Multi-Job Intelligent Scheduling With Cross-Device Federated Learning

Ji Liu, Juncheng Jia, Beichen Ma, Chendi Zhou, Jingbo Zhou, Yang Zhou, Huaiyu Dai, Fellow, IEEE, and Dejing Dou, Senior Member, IEEE

Abstract—Recent years have witnessed a large amount of decentralized data in various (edge) devices of end-users, while the decentralized data aggregation remains complicated for machine learning jobs because of regulations and laws. As a practical approach to handling decentralized data, Federated Learning (FL) enables collaborative global machine learning model training without sharing sensitive raw data. The servers schedule devices to jobs within the training process of FL. In contrast, device scheduling with multiple jobs in FL remains a critical and open problem. In this article, we propose a novel multi-job FL framework, which enables the training process of multiple jobs in parallel. The multi-job FL framework is composed of a system model and a scheduling method. The system model enables a parallel training process of multiple jobs, with a cost model based on the data fairness and the training time of diverse devices during the parallel training process. We propose a novel intelligent scheduling approach based on multiple scheduling methods, including an original reinforcement learning-based scheduling method and an original Bayesian optimization-based scheduling method, which corresponds to a small cost while scheduling devices to multiple jobs. We conduct extensive experimentation with diverse jobs and datasets. The experimental results reveal that our proposed approaches significantly outperform baseline approaches in terms of training time (up to 12.73 times faster) and accuracy (up to 46.4% higher).

Index Terms—Federated learning, scheduling, multi-job, parallel execution, distributed learning

1 INTRODUCTION

In recent years, we have witnessed a large amount of decentralized data in various Internet of Things (IoT) devices, mobile devices, etc. [1], which can be exploited to train machine learning models of high accuracy for diverse artificial intelligence applications. Since the data contain sensitive information of end-users, a few stringent legal restrictions [2], [3], [4], [5] have been put in place to protect data security and privacy. In this case, it is difficult or even impossible to aggregate the decentralized data into a single server or a data center to train machine learning models. To enable collaborative training with decentralized data, Federated Learning (FL) [6], which does not transfer raw data, have emerged as a practical approach.

FL was first introduced to collaboratively train a global model with non-Independent and Identically Distributed (non-IID) data distributed across mobile devices [6]. During the training process of FL, the raw data remains decentralized without being transferred to a single server or a single data center [7], [8]. FL only allows the intermediate data to be transferred from the distributed devices, which can be the weights or the gradients of a model. FL generally utilizes a parameter server architecture [9], [10], [11], where a server (or a group of servers) coordinates the training process with numerous devices. To collaboratively train a global model, the server selects (schedules) several devices to perform local model updates based on their local data, and then it aggregates the local models to obtain a new global model. This process is repeated multiple times to generate a global model of high accuracy.

While current FL solutions [6], [12] focus on a single-task job or a multi-task job [13], FL with multiple jobs [14] remains an open problem. The major difference between the multi-task job and multiple jobs is that the tasks of the multi-task job share some common parts of the model, while the multiple jobs do not interact with each other in terms of the model. The multi-job FL deals with the simultaneous training process of multiple independent jobs. Each job corresponds to multiple updates during the training process of a global model with the corresponding decentralized data. While the FL with a single job generally selects a portion of devices to update the model, the other devices remain idle, and the efficiency thus is low. The multi-job FL can well exploit diverse devices for multiple jobs simultaneously, which brings high efficiency. The available devices are generally heterogeneous [15], [16], i.e., the computing and communication capacity of each device is different, and the data in each device may also differ. For instance, multiple machine learning jobs, e.g., CTR models [17], [18], mobile keyboard prediction [19], and travel time prediction [20], may be concurrently executed with FL. The concurrent execution can be carried out with the same group of users. In addition, industry-level machine learning jobs,
We propose MJ-FL, a multi-job FL framework composed of a parallel training process for multiple jobs and a cost model for scheduling methods. We propose combining the capability and data fairness in the cost model to improve the efficiency of the training process and the accuracy of the global model.

