paper is organized as follows. The system model is described and the optimization problem is formulated in Section II. The transformation of the optimization problem and a feasible solution are given in Section III. The scenario with the constraint of indivisible tasks is further studied in Section IV. Simulation results are shown and discussed in Section V and the conclusions are drawn in Section VI.

II. System Model

Consider a multi-layer MEC architecture, which can utilize the federated learning to support various IoT applications spreading across wide areas, e.g. traffic flow analysis and congestion prediction. As shown in Fig. 1, this multi-layer architecture includes device layer, MEC server layer and the cloud server layer. In the device layer, multiple distributed

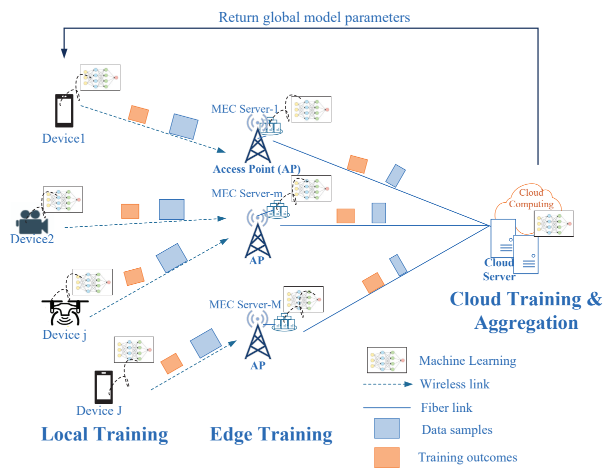


Fig. 2. Multi-layer MEC assisted federated learning architecture

IoT devices, such as cameras and unmanned aerial vehicles (UAVs), need to collect the traffic data samples. MEC server layer provides computation service for training data offloaded from nearby IoT devices. The cloud server can also train received data samples and achieve the central aggregation for local model parameters from all distributed nodes including IoT devices and MEC servers. After aggregation, the cloud server will broadcast updated global traffic flow model parameters to all distributed nodes. Fig. 2 shows the details of training process in the MEC assisted multi-layer federated learning.

A. Federated learning process in MEC assisted multi-layer architecture

The multi-layer architecture consists of a set of devices, a set of access points (APs) equipped with MEC servers and the central cloud server. These MEC servers can process data samples and also can transmit data samples to the cloud server through wired fiber link with high transmission rate. $\mathcal{J} = \{1, 2, \cdots, J\}$ is defined as the set of devices. $\mathcal{M} = \{1, 2, \cdots, M\}$ presents as the set of MEC servers locating in corresponding wireless APs.

As shown in Fig 2, the devices can offload part of data samples to the MEC servers and the cloud server. After training local data samples through machine learning, devices and MEC servers will send local model parameters to the cloud server. The cloud server will train the received data samples and make an aggregation after collecting all distributed local model parameters. Finally, the cloud server returns the global model parameters to all devices to complete the federated learning task. This round of task processing is called one global iteration.

As described above, the data samples generated from the devices have three task processing choices which are training at local device, training at corresponding MEC server and training at cloud. Define u_j^D, u_j^M and u_j^C as the ratios of data samples to the total amount of data samples, which are trained at local devices, MEC server and the cloud server, respectively. We have

$$u_j^D + u_j^M + u_j^C = 1, \quad \forall j \in \mathcal{J},\ u_j^D, u_j^M, u_j^C \in [0,1]. \quad (1)$$

In the following, the learning process in each layer of MEC assisted multi-layer federated learning is described.

1) Device layer

In device layer, the data sample set collected by the device j is defined as $\mathcal{D}_j$. One data sample d, $d \in \mathcal{D}_j$, is described as $(\boldsymbol{x_d}, y_d)$, where $\boldsymbol{x_d}$ is the raw data vector, and y_d is the label. The federated learning target is to train the proper set of model parameters $\boldsymbol{\omega_j}$, which can output the label y_d according to the input $\boldsymbol{x_d}$ through linear regression iterations. Loss function $f_d(\boldsymbol{\omega_j}, \boldsymbol{x_d}, y_d)$ is introduced to evaluate the prediction performance for data sample d, which is determined by learning tasks [25]. For example, $f_d(\boldsymbol{\omega_j}, \boldsymbol{x_d}, y_d) = \frac{1}{2}(\boldsymbol{x_d^T}\boldsymbol{\omega_j} - y_d)^2$ is applied in a federated learning task. If the set of data $\mathcal{D}_j$ is processed at the local device, the task loss function at device j is denoted as

$$L_{j,a}(\boldsymbol{\omega_j}) = \frac{1}{N_j u_j^D} \sum_{n=1}^{N_j u_j^D} f_d(\boldsymbol{\omega_j}, \boldsymbol{x_d}, y_d), \quad (2)$$

where N_j denotes the total number of data samples collected by device j, i.e. $N_j = |\mathcal{D}_j|$. Then, the learning objective is to find optimal $\boldsymbol{\omega_j^*}$ to minimize $L_{j,a}(\boldsymbol{\omega_j})$ which is given by

$$\boldsymbol{\omega_j^*} = \arg\min L_{j,a}(\boldsymbol{\omega_j}). \quad (3)$$

In the federated learning procedure, stochastic gradient descent (SGD) method with intensive iterations is applied at each device to solve the problem in (3). In each iteration round n, each device j will compute $\boldsymbol{\omega_j}^{(n)}$ as follows,

$$\boldsymbol{\omega_j}^{(n)} = \boldsymbol{\omega_j}^{(n-1)} - \alpha \nabla L_{j,a}(\boldsymbol{\omega_j}^{(n-1)}), \quad (4)$$

where α denotes the learning rate, and $\nabla L_{j,a}(\boldsymbol{\omega_j}^{(n-1)})$ represents the gradient of loss function $L_{j,a}(\boldsymbol{\omega_j})$.

Generally, to compute the local model parameters $\boldsymbol{\omega_j}$, it needs multiple local iterations to obtain an accuracy of θ which means the local solvers achieve $f(\boldsymbol{\omega_j}) - f(\boldsymbol{\omega_j}^*) \leq \theta$. Therefore, the computation time at devices depends on the number of local iterations. In federated learning tasks, the upper bound of the number of local iterations is $\mathcal{O}(\log(1/\theta))$ for entensive iterative algorithms such as coordinate descent or gradient descent [26] [27]. As a result, it requires more iterations to get more accurate local model parameters. Assuming that one local iteration time is denoted as $T_j^{ite} = \frac{N_j u_j^D c_j}{f_j}$, where c_j is denoted as the number of central processing unit (CPU) cycles to compute one data sample at device j for one iteration. f_j is denoted as the CPU frequency of device j. The computation time of one global iteration is $T_j^{ite} \cdot \log(1/\theta)$. Then, to obtain the optimal set of local model parameters $\boldsymbol{\omega_j}^{(n)}$, the computation latency at device j is given by

$$t_j^{comp} = T_j^{ite} \cdot \log(1/\theta) = \frac{N_j u_j^D c_j}{f_j} \cdot \log(1/\theta). \quad (5)$$

At the same time, devices will offload the remaining data samples which are not privacy-sensitive to the MEC servers and the cloud server. It is assumed that each device is served by its nearest AP which is equipped with a MEC server due to the consideration of channel conditions and data privacy. The set of devices that are served by MEC server m is represented by $\mathcal{U}_m$. Each device accesses to one MEC server via a uniquely allocated channel with adaptive channel bandwidth $B_{j,m}$, so that there is no interference among the devices. Spectral efficiency between device j and MEC server m is given by

$$r_{j,m} = \log_2(1+\frac{p_j|h_{j,m}|^2}{\sigma_N^2}), \quad \forall j \in \mathcal{U}_m, \quad m \in \mathcal{M}, \tag{6}$$

where p_j is denoted as the transmit power of device j. $h_{j,m} = g_{j,m}d_{j,m}^{-\delta/2}$ represents the channel gain between device j and MEC server m. $g_{j,m}$ is the small scale fading, which follows Rayleigh distribution. $d_{j,m}$ is the distance between device j and MEC server m, and δ is the path loss exponent. σ_N^2 is the variance of the additive white Gaussian noise.

After finishing the computation for training data samples, device j uploads the local model parameters to the cloud server for aggregation. q is denoted as the size of local model parameters, which is much smaller than the size of data sample. Denote $r_{m,c}$ as the transmission data rate between MEC server m and the cloud server in the cloud center. The latency for transmitting model parameters from device j to MEC server m is obtained as

$$t_j^{out} = \frac{q}{B_{j,m}r_{j,m}} + \frac{q}{r_{m,c}}. \tag{7}$$

Due to the limited system bandwidth, we assume $\sum_{j\in\mathcal{U}_m} B_{j,m} \leq B_m^{max}$, where B_m^{max} is the maximum bandwidth allocated to MEC server m. Therefore, the latency from the local data samples arriving at the cloud is

$$\begin{aligned} t_j^{device} &= t_j^{comp} + t_j^{out} \\ &= \frac{N_j u_j^D c_j}{f_j}\cdot\log(1/\theta) + \frac{q}{B_{j,m}r_{j,m}} + \frac{q}{r_{m,c}}. \end{aligned} \tag{8}$$

2) MEC servers layer

The data samples offloaded from device j need to be transmitted to the MEC server m based on the strict encryption rule. The transmission time is given by

$$t_{j,m}^{tran} = \frac{N_j s(u_j^M + u_j^C)}{B_{j,m}r_{j,m}}, \tag{9}$$

where s represents the bit size of one data sample.

