Device Scheduling and Bandwidth Allocation for Federated Learning over Wireless Networks

Tinghao Zhang, Kwok-Yan Lam, Jun Zhao
School of Computer Science and Engineering
Nanyang Technological University
Singapore

***Abstract*—Federated Learning (FL) has been widely used to train shared machine learning models while addressing the privacy concerns. When deployed in wireless networks, bandwidth resources limitation is a key issue, thereby necessitating device scheduling and bandwidth allocation. It is challenging to carry out device scheduling due to the large combinatorial search space. Besides, the heterogeneous computing capabilities and uncertain channel states of wireless devices complicate the design of a bandwidth allocation method. In this paper, we propose a joint device scheduling and bandwidth allocation framework for implementing FL in wireless networks. Specifically, deep reinforcement learning (DRL) is employed to conduct device scheduling. To this end, the state space, action space, and reward function of DRL are carefully defined for a typical FL system. Long short-term memory (LSTM) is adopted as the DRL agent to analyze the sequential input data. Given the scheduled devices of each global iteration, the proposed bandwidth allocation method aims to minimize the weighted sum of the time delay and energy consumption. Numerical experiments on both independent and identically distributed (IID) and non-IID datasets demonstrate that the proposed framework enables FL to reach the desired accuracy with low time delay and energy consumption.**

I. Introduction

With the advancement of Internet-of-Things (IoT) and next generation mobile communication technologies, a variety of wireless systems, such as space-air-ground integrated network (SAGIN) [1], [2], unmanned aerial vehicles (UAV) [3], [4], and Intelligent Transportation Systems [5], [6], have emerged to support numerous wireless devices to train machine learning models. As massive amounts of training data are collected for ML model training, traditional wireless networks typically employ some centralized methods [7] to implement model training. Specifically, all the gathered data are transmitted to a remote cloud server for training ML models. However, centralized training is unscalable and ineffective for practical applications because of the high communication latency. Besides, the need to upload raw local data from devices is likely to raise privacy concerns. To address these problems, a distributed learning method called Federated Learning (FL) has been proposed [8]. In FL, the wireless devices, referred hereinafter as local devices, train the local models on their local datasets in parallel. After several iterations, the local devices transmit the model parameters instead of the local datasets to the remote server. Upon receiving these model parameters, the centralized server conducts model aggregation to generate a new global model and broadcasts the global model to the local devices for the next iteration. Since FL does not need the transmission of local data, user privacy is better protected, and the time delay of data transmission is reduced.

Although FL has the potential to attain remarkable improvements, it is challenging to implement FL in practical wireless networks [9]. Specifically, wireless networks usually involve a large number of local devices to collect ample training data. In this case, executing the FL algorithm is still faced with prohibitive communication overheads due to the limited bandwidth resources available to local devices [10]. Scheduling a subset of local devices to take part in local update will allow FL to significantly mitigate bandwidth deficiency. Despite its advantage, device scheduling (i.e. optimally selecting the subset of devices to participate in FL) brings about several problems for executing FL. For example, scheduling fewer devices for update inevitably affects the convergence of FL as less training data are involved at each round. As a result, the FL model will require more communication rounds and, worse yet, may not reach the desired accuracy. More importantly, though scheduling certain devices may considerably boost the accuracy of the FL model, it may lead to more serious time delay and higher energy cost because of the uncertainty of their channel states. Therefore, the device scheduling method is expected to be able to balance the tradeoff between the update effectiveness and the system cost (i.e., energy consumption and time delay), which poses great challenges to the design of FL scheduling methods. Moreover, the deployment of FL necessitates an effective bandwidth allocation method. Unfortunately, the channel state and computational capability of the scheduled devices can vary significantly, which complicates the design of the bandwidth allocation method.

In this paper, we address the aforementioned challenges by presenting a joint device scheduling and bandwidth allocation framework. The goal of device scheduling in FL is finding an optimal subset of devices from a finite set of devices to minimize an objective function while meeting the bandwidth constraint, which is essentially a combinatorial optimization problem. As it is difficult to directly solve the scheduling problem, we resort to a data-driven approach for carrying out the scheduling process. After device scheduling, the proposed bandwidth allocation approach is performed for the scheduled devices. The main contributions are summarized as follows:

1) We study the mechanism of FL and formulate an optimization problem to minimize the weighted sum of the energy consumption and time delay for training the entire FL algorithm. A joint device scheduling and bandwidth allocation framework is proposed to solve this problem.
2) We propose a deep reinforcement learning (DRL)-based device scheduling method for FL. Specifically, we design state space, action space, and a reward function to implement DRL in FL. Besides, we employ long short-term memory (LSTM) as the DRL agent and discuss

the motivation behind the LSTM-based agent. Finally, we describe the workflow of training the LSTM-based agent for device scheduling.

3) We design a bandwidth allocation method to minimize the weighted sum of the energy consumption and time delay under one global iteration.
4) The proposed device scheduling and bandwidth allocation framework is evaluated on both independent and identically distributed (IID) and non-IID datasets. The experimental results show that the proposed framework outperforms other benchmarks w.r.t. minimizing energy consumption and time delay.

The rest of the paper is organized as follows. The system model and problem formulation are defined in Section II. The deep reinforcement learning (DRL)-based device scheduling method is described in Section III. Experimental results are provided in Section IV followed by the conclusion in Section V. Details for the bandwidth allocation can be found in our journal version [11].

II. System Model

Consider an FL system with N local devices indexed by $\mathcal{N} = \{1, \cdots, N\}$ and one remote server. Local device $n \in \mathcal{N}$ has a local dataset $\mathcal{D}_n = \{\boldsymbol{x}_\kappa, y_\kappa\}_{\kappa=1}^{D_n}$, where $\boldsymbol{x}_\kappa$ and y_κ are the κ-th input sample and output label, respectively. Let $\boldsymbol{w}$ be the model parameters of the global model. The local model n at the i-th global iteration is defined as $\boldsymbol{w}_n^i$.