We propose two scheduling methods, i.e., Reinforcement Learning (RL)-based and Bayesian Optimization (BO)-based methods, to schedule the devices for diverse jobs (more details including the method to estimate the loss in Section 3.3 are added compared with [31]). Each method has advantages in a specific situation. The BO-based method performs better for simple jobs, while the RL-based method is more suitable for complex jobs. In addition, we provide theoretical convergence analysis in Section 5.

- We propose a novel intelligent scheduling approach based on multiple scheduling methods (extra contribution compared with [31]). The novel intelligent scheduling approach is a meta-scheduling approach that coordinates multiple scheduling methods to achieve excellent performance with an adapted dynamic cost model.

- We carry out extensive experimentation to validate the proposed approach. We exploit multiple jobs, composed of ResNet18, CNN, AlexNet, VGG, and LeNet, to demonstrate the advantages of our proposed approach using both IID and non-IID datasets (with extra extensive experimental results for the meta-scheduling approach).

The rest of the paper is organized as follows. We present the related work in Section 2. Then, we explain the system model and formulate the problem with a cost model in Section 3. We present the scheduling methods in Section 4. We provide theoretical convergence analysis in Section 5. The experimental results with diverse models and datasets are given in Section 6. Finally, Section 7 concludes the paper.

2 RELATED WORK

In order to protect the security and privacy of decentralized raw data, FL emerges as a promising approach, which enables training a global model with decentralized data [1], [6], [8], [15]. Based on the data distribution, FL can be classified into three types, i.e., horizontal, vertical, and hybrid [1], [8]. The horizontal FL addresses the decentralized data of the same features, while the identifications are different. The vertical FL handles the decentralized data of the same identifications with different features. The hybrid FL deals with the data of different identifications and different features. In addition, FL includes two variants: cross-device FL and cross-silo FL [7]. The cross-device FL trains global machine learning models with a huge number of mobile or IoT devices, while the cross-silo FL handles the collaborative training process with the decentralized data from multiple organizations or geodistributed datacenters. In this paper, we focus on the horizontal and cross-device FL.

Current FL approaches [32], [33], [34], [35], [36], [37] generally deal with a single job, i.e., with a single global model. While some FL approaches have been proposed to handle multiple tasks [13], [38], the tasks share some common parts of a global model and deal with the same types of data. In addition, the devices are randomly selected (scheduled) in these approaches.

A few scheduling approaches [16], [16], [27], [28], [28], [29], [29], [30], [39] exist for single-job scheduling while the device scheduling with multi-job FL is rarely addressed. The scheduling methods in the above works are mainly based on some heuristics. For instance, the greedy method [40] and the random scheduling method [27] are proposed for FL, while genetic algorithms [30] are exploited for distributed systems. However, these methods do not consider the fairness of data, which may lead to low accuracy for multi-job FL. The black-box optimization-based methods, e.g., RL [39], BO [41], and deep neural network [42], have been proposed to improve the efficiency, i.e., the reduction of execution time, in distributed systems. They do not consider data fairness either, which may lead to low accuracy for multi-job FL. Although ensemble learning or ensemble method consisting of multiple models [43], has been
TABLE 1
Summary of Main Notations