After receiving data samples from the associated devices, the MEC servers start to train learning model. It is assumed that the MEC server adopts a space sharing strategy in its internal task processing mechanism and process multiple tasks simultaneously [28]. Each MEC server trains data samples received from connected devices according to the loss function as follows,

$$L_{j,m}(\boldsymbol{\omega_j}) = \frac{1}{N_j u_j^M}\sum_{n=1}^{N_j u_j^M} f_d(\boldsymbol{\omega_j}). \tag{10}$$

Given the SGD method, the computation time in the MEC server is given by

$$t_{j,m}^{comp} = \frac{N_j u_j^M c_j}{f_{j,m}}\cdot\log(1/\theta), \tag{11}$$

where $f_{j,m}$ represents as computation resource allocated to device j at server m. The computation resource allocated to devices should not exceed each MEC server's maximum computing capacity which means $\sum_{j\in\mathcal{U}_m} f_{j,m} \leq f_m^{max}$.

After finishing computation, MEC server m needs to upload the training results to the cloud server. The uploading time is given by $t_{j,m}^{out} = \frac{q}{r_{m,c}}$. Therefore, the total latency of data samples trained in the MEC server is

$$\begin{aligned} t_j^{mec} &= t_{j,m}^{tran} + t_{j,m}^{comp} + t_{j,m}^{out} \\ &= \frac{N_j s(u_j^M + u_j^C)}{B_{j,m}r_{j,m}} + \frac{N_j u_j^M c_j}{f_{j,m}}\cdot\log(1/\theta) + \frac{q}{r_{m,c}}. \end{aligned} \tag{12}$$

3) Cloud server layer

Data samples offloaded to the cloud server will experience wireless transmission from device to the associated MEC server and wired transmission from MEC server to the cloud server. Therefore, the transmission time is given by

$$t_{j,m}^{tran} + t_{j,m,c}^{tran} = \frac{N_j s(u_j^M + u_j^C)}{B_{j,m}r_{j,m}} + \frac{N_j s u_j^C}{r_{m,c}}. \tag{13}$$

The cloud server will train data samples of device j by using the loss function

$$L_{j,c}(\boldsymbol{\omega_j}) = \frac{1}{N_j u_j^C}\sum_{n=1}^{N_j u_j^C} f_d(\boldsymbol{\omega_j}). \tag{14}$$

Then, the computation time for computing model parameters in the cloud server is

$$t_{j,m,c}^{comp} = \frac{N_j u_j^C c_j}{f_c}\cdot\log(1/\theta), \tag{15}$$

where f_c represents computation resource of the cloud which has unlimited computation capacity.

Since the whole federated learning is performed by three layers, the global loss function is given by

$$\begin{aligned} F(\boldsymbol{\omega^n}) &= \frac{\sum_{j=1}^{J} N_j(N_j u_j^a L_{j,a}(\boldsymbol{\omega_j}) + N_j u_j^m L_{j,m}(\boldsymbol{\omega_j}) + N_j u_j^c L_{j,c}(\boldsymbol{\omega_j}))}{\sum_{j=1}^{J} N_j} \\ &= \frac{\sum_{j=1}^{J} N_j \sum_{d\in D_j} f_d(\boldsymbol{\omega_j})}{\sum_{j=1}^{J} N_j}. \end{aligned} \tag{16}$$

It can be proved that the global loss function (16) is the same as that of conventional federated learning based on the average federated approach [29]. To minimize the global loss function, the cloud server combines and aggregates all devices' training results to get a global model parameters in the n th iteration $\boldsymbol{\omega}^n$ as

$$\begin{aligned} \boldsymbol{\omega}^n &= \frac{\sum_{j=1}^{J} N_j \boldsymbol{\omega_j}^n}{\sum_{j=1}^{J} N_j} \\ &= \frac{\sum_{j=1}^{J} N_j u_j^a \boldsymbol{\omega_j}^n + N_j u_j^m \boldsymbol{\omega_j}^n + N_j u_j^c \boldsymbol{\omega_j}^n}{\sum_{j=1}^{J} N_j}. \end{aligned} \tag{17}$$

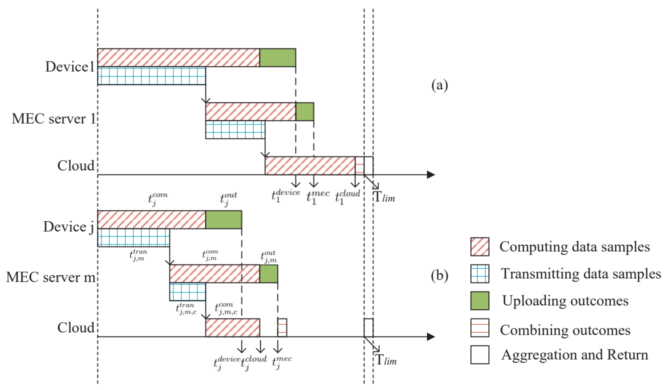


Fig. 3. A whole implementation progress of federated learning in multi-layer architecture

The latency of combination and aggregation can be neglected. Therefore, latency for getting results trained in the cloud server is

$$\begin{aligned} t_j^{cloud} &= t_{j,m}^{tran} + t_{j,m,c}^{tran} + t_{j,m,c}^{comp} \\ &= \frac{N_j s(u_j^M + u_j^C)}{B_{j,m} r_{j,m}} + \frac{N_j s u_j^C}{r_{m,c}} + \frac{N_j c_j u_j^C}{f_c} \cdot \log(1/\theta). \end{aligned} \tag{18}$$

After aggregation, the cloud server will broadcast the updated global model parameters to all distributed nodes in the downlink. Due to the small size of global model parameters, the return time can be neglected compared to the data transmission time in the uplink.

B. Convergence Analysis

Assuming that the federated learning algorithm achieves global accuracy η, the optimum solution of model parameter needs to satisfy

$$F(\boldsymbol{\omega}^{(n)}) - F(\boldsymbol{\omega}^*) \leq \eta(F(\boldsymbol{\omega}^{(0)}) - F(\boldsymbol{\omega}^*)) \tag{19}$$

Because loss function such as linear or logistic loss function has been widely applied in federated learning, the following assumptions are assumed, (1) $F_j(\boldsymbol{\omega})$ is L-Lipschitz which means $F_j(\boldsymbol{\omega}) \leq L\boldsymbol{I}$, (2) $F_j(\boldsymbol{\omega})$ is γ-strongly convex, i.e. $F_j(\boldsymbol{\omega}) \geq \gamma\boldsymbol{I}$. The values of γ and L are determined by the loss function. From [30], the multi-layer federated learning converges to achieve the global accuracy. It has proved the convergence rate which means that the number of iterations n are ruled as follows,

$$n \geq \frac{\frac{2L^2}{\gamma^2\delta}\log(1/\eta)}{1-\theta} \tag{20}$$

to realize $F(\boldsymbol{\omega}^{(n)}) - F(\boldsymbol{\omega}^*) \leq \eta(F(\boldsymbol{\omega}^{(0)}) - F(\boldsymbol{\omega}^*))$. Therefore, in order to achieve convergence, the latency of one global iteration has a upper bound $\frac{t_{budget}}{n}$, where t_{budget} represents the total latency limitation.

C. Problem formulation

An iteration of federated learning is shown in Fig. 3. The widely adopted synchronous update scheme is considered so that all devices start a new global training round simultaneously. In addition, all the devices are initialized by the same initial global parameters which is from last iteration updates. However, the computation and communication delay from different devices may be different. As shown in the case (a) of Fig. 3, the total delay depends on computation time spent in cloud server. However, it can be observed from the case (b) that the MEC server is the bottleneck for the total delay. Therefore, the total latency T_{lim} can be formulated as

$$T_{lim} = \max_{j \in \mathcal{J}} \{t_j^{device}, t_j^{mec}, t_j^{cloud}\}. \tag{21}$$

Our goal is to minimize the total learning latency in this multi-layer scheme by jointly optimizing the data samples offloading decision $\boldsymbol{u_j}$, the computation resource allocation $f_{j,m}$ and the bandwidth allocation $B_{j,m}$. The optimization problem can be formulated as follows,

$$\min_{\boldsymbol{u_j}, \{f_{j,m}\}, \{B_{j,m}\}} \quad T_{lim} \tag{22a}$$

$$s.t. \; u_j^D + u_j^M + u_j^C = 1, \quad \forall j \in \mathcal{J}, \tag{22b}$$

$$u_j^D, u_j^M, u_j^C \in [0,1], \quad \forall j \in \mathcal{J}, \tag{22c}$$

$$\sum_{j \in \mathcal{U}_m} B_{j,m} \leq B_m^{max}, \quad \forall m \in \mathcal{M}, \tag{22d}$$

$$\sum_{j \in \mathcal{U}_m} f_{j,m} \leq f_m^{max}, \quad \forall m \in \mathcal{M}, \tag{22e}$$

$$T_{lim} \leq \frac{t_{budget}}{n}. \tag{22f}$$

Constraint (22b) means that training tasks can be jointly executed on different layers and constraint (22c) guarantees data samples can be partially offloaded. Constraint (22d) means that the bandwidth between MEC servers and the associated devices is limited. Constraint (22e) is the maximum computational capacity constraint of MEC servers. Constraint (22f) means the upper bound of the latency of one global iteration to achieve convergence.