A. FL Model

We define $\xi(\boldsymbol{w}, \boldsymbol{x}_j, y_j)$ as the loss function of one sample data. The total loss function of local device n is $F_n(\boldsymbol{w}) = \frac{1}{D_n}\sum_{\kappa=1}^{D_n} \xi(\boldsymbol{w}, \boldsymbol{x}_j, y_j)$. The FL system aims to obtain $\boldsymbol{w}$ that minimizes the loss function on the whole dataset. The problem of the training process is expressed as

$$\min_{\boldsymbol{w}} F(\boldsymbol{w}) \triangleq \frac{1}{D}\sum_{n=1}^{N} D_n F_n(\boldsymbol{w}), \tag{1}$$

where $D = \sum_{n=1}^{N} D_n$. FL iterates between three steps to solve problem (1).

1) **Device Scheduling**: The remote server schedules M_i devices represented by the set $\mathcal{M}_i \in \mathcal{N}$ to participate in the i-th global iteration. For the local devices that are not scheduled, we assume that their local models remain unchanged. After the server broadcasts the global model, the local models $\boldsymbol{w}_n^i$ are derived as

$$\boldsymbol{w}_n^i = \begin{cases} \boldsymbol{w}^i, & n \in \mathcal{M}_i \\ \boldsymbol{w}_n^{i-1}, & n \in \complement_{\mathcal{N}}\mathcal{M}_i, \end{cases} \tag{2}$$

where $\complement_{\mathcal{N}}\mathcal{M}_i$ represents a complement of the set $\mathcal{M}_i$ with a universal set $\mathcal{N}$. In this paper, we define $\boldsymbol{w}_n^0 = 0,\ \forall n \in \mathcal{N}$.

2) **Local Update**: Local device $n \in \mathcal{M}_i$ receives $\boldsymbol{w}^i$ (i.e., $\boldsymbol{w}_n^i(0) = \boldsymbol{w}^i$) and performs local update using the gradient descent algorithm:

$$\boldsymbol{w}_n^i(k) = \boldsymbol{w}_n^i(k-1) - \alpha \nabla F_n(\boldsymbol{w}_n^i(k-1)),\ \forall n \in \mathcal{M}_i, \tag{3}$$

where $k = 1, \cdots, K$, K denotes the number of local iterations, and α is the learning rate.

3) **Global Aggregation**: After local update, local device $n \in \mathcal{M}_i$ transmits model parameters $\boldsymbol{w}_n^i(K)$ to the server. The server aggregates them as follows

$$\boldsymbol{w}^{i+1} = \frac{\sum_{n\in\mathcal{M}_i} D_n \boldsymbol{w}_n^i(K)}{\sum_{n\in\mathcal{M}_i} D_n}. \tag{4}$$

FL reaches convergence when the accuracy of the global model achieves the target accuracy.

B. Energy consumption and time delay of FL

In this section, the energy consumption and time delay for training the FL model are obtained. We define C_n as the number of CPU cycles for processing one data sample at local device n. Denote the CPU frequency of local device n by f_n. Thus, the time delay t_n^{cmp} and energy consumption e_n^{cmp} for local device n to process D_n data samples are:

$$t_n^{\text{cmp}} = \frac{KC_nD_n}{f_n}, \tag{5}$$

$$e_n^{\text{cmp}} = \beta K C_n D_n f_n^2, \tag{6}$$

where β denotes the effective switched capacitance of local device n's computing chipset.

In this work, the model parameters are transmitted via a frequency domain multiple access (FDMA[1]) protocol. The achievable transmission rate of local device n is defined as

$$\lambda_n = b_n \log_2\left(1 + \frac{\overline{g}_n q_n}{N_0 b_n}\right), \tag{7}$$

where b_n is the bandwidth resources allocated to local device n, q_n is the average transmit power of local device n, $\overline{g}_n$ is the average channel gain between local device n and the server during the training period of FL, and N_0 is the background noise. As all the local devices share the same model architecture, the sizes of the model parameters are the same and can be represented by z. Therefore, the time delay t_n^{com} and energy consumption e_n^{com} for local device n to transmit the model parameters are:

$$t_n^{\text{com}} = \frac{z}{\lambda_n} = \frac{z}{b_n \log_2\left(1 + \frac{\overline{g}_n q_n}{N_0 b_n}\right)}, \tag{8}$$

$$e_n^{\text{com}} = t_n^{\text{com}} q_n, \tag{9}$$

According to the existing works [12]–[21], the latency and the energy consumption for model broadcasting are neglectable. This is because the average transmit power of the remote server is much higher than that of the users, and the remote server will utilize the whole downlink bandwidth when broadcasting the model. As a result, the time delay T and energy consumption E for training the entire FL algorithm are:

$$T_i = \max_{n\in\mathcal{M}_i}\left\{t_n^{\text{cmp}} + t_n^{\text{com}}\right\},\ T = \sum_{i=1}^{I} T_i, \tag{10}$$

$$E_i = \sum_{n\in\mathcal{M}_i}\left(e_n^{\text{cmp}} + e_n^{\text{com}}\right),\ E = \sum_{i=1}^{I} E_i. \tag{11}$$

where I is the total number of the global iterations for FL to achieve the target accuracy, T_i and E_i are the time delay and energy consumption of FL at the i-th iteration, respectively.

C. Problem Formulation

Our work aims to minimize the weighted sum of the time delay T and energy consumption E. The optimization

[1]If considering the interference between different frequencies, guard bands can be used to avoid the interference. In this situation, the available bandwidth B is determined by subtracting the bandwidth resources used for guard bands from the total bandwidth of the remote server.

problem is formulated as follows

$$\min_{\boldsymbol{\mathcal{B}},I,\boldsymbol{\mathcal{M}}} \; E+\gamma T \tag{12}$$

$$\text{s.t.} \quad \sum_{n\in\mathcal{M}_i} b_n \le B, \tag{12a}$$

$$Acc_i < Acc^{\text{target}}, i=1,\cdots,I-1, \tag{12b}$$

$$Acc_I \ge Acc^{\text{target}}. \tag{12c}$$

where $\boldsymbol{\mathcal{B}} = \left[\boldsymbol{b}_i \middle| i=1,\cdots,I\right]$, $\boldsymbol{b}_i = \left[b_n \middle| n\in\mathcal{M}_i\right]$, $\boldsymbol{\mathcal{M}} = \left[\mathcal{M}_i \middle| i=1,\cdots,I\right]$. B is total bandwidth. γ is a weight to indicate the tradeoff between the energy consumption and time delay, Acc_i is the accuracy of the global model at the i-th global iteration, and Acc^{target} is the target accuracy. (12a) indicates the bandwidth constraint. (12b) and (12c) denote that the accuracy of the global model will not exceed the target accuracy until the I-th global iteration. Note that (12b) and (12c) may be excessively stringent in some practical scenarios, and the average accuracy of few last iterations can be more suitable. In this paper, we assume that the number of selected devices is fixed at each iteration (i.e., $M_1=\cdots=M_I=M$).

Due to the set $\boldsymbol{\mathcal{M}}$, solving problem (12) involves combinatorial optimization. Besides, the bandwidth constraint (12a) makes problem (12) more intractable. Therefore, we propose a novel FL framework to deal with this optimization problem. Specifically, a DRL agent is used to carry out device scheduling, which considers the one-round system cost (i.e., $E_i+\gamma T_i$) as well as the update effectiveness (i.e., $Acc_i - Acc_{i-1}$). After obtaining the set of the selected devices $\mathcal{M}_i$, a spectrum allocation optimization method is adopted to determine the allocated bandwidth of each selected device. The proposed framework enables FL to achieve a lower global system cost (i.e., $E+\gamma T$) for training the entire FL algorithm.

III. Deep Reinforcement Learning based Device Scheduling

In this section, we develop a deep reinforcement learning (DRL) model to carry out device scheduling in FL. First of all, we provide the preliminaries of DRL and design a DRL environment for FL. Next, we adopt a long short-term memory (LSTM)-based agent and describe the training algorithm of DRL. Finally, we present the workflow of the DRL-based device scheduling method. The proposed method allows the FL model to achieve the target accuracy while significantly minimizing objective value (12).

A. Design of DRL Environment

DRL can be used to solve the optimization problem that follows a Markov decision process (MDP). At a time slot i, the DRL agent receives a state $\boldsymbol{s}_i$ and conducts an action $\boldsymbol{a}_i \sim \pi_\theta(\boldsymbol{a}_i|\boldsymbol{s}_i)$, where π is a policy, θ is the parameters of π. An immediate reward r_i is obtained after the environment state transforms from $\boldsymbol{s}_i$ to $\boldsymbol{s}_{i+1}$. The goal of DRL is to train the policy π_θ to maximize the discounted cumulative reward $R_i = \sum_{l=0}^{\infty}\varphi^l r_{i+l}$, where $\varphi\in(0,1]$ is a discounted factor. Besides, the state-action value function $Q(\boldsymbol{s}_i,\boldsymbol{a}_i)$ and the state-value function $V(\boldsymbol{s}_i)$ in DRL are represented as [22]

$$Q(\boldsymbol{s}_i,\boldsymbol{a}_i) = \mathbb{E}_{\boldsymbol{s}_{i+1:\infty},\boldsymbol{a}_{i+1:\infty}}\left[\sum_{l=0}^{\infty}\varphi^l r_{i+l}|\boldsymbol{s}_i,\boldsymbol{a}_i\right], \tag{13}$$

$$V(\boldsymbol{s}_i) = \mathbb{E}_{\boldsymbol{a}_i}\left[Q(\boldsymbol{s}_i,\boldsymbol{a}_i)|\boldsymbol{s}_i\right]. \tag{14}$$

where the expectations $\mathbb{E}[\cdot]$ are related to state $\boldsymbol{s}_i$ and the actions produced by the policy π, $\boldsymbol{s}_{i+1:\infty}$ denotes a state trajectory starting from the time slot $i+1$.

Device scheduling in FL is a sequential decision-making process and can be further modeled as MDP. We design the state space, action space, and reward function for executing DRL-based device scheduling.

State space: State $\boldsymbol{s}_i$ should contain the information that relates to update effectiveness and the system cost. On one hand, the system cost depends on b_n and device characteristics such as f_n, q_n, $\overline{g}_n$, C_n, and D_n according to Section II-B. As bandwidth allocation will be performed after device scheduling, b_n is not available during constructing $\boldsymbol{s}_i$. In addition, the order of magnitudes of $\overline{g}_n$ ranges from millionth to quadrillionth. Including $\overline{g}_n$ in $\boldsymbol{s}_i$ may deteriorate the training process of DRL. Since $\overline{g}_n$ is affected by the distance between local device n and the remote server (represented as d_n), we use d_n to approximate the channel states. On the other hand, the weight divergence between the global model $\boldsymbol{w}^i$ and local device $\boldsymbol{w}_n^{i-1}$ (i.e., $\left\{\|\boldsymbol{w}^i-\boldsymbol{w}_n^{i-1}\|\middle|n\in\mathcal{N}\right\}$) is used to estimate the update effectiveness of each local model [23]. As a result, $\boldsymbol{s}_i$ is defined as

$$\boldsymbol{s}_i = \left\{\boldsymbol{\eta}_n^i\middle|n\in\mathcal{N}\right\}. \tag{15}$$

where $\boldsymbol{\eta}_n^i = \left(\|\boldsymbol{w}^i-\boldsymbol{w}_n^{i-1}\|, f_n, q_n, d_n, D_n, C_n\right)^{\mathrm{T}}$. Note that FL requires I iterations to reach the target accuracy Acc^{target}. Therefore, the terminal state of DRL in this paper will be $\boldsymbol{s}_I$.