III. Jointly optimizing offloading decisions and resource allocation solution

A. Problem statements

It can be seen that problem (22) is an unbounded knapsack problem which is NP-hard problem [31]. The offloading ratios can be considered as the number of knapsack. $B_{j,m}$ and $f_{j,m}$ can be considered as the weight constraints and latency cost by data samples can be considered as the profit. There also exists a strong coupling caused by limited communication and computation constraints $\{f_{j,m}\}$ and $\{B_{j,m}\}$ in MEC servers making the problem for offloading decisions complicated. Therefore, it is a challenging problem to solve. The objective is first equivalently transformed by introducing the auxiliary variable. To solve the optimization problem, we then transform the optimization problem into QCQP format. The second step is to present our proposed solution based on SDR approach to relax the original problem. Then, we can obtain the offloading decisions and resource allocation scheme from the recovered solution.

B. SDR method based jointly optimizing offloading decisions and resource allocation solution

For the transformation of QCQP, constraint (22c) can be rewrite as follows,

$$0 \le u_j^s(1-u_j^s) \le 1, \quad s=\{D,M,C\}. \tag{23}$$

To solve the problem (22), the delay term can be moved from the objective to the constraints by introducing an auxiliary variable T. Defining $A_j = N_j c_j \log(1/\theta)$, $Q=\frac{q}{r_{m,c}}$, $S_j = N_j s$, the optimization problem (22) is equivalent to the following problem

$$\min_{\boldsymbol{u_j},\{f_{j,m}\},\{B_{j,m}\},T} T \tag{24a}$$

$$s.t. \quad 22(b), 22(d), 22(e), 22(f), (23),$$

$$\frac{A_j u_j^D}{f_j} + \frac{q}{B_{j,m} r_{j,m}} + Q \le T, \tag{24b}$$

$$\frac{S_j}{B_{j,m} r_{j,m}}(u_j^M + u_j^C) + \frac{A_j u_j^M}{f_{j,m}} + Q \le T, \tag{24c}$$

$$\frac{S_j}{B_{j,m} r_{j,m}}(u_j^M + u_j^C) + (\frac{S_j}{r_{m,c}} + \frac{A_j}{f_c})u_j^C \le T. \tag{24d}$$

It can be observed that the problem (24) is non-convex due to the nonconvexity of the formulation (24b) $\sim$ (24d). The following steps show the optimization problem (24) can transformed into a QCQP problem.

First, in order to transform (24b)$\sim$(24d) into quadratic form, we introduce additional auxiliary variables D_j^d, D_j^a and D_j^f. Constraints (24b)$\sim$(24d) can be equivalently replaced by the following forms,

$$\frac{A_j}{f_j} u_j^D + D_j^d + Q \le T \tag{25}$$

$$D_j^a + D_j^f + Q \le T \tag{26}$$

$$D_j^a + (\frac{S_j}{r_{m,c}} + \frac{A_j}{f_c})u_j^C \le T \tag{27}$$

Besides, the auxiliary variables should satisfy that

$$\frac{q}{B_{j,m} r_{j,m}} \le D_j^d \tag{28}$$

$$\frac{S_j}{B_{j,m} r_{j,m}}(u_j^C + u_j^M) \le D_j^a \tag{29}$$

$$\frac{A_j}{f_{j,m}} u_j^M \le D_j^f \tag{30}$$

Next, to transform the problem (24) into a vectorial form, a new decision vector $\mathbf{w}_j$ is defined as follows,

$$\mathbf{w}_0 = [T, T-Q, \mathbf{0}_{1*6}]^T, \tag{31}$$

and

$$\mathbf{w}_j = [u_j^D, u_j^M, u_j^C, B_{j,m}, D_j^d, D_j^a, f_{j,m}, D_j^f]^T, \quad \forall j \in \mathcal{J}, \tag{32}$$

where $\mathbf{w}_0$ contains the optimization objective and $\mathbf{w}_j$ contains all decision variables of device j. Then, the optimization objective (24a) can be rewritten as

$$\min \sum_{j=0}^{J} \mathbf{b}_j^T \mathbf{w}_j, \tag{33}$$

where $\mathbf{b}_0 = [1, \mathbf{0}_{1*7}]^T$ and $\mathbf{b}_j = [\mathbf{0}_{1*8}]^T$.

In the following, we present each constraint in problem (24) into a corresponding matrix form. The offloading ratio constraint (22b) can be converted as

$$(\mathbf{b}_j^E)^T \mathbf{w}_j = 1, \quad \forall j \in J, \tag{34}$$

where $\mathbf{b}_j^E = [1,1,1,\mathbf{0}_{1*5}]$.

The bandwidth resource constraint of (22d) can be written as

$$\sum_{j \in \mathcal{U}_m} (\mathbf{b}_j^N)^T \mathbf{w}_j \le B_m^{max}, \tag{35}$$

where $\mathbf{b}_j^N = [0,0,0,1,0,0,0,0]^T$.

Similarly, the computation resource constraint of (22e) can be written as

$$\sum_{j \in U_m} (\mathbf{b}_j^M)^T \mathbf{w}_j \le f_m^{max}, \tag{36}$$

where $\mathbf{b}_j^M = [\mathbf{0}_{1*6}, 1, 0]^T$. The latency for convergence constraint (22f) can be written as $\sum_{j=0}^{J} \mathbf{b}_j^T \mathbf{w}_j \le \frac{t_{budget}}{n}$.

The constraint (23) of the offloading ratio range can be rewritten as

$$0 \le \mathbf{w}_j^T diag(\mathbf{e}_i)^T \mathbf{w}_j - (\mathbf{e}_i)^T \mathbf{w}_j \le 1, \quad j \in \mathcal{J}, \; i \in \{1,2,3\}, \tag{37}$$

where each $\mathbf{e}_i$ is a $8*1$ standard unit vector with the ith entry being 1.

The constraints of (25)$\sim$(27) can be rewritten as

$$\sum_{k=0}^{J} (\mathbf{b}_{jk}^s)^T \mathbf{w}_k \le 0, \quad j \in \mathcal{J}, \quad s \in \{d,a,f\}, \tag{38}$$

where

$$\mathbf{b}_{j0}^d = \mathbf{b}_{j0}^a = [0,-1,\mathbf{0}_{1*6}]^T,$$
$$\mathbf{b}_{j0}^f = [-1, \mathbf{0}_{1*7}]^T,$$
$$\mathbf{b}_{jj}^d = [\tfrac{A_j}{f_j},0,0,0,1,0,0,0],$$
$$\mathbf{b}_{jj}^a = [0,0,0,0,0,1,0,1],$$
$$\mathbf{b}_{jj}^f = [0,0,\tfrac{S_j}{r_{m,c}} + \tfrac{A_j}{f_{j,c}},0,0,1,0,0],$$
$$\mathbf{b}_{jk}^s = \mathbf{0}, \quad s \in \{d,a,f\}, \quad k \ne \{0,j\}.$$

The constraint of (28) can be rewritten as

$$\mathbf{w}_j^T \mathbf{A}_j^g \mathbf{w}_j \le -(\mathbf{b}_j^g)^T \mathbf{b}_j^{\overline{g}}, \quad j \in \mathcal{J}, \tag{39}$$

where

$$\mathbf{A}_j^g = \begin{bmatrix} \mathbf{0}_{3*3} & 0 & 0 & \mathbf{0}_{3*3} \\ \mathbf{0}_{1*3} & 0 & -0.5r_{j,m} & \mathbf{0}_{1*3} \\ \mathbf{0}_{1*3} & -0.5r_{j,m} & 0 & \mathbf{0}_{1*3} \\ \mathbf{0}_{3*3} & \mathbf{0}_{3*1} & \mathbf{0}_{3*1} & \mathbf{0}_{3*3} \end{bmatrix},$$

$$\mathbf{b}_j^g = [q, \mathbf{0}_{1*7}]^T, \quad \mathbf{b}_j^{\overline{g}} = [1, \mathbf{0}_{1*7}]^T.$$