Action space: The DRL agent is anticipated to select M devices from N devices. If adopting discrete action, the action space will be $\binom{N}{M}$. It is difficult to train the DRL agent with such a large action space. Therefore, the DRL agent employs a continuous action space. Specifically, the action $\boldsymbol{a}_i$ is a N-dimensional probability vector as follows

$$\boldsymbol{a}_i = \left\{a_n^i\middle|n\in\mathcal{N}\right\}, \tag{16}$$

where a_n^i indicates the probability that local device n is scheduled at the i-th global iteration, and $\sum_{n=1}^{N}a_n^i = 1$. Moreover, the proposed scheduling method follows sampling without replacement in this paper. As a result, the probability vector of the devices at the m-th scheduling round is derived as follows

$$p_n^i(m) = \begin{cases} 0, & n\in\mathcal{M}_i(m-1), \\ \frac{a_n^i}{1-\sum_{n'\in\mathcal{M}_i(m-1)}a_{n'}^i}, & n\notin\mathcal{M}_i(m-1), \end{cases} \tag{17}$$

where $m=1,\cdots,M$, $\mathcal{M}_i(m)$ denotes the union of the sets of scheduled devices at scheduling rounds $1,2,\ldots,m$, and $\mathcal{M}_i(0)=\emptyset$. One device will be scheduled per scheduling round based on (17) until $m=M$[2].

Reward Function: The reward function r_i guides the training process of DRL. In FL, scheduling certain devices may significantly boost the accuracy while resulting in high $E_i+\gamma T_i$, which will still result in high $E+\gamma T$. In this case, we define the reward function as follows

$$r_i = -(E_i+\gamma T_i). \tag{18}$$

Intuitively, the DRL agent aims to minimize the one-round system cost. Moreover, the cumulative reward for the DRL

[2]In practical situations, we can use the function $choice(\cdot)$ from the Numpy library [24] or the function $multinomial(\cdot)$ from the Pytorch library [25] to finish this process with ultra-low latency.

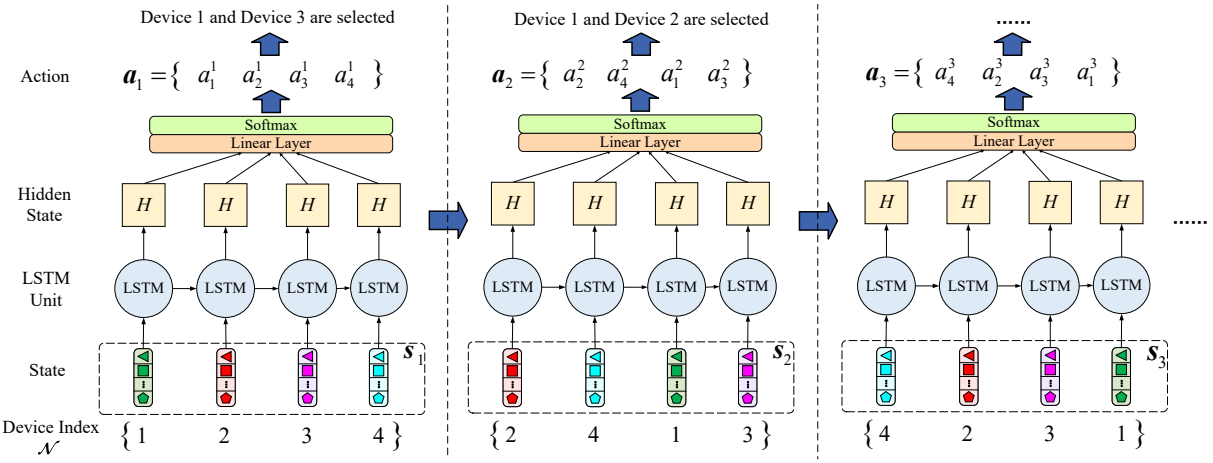


Fig. 1: Workflow of the LSTM-based agent with $N = 4$ devices. Two devices are scheduled per global iteration (i.e., $M = 2$). At the initial iteration, the state $\boldsymbol{s}_1$ sorts the feature vectors $\boldsymbol{\eta}_1^1 \cdots \boldsymbol{\eta}_4^1$ based on the device index. The output of LSTM $\boldsymbol{a}_1$ is a probability vector, and the server schedules local device 1 and 3 based on (17), which is elaborated in Section III-A. At the 2-nd iteration, the feature vectors of the selected devices (i.e., local device 1 and 3) are placed at the end of the sequence. Then, the DRL agent generates $\boldsymbol{a}_2$ after receiving $\boldsymbol{s}_2$, and the server selects local device 1 and 2. Note that the order of $a_1^i \cdots a_4^i$ in $\boldsymbol{a}_i$ is always consistent with the order of the feature vectors in $\boldsymbol{s}_i$. After two global iterations, local device 1 has been scheduled twice, local device 2 and 3 have been selected once, and local device 4 has not been selected yet. At the 3-rd iteration, the feature vectors in $\boldsymbol{s}_3$ along with $a_1^3 \cdots a_4^3$ in $\boldsymbol{a}_3$ are sorted based on the scheduling frequency.

agent to maximize is $\sum_{i=1}^{I} r_i = -(E + \gamma T)$, which is in line with the goal of optimization problem (12). Note that our work sets the discount factor φ to be 1. From this perspective, reward function (18) encourages the DRL agent to consider update effectiveness. Specifically, the DRL agent will select the devices that help the FL model achieve the target accuracy swiftly (i.e., reduce the number of global iteration I).

B. LSTM-based Agent

The DRL agent is usually a deep neural network that receives state information $\boldsymbol{s}_i$ and generates an action $\boldsymbol{a}_i$. Based on the DRL environment in Section III-A, it is intuitive to employ a multilayer perceptron (MLP) as the DRL agent. Under this situation, the state $\boldsymbol{s}_i$ will be flattened into a vector to feed into the agent. However, it is impractical to train the MLP-based agent to carry out device scheduling for FL. For instance, some parameters in $\boldsymbol{s}_i$ (e.g., d_n, D_n, etc) are fixed and thus redundant for the DRL training. Besides, the frequency of the devices being scheduled is a critical factor that affects the convergence of FL. If some devices are repeatedly selected while some other devices are rarely scheduled, FL will require more iterations (i.e., a larger I) to achieve the target accuracy as it fails to learn from a variety of training data. Whereas, the flattened state $\boldsymbol{s}_i$ does not reflect such scheduling frequency to the MLP-based agent.

In this paper, we take two steps to solve the above issues. Firstly, instead of flattening $\boldsymbol{s}_i$ into a single vector, we treat $\boldsymbol{s}_i$ as a sequence that contains N vectors, where the n-th vector $\boldsymbol{\eta}_n^i$ is regarded as a feature vector of local device n. More importantly, these N vectors are sorted in ascending order based on the number of times that the local devices have been scheduled. As a result, $\boldsymbol{s}_i$ can reflect the scheduling frequency of the local devices. Secondly, we employ long short-term memory (LSTM) as the DRL agent to deal with the sequence data $\boldsymbol{s}_i$. Since the output of LSTM depends on the values and the order of input vectors, the fixed parameters d_n and D_n still make contributions to DRL training. Fig. 1 presents an example of how the LSTM-based agent handles the sequence data $\boldsymbol{s}_i$. To sort the vectors $\boldsymbol{\eta}_n^i$ in $\boldsymbol{s}_i$ during the practical implementation, we introduce an array that records how many times each local device is selected. Note that the local devices with the same scheduling frequency are sorted by the device index. Implementation details for training the DRL agent are provided in Algorithm 1 and the last paragraph of Section III-C.

C. Training Algorithm of DRL

This paper adopts an actor-critic framework to train the DRL agent. The actor network with the parameters θ determines which action should be taken, while the critic network with the parameters ϕ evaluates the selected action. For the actor network, the gradient estimator is derived as follows [26]

$$\nabla J(\theta) = \mathbb{E}_i \left[\nabla_\theta \log \pi_\theta(\boldsymbol{a}_i | \boldsymbol{s}_i) A_i \right], \tag{19}$$

where $\mathbb{E}_i[\cdot]$ denotes the empirical average over a finite batch of samples, in an algorithm that alternates between sampling and optimization, $A_i = Q(\boldsymbol{s}_i, \boldsymbol{a}_i) - V(\boldsymbol{s}_i)$ is an advantage function that measures the advantage of taking the action $\boldsymbol{a}_i$ under the state $\boldsymbol{s}_i$. As the critic network can only estimate either $Q(\boldsymbol{s}_i, \boldsymbol{a}_i)$ or $V(\boldsymbol{s}_i)$, the advantage function is usually modified as $A_i = r_i + \varphi V(\boldsymbol{s}_{i+1}) - V(\boldsymbol{s}_i)$. As a result, the critic network is updated to minimize the square error between the estimated state value and its true value.

$$L(\phi) = \mathbb{E}_i \left[\left(r_i + \varphi V(\boldsymbol{s}_{i+1}) - V(\boldsymbol{s}_i) \right)^2 \right], \tag{20}$$

However, the advantage function $A_i = r_i + \varphi V(\boldsymbol{s}_{i+1}) - V(\boldsymbol{s}_i)$ is affected by the value function estimator. If the estimator exhibits bias, A_i will be similarly biased. Including extra time steps in A_i can reduce the bias while increasing the variance. To balance the tradeoff between bias and variance, we employ a generalized advantage estimator (GAE) [26]

$$A_i^{\text{GAE}} = \sum_{l=0}^{\infty} (\varphi\nu)^l \delta_{i+l}, \tag{21}$$

where $\delta_i = r_i + \varphi V(\boldsymbol{s}_{i+1}) - V(\boldsymbol{s}_i)$, $\nu \in [0, 1]$. GAE introduces a parameter ν that can balance the bias-variance tradeoff.

Algorithm 1 Training process of the DRL agent for device scheduling

Input: Set of devices $\mathcal{N} = \{1, 2, \cdots, N\}$; number of scheduled devices M; replay buffer Ω; number of steps per update O; number of epochs for updating the DRL agent W; number of minibatches per epoch Y; maximum number of episodes Γ.
Output: The actor π_θ

1: All the local devices send the local information (f_n, q_n, d_n, D_n) to the server
2: Initialize the actor parameters θ and critic parameters ϕ
3: **for** $e = 1, ..., \Gamma \cdots$ **do**
4: $i = 1$
5: Initialize the global model $\boldsymbol{w}^i$ and a N-dimensional array with all zeros $count\_arr = [0, \cdots, 0]$
6: Initialize $\boldsymbol{s}_i$ based on (15)
7: **repeat**
8: $\boldsymbol{a}_i$ is generated based on $\pi_\theta(\cdot|\boldsymbol{s}_i)$
9: **for** $m = 1$ to M **do**
10: Calculate $\left\{p_n^i(m)\middle|n \in \mathcal{N}\right\}$ based on (17)
11: Randomly schedule one device based on $\left\{p_n^i(m)\middle|n \in \mathcal{N}\right\}$
12: **end for**
13: **for** each local device $n \in \mathcal{M}_i$ **do**
14: $count\_arr[n] = count\_arr[n] + 1$
15: Download $\boldsymbol{w}^i$ from the server and perform local update based on (3)
16: Transmit local model $\boldsymbol{w}_n^i$ to the server
17: **end for**
18: The server conducts global aggregation based on (4) and obtains r_i
19: Obtain $\boldsymbol{s}_{i+1}$ based on (15), where the vectors in (15) (i.e., $\left\{\boldsymbol{\eta}_n^i\middle|n \in \mathcal{N}\right\}$) are sorted order in ascending based on $count\_arr$
20: put $(\boldsymbol{s}_i, \boldsymbol{a}_i, r_i, \boldsymbol{s}_{i+1})$ into the replay buffer Ω
21: **if** $\zeta < O$ **then**
22: $\zeta = \zeta + 1$
23: **else**
24: **for** $\chi = 1, \cdots, W$ **do**
25: Obtain Y samples from Ω to update θ and ϕ based on (22) and (23)
26: **end for**
27: $\theta_{\text{old}} \leftarrow \theta$
28: Clear all the data in Ω
29: $\zeta = 1$
30: **end if**
31: $i = i + 1$
32: **until** The accuracy of the global model Acc_i reaches Acc^{target}
33: **end for**
34: **return** π_θ.