The constraints of (29) and (30) can be rewritten as

$$\mathbf{w}_j^T \mathbf{A}_j^v \mathbf{w}_j + (\mathbf{b}_j^v)^T \mathbf{w}_j \le 0, \quad j \in \mathcal{J}, v \in \{c,p\}, \tag{40}$$

where

$$\mathbf{A}_j^c = \begin{bmatrix} \mathbf{0}_{3*3} & \mathbf{0}_{3*1} & \mathbf{0}_{3*1} & \mathbf{0}_{3*1} & \mathbf{0}_{3*2} \\ \mathbf{0}_{1*3} & 0 & 0 & -0.5 & \mathbf{0}_{1*2} \\ \mathbf{0}_{1*3} & 0 & 0 & 0 & \mathbf{0}_{1*2} \\ \mathbf{0}_{1*3} & -0.5 & 0 & 0 & \mathbf{0}_{1*2} \\ \mathbf{0}_{2*3} & \mathbf{0}_{2*1} & \mathbf{0}_{2*1} & \mathbf{0}_{2*1} & \mathbf{0}_{2*2} \end{bmatrix},$$

$$\mathbf{A}_j^p = \begin{bmatrix} \mathbf{0}_{6*6} & 0 & 0 \\ \mathbf{0}_{1*6} & 0 & -0.5 \\ \mathbf{0}_{1*6} & -0.5 & 0 \end{bmatrix},$$

$$\mathbf{b}_j^c = [0, \frac{S_j}{r_{j,m}}, \frac{S_j}{r_{j,m}}, \mathbf{0}_{1*5}]^T, \quad \mathbf{b}_j^p = [0, A_j, 0, \mathbf{0}_{1*5}]^T.$$

After the above transformations, we define a new variable $\mathbf{x}_j = [\mathbf{w}_j, 1]^T$ and $j \in J \cup \{0\}$. The optimization problem can be converted into the QCQP formulation as

$$\min \sum_{j=0}^{J} \mathbf{x}_j^T \mathbf{G}_j \mathbf{x}_j \tag{41a}$$

$$s.t.\ \ \mathbf{x}_j^T \mathbf{G}_j^E \mathbf{x}_j = 1, \ \ j \in \mathcal{J}, \tag{41b}$$

$$\sum_{j \in \mathcal{U}_m} \mathbf{x}_j^T \mathbf{G}_j^N \mathbf{x}_j \leq B_m^{max}, \tag{41c}$$

$$\sum_{j \in \mathcal{U}_m} \mathbf{x}_j^T \mathbf{G}_j^M \mathbf{x}_j \leq f_m^{max}, \tag{41d}$$

$$0 \leq \mathbf{x}_j^T \mathbf{G}_i^I \mathbf{x}_j \leq 1, \ \ j \in \mathcal{J}, \ i \in \{1,2,3\}, \tag{41e}$$

$$\sum_{k=0}^{J} \mathbf{x}_k^T \mathbf{G}_{jk}^s \mathbf{x}_k \leq 0, \quad j \in \mathcal{J}, \ s \in \{d, a, f\} \tag{41f}$$

$$\mathbf{x}_j^T \mathbf{G}_j^g \mathbf{x}_j \leq -(\mathbf{b}_j^g)^T \mathbf{b}_j^{\overline{g}}, \quad j \in \mathcal{J}, \tag{41g}$$

$$\mathbf{x}_j^T \mathbf{G}_j^v \mathbf{x}_j \leq 0, \quad j \in \mathcal{J}, \ v \in \{c, p\}, \tag{41h}$$

$$\sum_{j=0}^{J} \mathbf{x}_j^T \mathbf{G}_j \mathbf{x}_j \leq \frac{t_{budget}}{n}, \tag{41i}$$

$$\mathbf{x}_j \geq 0, \quad j \in \mathcal{J} \cup \{0\} \tag{41j}$$

where

$$\mathbf{G}_j = \begin{bmatrix} 0 & \frac{1}{2}\mathbf{b}_j \\ \frac{1}{2}\mathbf{b}_j^T & 0 \end{bmatrix}, \mathbf{G}_j^g = \begin{bmatrix} \mathbf{A}_j^g & 0 \\ 0 & 0 \end{bmatrix},$$

$$\mathbf{G}_j^\pi = \begin{bmatrix} 0 & \frac{1}{2}\mathbf{b}_j^\pi \\ \frac{1}{2}(\mathbf{b}_j^\pi)^T & 0 \end{bmatrix}, \ \pi \in \{E, N, M\},$$

$$\mathbf{G}_i^I = \begin{bmatrix} diag(\mathbf{e}_i) & -\frac{1}{2}\mathbf{e}_i \\ -\frac{1}{2}(\mathbf{e}_i)^T & 0 \end{bmatrix}, \ i \in \{1,2,3\},$$

$$\mathbf{G}_{jk}^s = \begin{bmatrix} 0 & \frac{1}{2}\mathbf{b}_{jk}^s \\ \frac{1}{2}(\mathbf{b}_{jk}^s)^T & 0 \end{bmatrix}, \ s \in \{d, a, f\},$$

$$\mathbf{G}_j^v = \begin{bmatrix} \mathbf{A}_j^v & \frac{1}{2}\mathbf{b}_j^v \\ \frac{1}{2}(\mathbf{b}_j^v)^T & 0 \end{bmatrix}, \ v \in \{c, p\}.$$

The optimization problem (41) is equivalent to the original problem (22), since all constraints have one-to-one corresponding matrix representations. Thus, it is still a non-convex and NP-hard problem in general. To solve problem (41), we further apply SDR approach to relax the problem as many practical applications have proved that SDR approach can obtain accurate and near optimal solutions [32]–[34]. Defining $\mathbf{X}_j = \mathbf{x}_j \mathbf{x}_j^T$, we have

$$\mathbf{x}_j^T \mathbf{G}_j \mathbf{x}_j = Tr(\mathbf{G}_j \mathbf{X}_j), \tag{42}$$

where the rank of matrix $\mathbf{X}_j$ equals one. We can relax problem (41) by dropping the rank constraint rank$(\mathbf{X}_j) = 1$ as follows,

$$\min \sum_{j=0}^{J} Tr(\mathbf{G}_j \mathbf{X}_j) \tag{43a}$$

$$s.t.\ \ Tr(\mathbf{G}_j^E \mathbf{X}_j) = 1, \ \ j \in \mathcal{J}, \tag{43b}$$

$$\sum_{j \in \mathcal{U}_m} Tr(\mathbf{G}_j^M \mathbf{X}_j) \leq f_m^{max}, \tag{43c}$$

$$\sum_{j \in \mathcal{U}_m} Tr(\mathbf{G}_j^N \mathbf{X}_j) \leq B_m^{max}, \tag{43d}$$

$$0 \leq Tr(\mathbf{G}_i^I \mathbf{X}_j) \leq 1, \ \ j \in \mathcal{J}, i \in \{1,2,3\}, \tag{43e}$$

$$\sum_{k=0}^{J} Tr(\mathbf{G}_{jk}^s \mathbf{X}_k) \leq 0, \ \ j \in \mathcal{J}, s \in \{d, a, f\}, \tag{43f}$$

$$Tr(\mathbf{G}_j^g \mathbf{X}_j) \leq -(\mathbf{b}_j^g)^T \mathbf{b}_j^{\overline{g}}, \quad j \in \mathcal{J}, \tag{43g}$$

$$Tr(\mathbf{G}_j^v \mathbf{X}_j) \leq 0, \quad j \in \mathcal{J}, v \in \{c, p\}, \tag{43h}$$

$$\sum_{j=0}^{J} Tr(\mathbf{G}_j \mathbf{X}_j) \leq \frac{t_{budget}}{n}, \tag{43i}$$

$$\mathbf{X}_j \geq 0, \quad j \in \mathcal{J} \cup \{0\}. \tag{43j}$$

The optimization problem (43) has been converted into a semidefinite programming (SDP) problem after relaxing the rank constraints. It can be efficiently solved in polynomial time since the problem is also a convex problem. We can get the optimal solution denoted as $\mathbf{X}_j^*$ by applying some convex optimization methods and toolboxes, such as interior point method. However, the problem (43) is a SDR relaxed form of problem (41). The optimal solution $\mathbf{X}_j^*$ may not be feasible for the original problem (22) since the contraint of rank$(\mathbf{X}_j) = 1$ has been removed. If the solution $\mathbf{X}_j^*$ is of rank-one matrix, it is exactly the optimal solution of the problem (22) since the problem (43) is the convex problem. On the other hand, if the rank of $\mathbf{X}_j^*$ is larger than 1, we need to extract a feasible vector $\mathbf{x}_j^*$ from it in an efficient manner for problem (22).

Generally, the relaxed problem may not lead to a rank-one solution. Therefore, we need to recover the approximate solution from the relaxed solution obtained from (43). Gaussian randomization method is considered as an efficient method to recover the approximation solution from the relaxed solution obtained from (43) based on [32]. We firstly generate a random matrix $\boldsymbol{\xi}_j$ following Gaussian distribution with zero mean and covariance $\mathbf{X}_j^*$, i.e. $\boldsymbol{\xi}_j \sim N(0, \mathbf{X}_j^*)$. Then, we choose the optimal solution candidate $\boldsymbol{\xi}_j^*$ which satisfies $\boldsymbol{\xi}_j^* = argmin\ \boldsymbol{\xi}_j^T C_{\boldsymbol{\xi}_j}$. After obtaining the optimal $\boldsymbol{\xi}_j^*$, we need to normalize the first three elements $\boldsymbol{\xi}_j^*(1)$, $\boldsymbol{\xi}_j^*(2)$, $\boldsymbol{\xi}_j^*(3)$ which have to satisfy the offloading ratio constraints. Based on the reasonable offloding ratio solutions, we can solve the convex problem to get communication and computation resource strategies.

To solve the latency optimization problem (22), the main complexity results from the iteration in semidefinite programming (SDP) problem (43). We can notice that SDP problem can be mathematically solved by the interior point method. Within the precision ϵ, the computation complexity of the proposed algortihm is $\mathcal{O}(J^{3.5}\log(1/\epsilon))$ [35].

IV. Offloading scheme with integer offloading constraints

In some IoT applications, there may be correlation between data samples. For example, IoT devices collect data to learn daily behavious of people. Data samples collected at different time need to be trained together to get a whole behavious model. Therefore, these kind of data samples cannot be partially offloaded. In this section, the optimization problem of minimizing federated learning latency with the constraint of indivisible tasks is considered. In other words, data samples only can be offloaded into one place in the proposed multi-layer federated learning architecture. It means the offloading decisions is given by

$$u_j^D + u_j^M + u_j^C = 1, \quad \forall j \in \mathcal{J},\ u_j^D, u_j^M, u_j^C \in \{0,1\}. \tag{44}$$

This constraint of indivisible tasks can be replaced by

$$u_j^s(u_j^s - 1) = 0, \quad s \in \{D, M, C\}. \tag{45}$$

Similar to Section III, we can formulate the latency for training results of local data samples arriving at cloud as

$$\begin{aligned} t_j^{device} &= (t_j^{comp} + t_j^{out})u_j^D \\ &= (\frac{N_j c_j}{f_j}\cdot\log(1/\theta) + \frac{q}{B_{j,m}r_{j,m}} + \frac{q}{r_{m,c}})u_j^D. \end{aligned} \tag{46}$$

The total latency of data samples trained in the MEC server is

$$\begin{aligned} t_j^{mec} &= (t_{j,m}^{tran} + t_{j,m}^{comp} + t_{j,m}^{out})u_j^M \\ &= (\frac{N_j s}{B_{j,m}r_{j,m}} + \frac{N_j c_j}{f_{j,m}}\cdot\log(1/\theta) + \frac{q}{r_{m,c}})u_j^M. \end{aligned} \tag{47}$$

Latency for getting results trained in cloud is

$$\begin{aligned} t_j^{cloud} &= (t_{j,m}^{tran} + t_{j,m,c}^{tran} + t_{j,m,c}^{comp})u_j^C \\ &= (\frac{N_j s}{B_{j,m}r_{j,m}} + \frac{N_j s}{r_{m,c}} + \frac{N_j c_j}{f_c}\cdot\log(1/\theta))u_j^C. \end{aligned} \tag{48}$$

Similar to Section III, we can introduce a variable $\widetilde{T}$ to represent the value of $\max\{t_j^{device}, t_j^{mec}, t_j^{cloud}\}$. Since data samples only can be offloaded to one place, the optimum objective can be transformed into

$$\max\{t_j^{device}, t_j^{mec}, t_j^{cloud}\} = t_j^{device} + t_j^{mec} + t_j^{cloud}. \tag{49}$$

Therefore, the constraints of (25)$\sim$(27) can be converted into one constaint as

$$t_j^{device} + t_j^{mec} + t_j^{cloud} \leq \widetilde{T}. \tag{50}$$

By introducing additional auxiliary variables D_j^d, D_j^a and D_j^f, constraint (50) can be equivalently replaced by the following forms:

$$(\frac{A_j}{f_j} + Q)u_j^D + D_j^d + D_j^a + D_j^f + Qu_j^M + (\frac{S_j}{r_{m,c}} + \frac{A_j}{f_c})u_j^C \leq \widetilde{T}. \tag{51}$$

With the constraint of integer offloading, the optimization problem is

$$\begin{aligned} \min_{\{\boldsymbol{u_j}\},\{f_{j,m}\},\{B_{j,m}\},\widetilde{T}} \quad & \widetilde{T} \\ s.t. \quad & (22b), (22d), (22e), (22f), (45), (51) \end{aligned} \tag{52}$$

Because $\boldsymbol{u_j}$ is a binary parameter, offloading decision of data samples will be a finite number of choices. A globally optimal solution to problem (52) can be obtained by exhaustive search among 3^J possible offloading decisions. However, the complexity increases exponentially with the number of users and thus the globally optimal solution is impractical. We still choose SDR method to obtain the resource allocation solution.

In order to transform the problem into vectorial form, the variable $\mathbf{w}_j$ which contains all offloading decisions is still need to be introduced. The constraint of (45) can be rewritten as

$$\mathbf{w}_j^T diag(\mathbf{e}_i)^T\mathbf{w}_j - (\mathbf{e}_i)^T\mathbf{w}_j = 0, \quad j \in \mathcal{J}, i \in \{1,2,3\}, \tag{53}$$

where each $\mathbf{e}_j$ is a $8*1$ standard unit vector with the ith entry being 1. We still need to introduce the variable $\mathbf{x}_j = [\mathbf{w}_j, 1]^T$ and $j \in J \cup \{0\}$. The convertion of QCQP is given by,

$$\mathbf{x}_j^T\mathbf{G}_i^I\mathbf{x}_j = 0, \quad j \in \mathcal{J}, i \in \{1,2,3\}, \tag{54}$$

where

$$\mathbf{G}_i^I = \begin{bmatrix} diag(\mathbf{e}_i) & -\frac{1}{2}\mathbf{e}_i \\ -\frac{1}{2}(\mathbf{e}_i)^T & 0 \end{bmatrix}.$$

Furthermore, the constraint (51) can be rewritten as

$$\sum_{k=0}^{J}(\mathbf{b}_{jk}^p)^T\mathbf{w}_k \leq 0, \quad j \in \mathcal{J}, \tag{55}$$

where $\mathbf{b}_{j0}^p = [-1, \mathbf{0}_{1*7}]^T$, $\mathbf{b}_{jj}^p = [\frac{A_j}{f_j} + Q, Q, \frac{N_j s}{r_{m,c}} + \frac{A_j}{f_{j,c}}, 0, 1, 1, 0, 1]^T$, and $\mathbf{b}_{jk}^p = 0, k \neq \{0, j\}$. Rewriting the constraint (55) into QCQP format, we have

$$\sum_{k=0}^{J}\mathbf{x}_k^T\mathbf{G}_{jk}^p\mathbf{x}_k \leq 0, \quad j \in \mathcal{J}, \tag{56}$$

where

$$\mathbf{G}_{jk}^p = \begin{bmatrix} 0 & \frac{1}{2}\mathbf{b}_{jk}^p \\ \frac{1}{2}(\mathbf{b}_{jk}^p)^T & 0 \end{bmatrix}.$$

Similar to Section IV, we still define $\mathbf{X}_j = \mathbf{x}_j\mathbf{x}_j^T$ where the rank of matrix $\mathbf{X}_j$ equals one. The SDR transformation for the optimization problem and constraints are listed as follows,

$$\min\sum_{j=0}^{J}Tr(\mathbf{G}_j\mathbf{X}_j) \tag{57a}$$

$$s.t. \quad 43(b) \sim 43(d), 43(g) \sim 43(i), \tag{57b}$$

$$Tr(\mathbf{G}_i^I\mathbf{X}_j) = 0, \quad j \in \mathcal{J}, i \in \{1,2,3\}, \tag{57c}$$

$$\sum_{k=0}^{J}Tr(\mathbf{G}_{jk}^p\mathbf{X}_k) \leq 0, \quad j \in \mathcal{J}, \tag{57d}$$

$$\mathbf{X}_j \geq 0, \quad j \in \mathcal{J} \cup \{0\}. \tag{57e}$$

To solve the problem (57), we can first apply the similar method in Section III which is utilizing traditional CVX toolbox to get optimal solution $\mathbf{X}^*$ after relaxing rank constraints. Because the solution is still obtained by relaxing SDR problem, we still need to find feasible solution from $\mathbf{X}^*$. Similar to Section III, the values of $\mathbf{X}_j^*(9,i)$ can be used to recover the offloading decisions $\mathbf{x}_j(i)$ for device j because we have defined that $\mathbf{x}_j = [\mathbf{w}_j, 1]^T$. However, the final solution cannot be obtained directly by $\mathbf{X}_j^*(9,i) = \mathbf{x}_j(i)$ with the constraint of integer offloading decisions. Therefore, we need further step to get binary offloading decisions and resource allocation decisions.