In addition, we adopt proximal policy optimization (PPO) algorithm to formulate the objective function of the actor network [27]

$$J^{\text{CLIP}}(\theta) = \mathbb{E}_i[\min(u_i(\theta)A_i^{\text{GAE}}, clip(u_i(\theta), 1-\rho, 1+\rho)A_i^{\text{GAE}})], \tag{22}$$

where $u_i(\theta) = \frac{\pi_\theta(\boldsymbol{a}_i|\boldsymbol{s}_i)}{\pi_{\theta_{\text{old}}}(\boldsymbol{a}_i|\boldsymbol{s}_i)}$, θ_{old} is the policy parameters before the update, $clip(\cdot, 1-\rho, 1+\rho)$ refers to a clipping function with a lower bound $(1-\rho)$ and an upper bound $(1+\rho)$. The critic network in PPO aims to minimize the loss function as follows:

$$L^{\text{PPO}}(\phi) = \mathbb{E}_i\left[\left(V_i^{\text{target}} - V(\boldsymbol{s}_i)\right)^2\right], \tag{23}$$

where V_i^{target} is the target value function obtained by GAE. PPO inherits the data efficiency and reliable performance from trust region policy optimization while is much simpler for implementation and tuning [28], which is the motivation for this paper to employ this algorithm. The actor generates normal distribution with mean $\boldsymbol{\mu}_i$ and the standard deviation $\boldsymbol{\sigma}_i$. The shape of $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i$ is $(batch\_size, N)$. The action $\boldsymbol{a}$ is obtained by sampling from the normal distribution and feeding into a softmax layer. The critic produces the state-value function $V(\boldsymbol{s})$.

Algorithm 1 presents the workflow of training the DRL agent in FL. In Lines 1-2, the server receives the local information, initializes θ and ϕ. In Lines 4-6, the index of the global iteration i is set as 1, and the global model $\boldsymbol{w}^1$ is initialized; $\boldsymbol{s}_1$ is obtained based on (15), and the vectors in $\boldsymbol{s}_0$ is sorted by the device indices; $count\_arr$ is created to record how many times each local devices has been scheduled. In Lines 8-12, a set of local devices $\mathcal{M}_i$ is selected based on the probability vector $\boldsymbol{a}_i$. In Lines 13-17, the n-th element ($n \in \mathcal{M}_i$) in $count\_arr$ is incremented by 1; the selected devices perform local update and send the local models back to the server. In Lines 18-20, the server performs global aggregation to obtain $\boldsymbol{w}^{i+1}$; the server constructs $\boldsymbol{s}_{i+1}$ based on $count\_arr$; for example, if $count\_arr = [3, 5, 1]$, $\boldsymbol{s}_{i+1}$ will be $\{\boldsymbol{\eta}_3, \boldsymbol{\eta}_1, \boldsymbol{\eta}_2\}$. In Lines 21-30, the DRL agent is trained for W epochs every O steps; at each training epoch, Y samples are randomly collected from the replay buffer Ω to update θ and ϕ.

The convergence of the proposed device scheduling method is warranted via the theoretical analysis in [29]. Note that [29] is based on several assumptions on the loss function of FL (e.g., smooth, strongly convex, etc). The empirical results in Section IV have demonstrated that the proposed device scheduling method enables the FL model to achieve convergence (i.e., target accuracy) even though the loss function of our paper does not meet those assumptions.

IV. Performance Evaluation

A. Experimental Setup

Consider an FL system with $N = 100$ local devices randomly distributed in a square of size 2 km×2 km, and the center of the area is a remote server. The path loss model is $128.1 + 37.6 \log_{10} d_n$(km), and the standard deviation of shadow fading is 8 dB. The proposed FL framework is trained on both IID and non-IID datasets. For the non-IID setup, 50% of the data in local dataset $\mathcal{D}_n$ come from the same label, while the rest of the data belong to other labels. The FL model is convolutional neural networks (CNNs) with the cross-entropy loss function. The details of these datasets as well as the network architectures of the FL model are listed as follows

MNIST: MNIST [30] is a dataset of small square 28×28-pixel gray-scale images of handwritten single digits between 0 and 9. The training set contains 60000 images while the testing set has 10000 testing images. The CNN model trained on MNIST consists of two 5×5 convolution layers. The output dimensions of the first convolution layer and the second convolution layer are 15 and 28, respectively. Each convolution layer is followed by a 2×2 max-pooling layer.

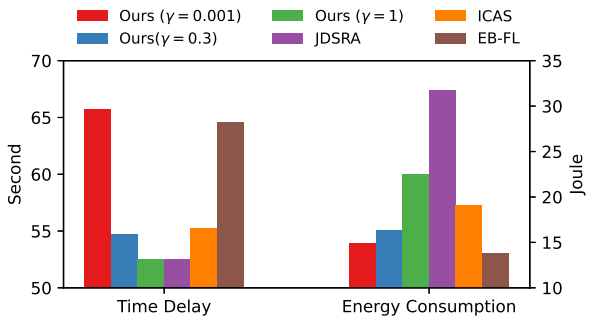


Fig. 2: Time delay and energy consumption at one global iteration under different bandwidth allocation methods.