It can be observed that the first three elements in $\mathbf{x}_j$ are offloading decisions u_j^D, u_j^M, u_j^C. Therefore, we can use $\mathbf{x}_j(9,i), i = \{1,2,3\}$ to recover the binary offloading decisions. Due to the constraints of (57c) and (57e), the values of $\mathbf{x}_j(9,i), i = \{1,2,3\}$ belongs to $[0,1]$. Then, the offloading decisions can be obtained from $\mathbf{x}_j(9,i), i = \{1,2,3\}$ by mapping its elements to binary number. We utilize the rounding down function which is floor($\mathbf{x}_j(i)$) to decide the offloading decisions. The rounding down results guarantee that constraint (44) can be satisfied strictly [36].

After getting the feasible offloading decisions, we still need to obtain communication and computation resource allocation. The remaining optimization problem is

$$\begin{aligned} &\min_{\{f_{j,m}\},\{B_{j,m}\},\widetilde{T}} \quad \widetilde{T}. \\ &s.t. \quad (22d), (22e) \end{aligned} \tag{58}$$

It can be seen that problem (58) is a convex optimization problem. By applying the standard convex optimization methods and toolboxes we can obtain the final communication and computation resource allocation decisions. The computation complexity consists of two parts: solving the relaxed problem by SDR method and recovering the offloading ratio to the binary numbers due to the constraint of integer offloading. The complexity from SDR method is $\mathcal{O}(J^{3.5}\log(1/\epsilon))$. The complexity from second part is $\mathcal{O}(J)$. Then, we can conclude that the total complexity of problem (57) is $\mathcal{O}(J^{3.5}\log(1/\epsilon)) + \mathcal{O}(J) = \mathcal{O}(J^{3.5})$. However, there are at most $\mathcal{O}(3^J)$ choices in exhaustive search to find the optimal solution with binary offloading constraint.

In order to garantee the optimized problem has the feasible solution, there still has requirements need to be satisfied. For the worst case, all data samples will be trained locally which means $\boldsymbol{u}_j = \{1,0,0\}$. When FL achieving convergence, it needs to satisfy $t_{device} \leq \frac{t_{budget}}{n}$ and the optimization problem is always feasible.

V. Simulation results

In this section, we present the performance of the proposed multi-layer MEC assisted architecture in federated learning. It is assumed that 10 IoT devices are randomly distributed in a circular area of which radius is 500m. 4 wireless APs are uniformly distributed around the circle and equipped with MEC servers. The path loss model is $128.1 + 37.6\log 10(d)$ and d means the distance in km between devices and APs. Each IoT device has image classification learning tasks. Computation and communication parameters of training tasks are presented in Table I.

TABLE I
The simulation parameters

Parameters	Value
The number of data samples N_j	[30000 50000]
Data sample size	20kbits
Required CPU cycles	[2*10^4 4*10^4] cycles
Maximum CPU capacity of MEC servers	20GHz
Maximum Bandwidth of AAU	20MHz
Wired transmit data rate	1Gbps
Noise power	10^{-8}Watt
Transmission power	[8 10]Watt

For performance comparison, we consider three cases as benchmark schemes, 1) Traditional learning scheme: all data samples are trained on local devices and parameters are aggregated at cloud server. It also can be regarded as traditional client-server mode, 2) Cloud assisted learning scheme: data samples can be trained on both the devices and the cloud server. The aggregation will be implemented at the cloud server, 3) Central learning scheme: all data samples are sent to cloud server to train and model parameters are aggregated at the cloud.

A. Performance of multi-layer MEC assisted architecture in federated learning

Fig. 4 shows the average learning latency of different learning schemes versus different achieved local training accuracy. When the value of accuracy is smaller, the trained model is more accurate. It can be seen from the figure that total training time increases with the value of training accuracy decreasing. However, within the same training time, the proposed architecture can achieve more accurate training progress than the other schemes. In addition, the proposed SDR method has a close performance with the approximate optimal solution by applying Gaussian randomization method. For the cloud assisted training scheme, it always has a worse accuracy performance than the proposed architecture which has more resources on the edge. For the traditional learning scheme, when it needs to achieve high accurate training, the latency is higher than the cloud training because the computation resource of devices is very limited. However, when the value of accuracy is larger than 0.1, the latency gap between centralized training and traditional learning scheme is smaller. The reason is that devices can handle less computing tasks to achieve less accurate learning process while the centralized learning scheme has a large latency on communication.

Fig. 5 shows the achieved global accuracy under different total learning time budget. It can be observed that the proposed scheme can achieve more accurate global model parameter compared with the traditional learning scheme under the same time budget. Furthermore, different local training accuracy can affect the total training time. More accurate local training generates longer training latency with the same global accuracy requirement for both schemes. The proposed multi-layer scheme has a lower latency compared with the traditional learning under the same local and global accuracy requirements. Therefore, the obtained resource allocations from the proposed solution can help reduce the total latency.

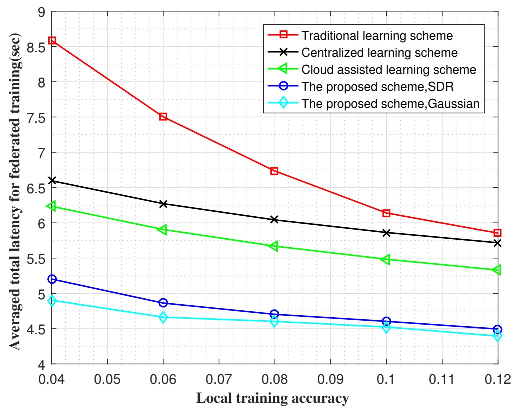


Fig. 4. Average training latency versus training accuracy

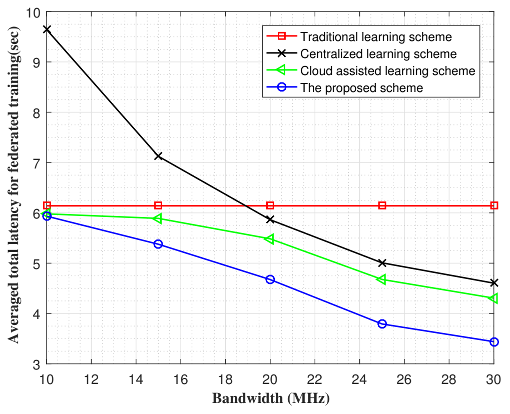


Fig. 6. Average training latency versus the bandwidth of APs

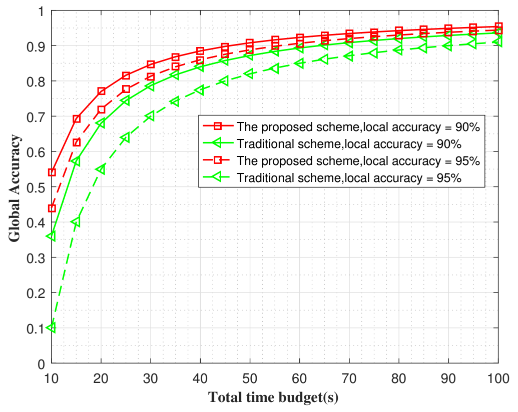


Fig. 5. Achieved global accuracy versus the total learning time budget

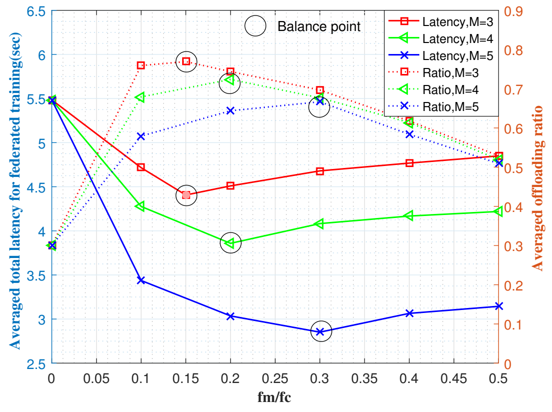


Fig. 7. Average training latency and data samples offloading ratio versus the ratio of MEC server computation resource to cloud server resource

Fig. 6 shows the average latency performance versus the bandwidth of the APs. It can be observed that the training latency decreases with the APs' bandwidth increasing. When the bandwidth is limited, such as bandwidth equals 10MHz, the latency of the proposed scheme and the cloud assisted scheme is close to the device training. The reason is that more data samples will be trained on devices since the latency of communication between devices and the MEC servers is large. For the centralized learning mode, when the bandwidth is smaller than 20MHz, it has a higher latency than local training since all data samples are offloaded to remote cloud. More data samples will be offloaded from devices to MEC servers and the cloud server when the bandwidth is sufficient. When the bandwidth becomes large enough, the change of latency becomes flat. That is reasonable because the latency will depend on computation delay when the latency cost on communication is small.