The output of the second pooling layer is flattened and fed into two linear layers. The output dimension of the first linear layer and the second linear layer are 224 and 10, respectively. The target accuracy Acc^{target} is 99% for both IID and non-IID setups.

CIFAR-10: CIFAR-10 [31] contains 60000 32×32 color images involving ten object labels. Each label has 6000 images, where 5000 images belong to the training set while the rest of the data belong to the testing set. The CNN model trained on CIFAR-10 contains two 5×5 convolution layers. The output dimensions of the first convolution layer and second convolution layer are 15 and 28, respectively. Each convolution layer is followed by a 2×2 max-pooling layer. The second pooling layer is connected with two linear layers. The output dimension of the first linear layer and the second linear layer are 300 and 10, respectively. The target accuracy Acc^{target} is 56.5% for the IID setup and 54.5% for the non-IID setup.

In terms of the DRL agent, both the actor and critic contain an LSTM layer with 256 hidden units followed by two linear layers. The output dimensions of the actor and critic are provided in Section III-C.

B. Performance evaluation of bandwidth allocation

The proposed bandwidth allocation method is compared with three benchmarks: JDSRA [19], ICAS [18], and EB-FL. JDSRA optimizes the time delay of model training and transmission. ICAS minimizes the latency of model transmission and allocates bandwidth according to the channel states. EB-FL equally allocates the bandwidth resources to the scheduled devices. We adopt the experimental setup for CIFAR-10 (i.e., $D_n \in [300, 700]$, $z = 882$ KB, $K = 10$) while performing bandwidth allocation. We randomly schedule 10 devices to formulate the set $\mathcal{M}_i$.

Fig. 2 presents the one-round time delay T_i and energy consumption E_i of FL. It can be seen that our method obtains lower time delay and higher energy consumption as γ increases. JDSRA achieves the same time delay as ours with $\gamma = 1$. However, the energy consumption is not considered in JDSRA and thus is the highest among all the benchmarks. EB-FL attains the lowest energy consumption but high time delay. When $\gamma = 0.3$, the proposed method exhibits lower time delay and energy consumption compared with the ICAS approach. The experimental result shows that the proposed bandwidth allocation method outperforms other benchmarks in balancing the time delay and energy consumption of FL.

C. Training process of the DRL agent

The LSTM-based DRL agent is trained to perform device scheduling for FL based on Algorithm 1. We also train the MLP-based agent for performance comparison. Fig. 3 depicts the training process of the DRL agents on two datasets. We set γ as 1 during the training process. 10 devices are scheduled at each global iteration (i.e., $M = 10$). Due to the dynamic learning process of FL, the value of the objective function $E + \gamma T$ varies even though the scheduling strategy is unchanged. Therefore, $E+\gamma T$ in Fig. 3 is the average value of 30 episodes, which can be obtained by a sliding window.

According to Fig. 3, the LSTM-based agent enables $E+\gamma T$ to decrease as the training process goes on. In contrast, the MLP-based agent can hardly be trained to reduce objective function (12), which is in line with the discussion in Section III-B.

V. Conclusions

In this paper, we proposed a joint device scheduling and bandwidth allocation framework to improve the efficiency of FL. Specifically, we formulated an optimization problem to minimize the weighted sum of the time delay and energy consumption for training the entire FL algorithm. To solve this problem, we presented a DRL-based device scheduling method. The DRL environment, including the state space, action space, and reward function, was carefully designed. We adopted LSTM as the DRL agent, and the PPO algorithm was employed to update the DRL agent. Given the scheduled devices, we proposed a bandwidth allocation method to minimize the one-round system cost. The proposed FL framework was evaluated on MNIST and CIFAR-10. The experimental results show that the proposed bandwidth allocation method enables FL to balance the tradeoff between the time delay and energy consumption. Besides, the DRL-based device scheduling method helps FL to reach convergence with a lower system cost.

Acknowledgment

This research is supported by the National Research Foundation, Singapore and Infocomm Media Development Authority under its Future Communications Research & Development Programme.

References

[1] H. Yang, J. Zhao, K.-Y. Lam, Z. Xiong, Q. Wu, and L. Xiao, "Distributed deep reinforcement learning-based spectrum and power allocation for heterogeneous networks," *IEEE Transactions on Wireless Communications*, vol. 21, no. 9, pp. 6935–6948, 2022.

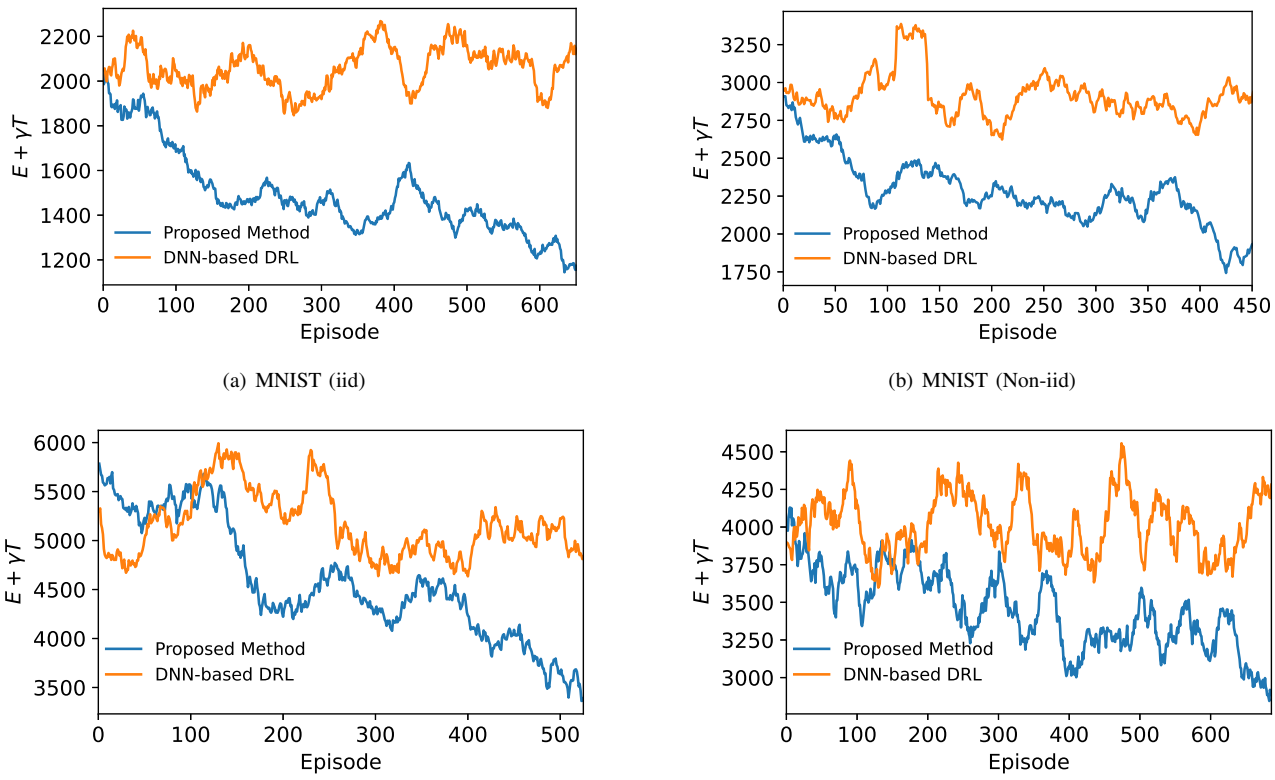
[2] F. Li, K.-Y. Lam, Z. Ni, D. Niyato, X. Liu, and L. Wang, "Cognitive carrier resource optimization for internet-of-vehicles in 5g-enhanced smart cities," *IEEE Network*, vol. 36, no. 1, pp. 174–180, 2022.

[3] H. Yang, J. Zhao, J. Nie, N. Kumar, K.-Y. Lam, and Z. Xiong, "Uav-assisted 5g/6g networks: Joint scheduling and resource allocation based on asynchronous reinforcement learning," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2021, pp. 1–6.

[4] H. Yang, J. Zhao, Z. Xiong, K.-Y. Lam, S. Sun, and L. Xiao, "Privacy-preserving federated learning for uav-enabled networks: Learning-based joint scheduling and resource management," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 10, pp. 3144–3159, 2021.

[5] A. H. Sodhro, M. S. Obaidat, Q. H. Abbasi, P. Pace, S. Pirbhulal, A.-U.-H. Yasar, G. Fortino, M. A. Imran, and M. Qaraqe, "Quality of service optimization in an iot-driven intelligent transportation system," *IEEE Wireless Communications*, vol. 26, no. 6, pp. 10–17, 2019.

[6] A. Chattopadhyay, K.-Y. Lam, and Y. Tavva, "Autonomous vehicle: Security by design," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 11, pp. 7015–7029, 2021.



(a) MNIST (iid)

(b) MNIST (Non-iid)

(c) CIFAR-10 (iid)

(d) CIFAR-10 (Non-iid)

Fig. 3: Objective value (12) during the training process of DRL-based device scheduling.

[7] 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.

[8] H. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017.

[9] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[10] P. K. *et al.*, "Advances and open problems in federated learning," *Foundations and Trends in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[11] T. Zhang, K.-Y. Lam, J. Zhao, and J. Feng, "Joint device scheduling and bandwidth allocation for federated learning over wireless networks," *IEEE Transactions on Wireless Communications*, 2023.

[12] S. Abdulrahman, H. Tout, A. Mourad, and C. Talhi, "Fedmccs: Multi-criteria client selection model for optimal iot federated learning," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4723–4735, 2021.

[13] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–7.

[14] J. Xu and H. Wang, "Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective," *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 1188–1200, 2021.

[15] 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)*, 2020, pp. 1–6.

[16] W. Gao, Z. Zhao, G. Min, Q. Ni, and Y. Jiang, "Resource allocation for latency-aware federated learning in industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 12, pp. 8505–8513, 2021.

[17] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Convergence time optimization for federated learning over wireless networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 4, pp. 2457–2471, 2021.

[18] J. Ren, Y. He, D. Wen, G. Yu, K. Huang, and D. Guo, "Scheduling for cellular federated edge learning with importance and channel awareness," *IEEE Transactions on Wireless Communications*, vol. 19, no. 11, pp. 7690–7703, 2020.

[19] W. Shi, S. Zhou, Z. Niu, M. Jiang, and L. Geng, "Joint device scheduling and resource allocation for latency constrained wireless federated learning," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 453–467, 2021.

[20] M. M. Amiri, D. Gündüz, S. R. Kulkarni, and H. Vincent Poor, "Update aware device scheduling for federated learning at the wireless edge," in *2020 IEEE International Symposium on Information Theory (ISIT)*, 2020, pp. 2598–2603.

[21] W. Shi, S. Zhou, and Z. Niu, "Device scheduling with fast convergence for wireless federated learning," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[22] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "Sample efficient actor-critic with experience replay," 2016.

[23] H. Zhu, J. Xu, S. Liu, and Y. Jin, "Federated learning on non-iid data: A survey," 2021.

[24] C. R. Harris, K. J. Millman, and *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020.

[25] A. Paszke, S. Gross, and e. a. Massa, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[26] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2015.

[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[28] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov, "Openai baselines," https://github.com/openai/baselines, 2017.

[29] Y. J. Cho, J. Wang, and G. Joshi, "Client selection in federated learning: Convergence analysis and power-of-choice selection strategies," 2020.

[30] Y. L. Cun, J. S. Denker, and S. A. Solla, *Optimal Brain Damage*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990, p. 598–605.

[31] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.