Fig. 7 shows the average training latency and data offloading ratio versus the ratio of MEC server computation resource to the cloud server resource where the number of devices is 10 and the number of MEC servers are 3, 4 and 5 respectively. We set the total computation resources of MEC servers and the cloud servers as a constant. When the ratio of MEC server computation resource to the cloud server equals zero which means there is no resource in the edge, the total latency is high. With the resource in the edge increasing, the total latency decreases obviously because partial data samples can be offloaded and trained in the MEC servers. Hence it can be observed that the offloading ratio will increase at first. However, when the resource in the edge continues to increase, the learning latency does not linearly decrease with the increase of f_m. This is because the computation resource in cloud server is reduced which affects the total latency and offloading ratio obviously. There exists a balanced point of computation resource ratio which can achieve the best latency performance. The change of balance point varies according to different resource allocation for MEC servers and cloud server. Furthermore, when the number of MEC servers increases which means computing resources become more distributed, the total latency will decreases.

Fig. 8 shows the average training latency versus the ratio of devices computation resource to the total computation resource. For our proposed scheme and cloud assisted training scheme, the total training latency decreases when the computing ability of devices becomes more powerful. However, when the computing capacity of devices' CPU continues to increase, the gap among three learning schemes becomes smaller. Because devices have enough computing ability which leads to more data samples to be trained at the local to save communication latency. Centralized training gives a stable

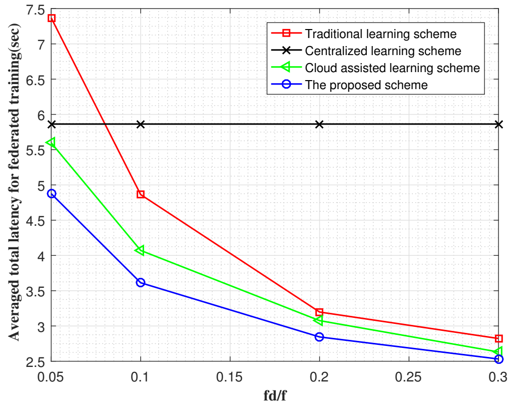


Fig. 8. Average training latency versus the ratio of device computation resource to total resource

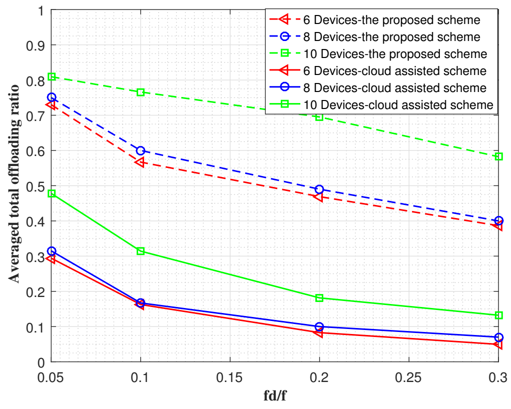


Fig. 9. Data samples offloading ratio versus the ratio of device computation resource to total resource

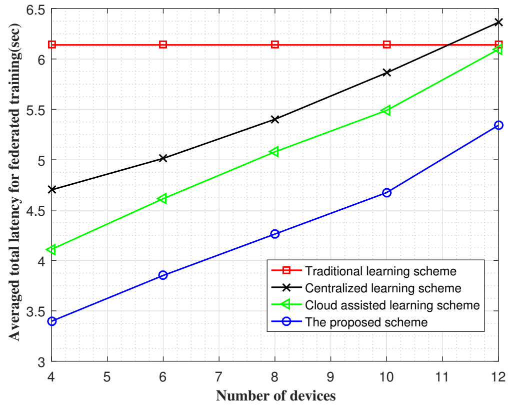


Fig. 10. Average training latency versus the number of devices

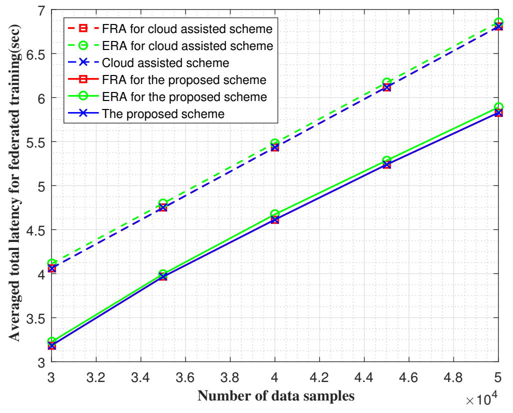


Fig. 11. Average training latency versus the number of data samples

latency performance as it can not be affected by the computing ability of devices. But when the computing ability of devices is very small, centralized training with powerful computing ability can have a better performance than traditional learning scheme even if training at cloud needs to experience a long distance.

Fig. 9 shows the data offloading ratio versus the device computing ability under different number of devices. It can be seen that offloading ratio in the proposed training scheme and cloud assisted training scheme decreases as the computing ability of devices increasing. The reason is that devices have more computation resources to allow more data samples to be trained on devices to save latency. With the same computing ability of devices, the proposed training scheme has a higher offloading ratio than that in cloud assisted scheme since the proposed scheme can offload data to MEC servers to improve training efficiency. Due to the same local computation resource, when the number of devices is small, devices will have more sufficient computing ability and more data samples will be trained at local. Therefore, it can be observed that the offloading ratio increases with the number of devices increasing.

Fig. 10 illustrates the changes of average latency with different numbers of devices. It is observed that when the number of devices increases, the total training latency increases since the total computing capacity in MEC servers and cloud server is set to be constant. When the number of devices continues to increase, the gap between traditional learning scheme and the proposed scheme becomes smaller. Because the computation resource in MEC allocated to devices becomes smaller while the computing ability of devices is constant. Latency performance of centralized learning scheme also becomes worse with the number of devices increasing because the communication and computation resources are both limited. That means each device will be allocated smaller bandwidth and computation resources. When the number of devices is more than 10, the proposed scheme still has the best latency performance than other three schemes of which performance is similar with each other.

Fig. 11 shows the changes of average training latency with different resource allocation methods. We define fractional resource allocation as ’FRA’ which means bandwidth and computation capacity are allocated according to the offloading ratio. Equal resource allocation is defined as ’ERA’ which means each device gets average reource. It can be observed that the training latency linearly increases with the number of data samples increasing for all schemes. As shown in the figure, FRA has the same performance as the optimized solution and ERA has a worse latency performance than

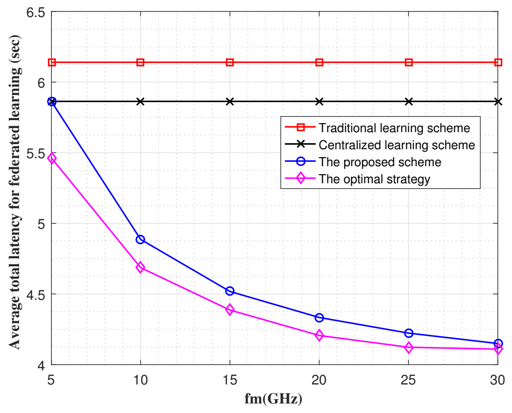


Fig. 12. Average training latency versus the computation resource of MEC server

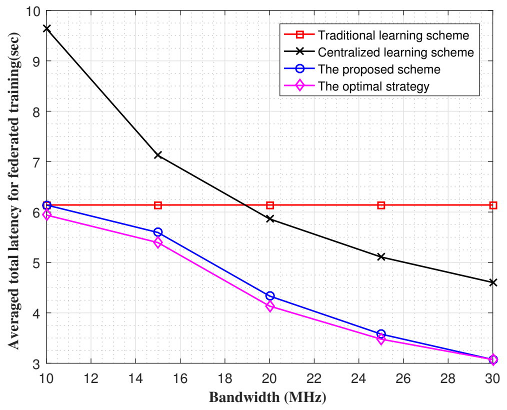


Fig. 13. Average training latency versus the bandwidth of APs

the FRA. This is because the total latency of multi-layer federated learning depends on the maximum value of each layer. Allocating resource based on the value of offloading ratio can reduce the difference of latency among devices to minimize the maximum value of each layer.

B. Performance of multi-layer MEC assisted architecture with the constraint of indivisible tasks

In this section, we evaluate the latency performance of the proposed scheme with the constraint of invisible tasks. Traditional learning scheme and centralized learning scheme are still considered as comparisons. The optimal strategy obtained by exhausted searching is additionally added as a benchmark.

Fig. 12 and Fig.13 illustrate the changes of average training latency with different computation resource and communication resource of MEC servers when there is a constraint of binary offloading decisions. It can be seen from Fig.12 and Fig.13 that when the bandwidth or CPU resource is limited, latency performance of the proposed scheme is the same as the traditional local training since devices tend to train locally to save latency. It can be observed from Fig.13 that except the traditional learning scheme which is affected by bandwidth slightly, other schemes have better performances with larger bandwidth. The optimal strategy obtained by exhausted searching can achieve the best training latency performance. The multi-layer learning scheme with the proposed resource allocation strategy has a close performance to the optimal strategy. However, when the number of devices increases, the evaluation time to obtain the final solution by exhausted searching will increase exponentially because of the 3^J computing complexity. Therefore, we can get the approximate optimal resource allocation strategy by saving evaluation time.

VI. Conclusion

In this paper, a MEC assisted multi-layer federated learning architecture for IoT devices has been investigated to achieve an efficient joint training procedure. In the proposed learning scheme, the offloading decisions, the computation resource allocation and the bandwidth have been optimized jointly to minimize the total federated learning latency. In addition, we have investigated an optimum resource allocation scheduling with the constraints of indivisible tasks. Simulation results have verified the effectiveness of the proposed training scheme. The following conclusions can be drawn:

- The proposed multi-layer MEC assisted federated learning architecture can significantly improve the performance of learning latency. Furthermore, our proposed learning architecture can achieve more efficient training process than the traditional federated learning method under the same accuracy requirement.
- Resource allocation strategy from our proposed solution based on SDR method has significant improvement on latency performance. Under a limited total computation resource, there exists a reasonable resource allocation ratio between MEC servers and the cloud server to achieve the best latency performance.
- When considering the scenario with the constraint of indivisible tasks, the resource allocation from the recovery solution with shorter evaluation time has the close performance to the exausted searching method.

References

[1] G. Zhu, D. Liu, Y. Du, C. You, J. Zhang, and K. Huang, "Toward an intelligent edge: Wireless communication meets machine learning," *IEEE communications magazine*, vol. 58, no. 1, pp. 19–25, 2020.
[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–44, 05 2015.
[3] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.
[4] P. Li, J. Li, Z. Huang, T. Li, C.-Z. Gao, S.-M. Yiu, and K. Chen, "Multi-key privacy-preserving deep learning in cloud computing," *Future Generation Computer Systems*, vol. 74, pp. 76–85, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17302005
[5] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 4427–4437.
[6] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Network*, vol. 33, no. 5, pp. 156–165, 2019.
[7] W. Saad, M. Bennis, and M. Chen, "A vision of 6g wireless systems: Applications, trends, technologies, and open research problems," *IEEE Network*, vol. 34, no. 3, pp. 134–142, 2020.

[8] Y. Zhou, Q. Ye, and J. Lv, "Communication-efficient federated learning with compensated overlap-fedavg," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 1, pp. 192–205, 2022.

[9] Z. Yang, W. Bao, D. Yuan, N. H. Tran, and A. Y. Zomaya, "Federated learning with nesterov accelerated gradient momentum method," *arXiv preprint arXiv:2009.08716*, 2020.

[10] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained on-device federated learning," *IEEE Communications Letters*, vol. 24, no. 6, pp. 1279–1283, 2020.

[11] S. Luo, X. Chen, Z. Zhou, X. Chen, and W. Wu, "Incentive-aware micro computing cluster formation for cooperative fog computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 4, pp. 2643–2657, 2020.

[12] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2015.

[13] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge ai: On-demand accelerating deep neural network inference via edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 447–457, 2019.

[14] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2016.

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

[16] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[17] J. Ren, G. Yu, and G. Ding, "Accelerating dnn training in wireless federated edge learning systems," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 219–232, 2020.

[18] G. Zhu, Y. Wang, and K. Huang, "Broadband analog aggregation for low-latency federated edge learning," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 491–506, 2020.

[19] Q. Zeng, Y. Du, K. Huang, and K. K. Leung, "Energy-efficient radio resource allocation for federated edge learning," in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2020, pp. 1–6.

[20] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 269–283, 2021.

[21] C. W. Zaw, S. R. Pandey, K. Kim, and C. S. Hong, "Energy-aware resource management for federated learning in multi-access edge computing systems," *IEEE Access*, vol. 9, pp. 34 938–34 950, 2021.

[22] W. Xu, Z. Yang, D. W. K. Ng, M. Levorato, Y. Eldar, and M. Debbah, "Edge learning for b5g networks with distributed signal processing: Semantic communication, edge computing, and wireless sensing," 06 2022.

[23] N. Yoshida, T. Nishio, M. Morikura, K. Yamamoto, and R. Yonetani, "Hybrid-fl for wireless networks: Cooperative learning mechanism using non-iid data," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–7.

[24] Y.-W. Chu, S. Hosseinalipour, E. Tenorio, L. Cruz Castro, K. Douglas, A. Lan, and C. Brinton, "Multi-layer personalized federated learning for mitigating biases in student predictive analytics," 12 2022.

[25] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Transactions on Wireless Communications*, 2020.

[26] J. Konečnỳ, Z. Qu, and P. Richtárik, "Semi-stochastic coordinate descent," *optimization Methods and Software*, vol. 32.

[27] C. Ma, J. Konečnỳ, M. Jaggi, V. Smith, M. I. Jordan, P. Richtárik, and M. Takáč, "Distributed optimization with arbitrary local solvers," *Optimization Methods and Software*, vol. 32, no. 4, pp. 813–848, 2017.

[28] J. Fang and A. Ma, "Iot application modules placement and dynamic task processing in edge-cloud computing," *IEEE Internet of Things Journal*, 2020.

[29] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54. PMLR, 20–22 Apr 2017, pp. 1273–1282. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

[30] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[31] R. Andonov, V. Poirriez, and S. Rajopadhye, "Unbounded knapsack problem: Dynamic programming revisited," *European Journal of Operational Research*, vol. 123, no. 2, pp. 394–407, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221799002659

[32] Z.-Q. Luo, W.-K. Ma, A. M.-C. So, Y. Ye, and S. Zhang, "Semidefinite relaxation of quadratic optimization problems," *IEEE Signal Processing Magazine*, vol. 27, no. 3, pp. 20–34, 2010.

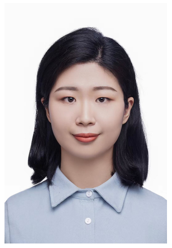
[33] A. Mobasher, M. Taherzadeh, R. Sotirov, and A. Khandani, "A near maximum likelihood decoding algorithm for mimo systems based on semi-definite programming," in *Proceedings. International Symposium on Information Theory, 2005. ISIT 2005.*, 2005, pp. 1686–1690.

[34] W.-K. Ma, C.-C. Su, J. Jalden, T.-H. Chang, and C.-Y. Chi, "The equivalence of semidefinite relaxation mimo detectors for higher-order qam," *IEEE Journal of Selected Topics in Signal Processing*, vol. 3, no. 6, pp. 1038–1052, 2009.

[35] Y. Nesterov and A. Nemirovskii, *Interior-point polynomial algorithms in convex programming*. SIAM, 1994.

[36] G. Zhang, S. Zhang, W. Zhang, Z. Shen, and L. Wang, "Joint service caching, computation offloading and resource allocation in mobile edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 20, no. 8, pp. 5288–5300, 2021.

Huibo Li received the BS degree, MS degree and Ph.D. degree in the School of Mechatronical Engineering from Beijing Institute of Technology in 2016, 2019 and 2023, respectively. Her research interests include wireless network simulation and emulation, mobile edge computing and resource management in wireless systems.



Yijin Pan is an associate professor in the School of Information Science and Engineering at Southeast University. She was selected as a Newton Fellow by the Royal Society of the United Kingdom from 2019 to 2021. Dr. Pan Yijin serves as a reviewer and the TPC member for prestigious international journals and conferences including IEEE Transactions, Globecom, and ICC, etc. Her research focuses on key technologies for the future communication networks, especially for the edge intelligence enhanced communication schemes.

Huiling Zhu received the B.S degree from Xidian University, China, and the Ph.D. degree from Tsinghua University, China. She is currently a Reader (Associate Professor) in the School of Engineering, University of Kent, United Kingdom. Her research interests are in the area of wireless communications. She was holding European Commission Marie Curie Fellowship from 2014 to 2016. She received the best paper award from IEEE Globecom 2011. She was Symposium Co-Chair for IEEE Globecom 2015 and IEEE ICC 2018, and Track Co-Chair of IEEE VTC2016-Spring and VTC2018-Spring. Currently, she serves as an Editor for IEEE Transactions on Vehicular Technology.

Peng Gong received the BS degree in Mechatronical Engineering from Beijing Institute of Technology, Beijing, China, in 2004, and the MS and Ph.D. degrees from the Inha University, Korea, in 2006 and 2010, respectively. In July 2010, he joined the School of Mechatronical Engineering, Beijing Institute of Technology, China. His research interests include link/system level performance evaluation and radio resource management in wireless systems, information security, and the next generation wireless systems such as 3GPP LTE, UWB, MIMO, Cognitive radio and so on.

Jiangzhou Wang (Fellow, IEEE) is a Professor with the University of Kent, U.K. He has published more than 400 papers and four books. His research focuses on mobile communications. He was a recipient of the 2022 IEEE Communications Society Leonard G. Abraham Prize and IEEE Globecom2012 Best Paper Award. He was the Technical Program Chair of the 2019 IEEE International Conference on Communications (ICC2019), Shanghai, Executive Chair of the IEEE ICC2015, London, and Technical Program Chair of the IEEE WCNC2013. He is/was the editor of a number of international journals, including IEEE Transactions on Communications from 1998 to 2013. Professor Wang is a Fellow of the Royal Academy of Engineering, U.K., Fellow of the IEEE, and Fellow of the IET.