| Notation | Definition |
|----------|------------|
| $K$; $|K|$ | Set of all devices; size of $K$ |
| $M$; $m$; $T$ | The total number of jobs; index of jobs; total training time |
| $D^m_k$, $D^m_k$, $s_k^m$ | Local dataset of Job $m$ on Device $k$; size of $D^m_k$; batch size of the local update of Device $k$ |
| $D^m$; $D^m_k$ | Global dataset of Job $m$; size of $D^m$ |
| $F^m_k(w)$; $F^m(w)$ | Local loss function of Job $m$ in Device $k$; global loss function of Job $m$ |
| $w^m_{i,j}(k)$ | Local model of Job $m$ in Device $k$ in the $j$-th local update of Round $r$ |
| $R_m$ | The maximum rounds for Job $m$ during the execution |
| $R'_m$ | The minimum rounds for Job $m$ to achieve the required performance (loss value or accuracy) |
| $l_m$ | The desired loss value for Job $m$ |
| $t_m$, $C_m$ | Number of local epochs of Job $m$; the ratio between the number of devices scheduled to Job $m$ and $|K|$ |
| $S_m$, $s^m_k$ | The frequency vector for Job $m$; the frequency of Device $k$ scheduled to Job $m$ at Round $r$ |
| $V^m_r$ | A set of devices scheduled to Job $m$ at Round $r$ |
| $V_o$, $V_o^r$ | A set of occupied devices; the set of occupied devices in Round $r$ |
| $s^m_{r,h}$ | The sampled dataset for Job $m$ at local iteration $h$ Round $r$ on Device $k$ |

exploited for scheduling parallel tasks [44], proper cost models and ensemble mechanism should be well designed.

Different from all existing works, we propose a system model for the multi-job FL with the consideration of both efficiency and accuracy. In addition, we propose a novel intelligent scheduling approach based on multiple scheduling methods. To improve the efficiency of the training process, we propose two original scheduling methods, i.e., RL and BO, for multi-job FL, which are suitable for diverse models and for both IID and non-IID datasets.

3 SYSTEM MODEL AND PROBLEM FORMULATION

In this section, we first explain the motivation for multi-job FL. Then, we propose our multi-job FL framework, consisting of a multi-job FL process and a cost model. Afterward, we formally define the problem to address in this paper. Please see the meanings of the major notations in Table 1.

3.1 Motivation for Multi-Job Federated Learning

Let us assume a scenario where multiple FL jobs are processed simultaneously, e.g., image classification, speech recognition, and text generation. These jobs can be trained in parallel to exploit the available devices efficiently. However, while each device can only update the model of one job at a given time slot, it is critical to schedule devices to different jobs during the training process. As the devices are generally heterogeneous, some may possess high computation or communication capability while others may not. In addition, the data fairness of multiple devices may also impact the convergence speed of the training process. For instance, if only specific powerful devices are scheduled for a job, the model can only learn from the data stored on these devices, while the knowledge from the data stored on other devices may be missed. In order to accelerate the training process of multiple jobs with high accuracy, it is critical to consider how to schedule devices while considering both the computing and communication capability and the data fairness.

A straightforward approach is to train each job separately using the mechanism explained in [27], while exploiting the existing scheduling of single-job FL, e.g., FedAvg [27]. In this way, simple parallelism is considered while the devices are not fully utilized and the system is of low efficiency. In addition, a direct adaptation of existing scheduling methods to multi-job FL cannot address the efficiency and the accuracy at the same time. Thus, it is critical to propose a reasonable and effective approach for the multi-job FL.

3.2 Multi-Job Federated Learning Framework

In this paper, we focus on an FL environment composed of a server module and multiple devices. The server module (Server) may consist of a single parameter server or a group of parameter servers [45]. In this section, we present a multi-job FL framework, which is composed of a process for the multi-job execution and a cost model to estimate the cost of the execution.

3.2.1 Multi-Job FL Process

Within the multi-job FL process, we assume that $K$ devices, denoted by the set $K$, collaboratively train machine learning models for $M$ jobs, denoted by the set $M$. Each Device $k$ is assumed to have $M$ local datasets corresponding to the $M$ jobs without loss of generality, and the dataset of the $m$-th job on Device $k$ is expressed as $D^m_k = \{x^m_k \in R^n_m, y^m_k \in R^n_{y_k} | x^m_k \in D^m_k\}$ with $D^m_k = \{D^m_k\}$ as the number of data samples, $x^m_k$ representing the $d$-th $n_k$-dimensional input data vector of Job $m$ at Device $k$, and $y^m_k$ denoting the labeled output of $x^m_k$. The whole dataset of Job $m$ is denoted by $D^m = \bigcup_{k \in K} D^m_k$ with $D^m = \sum_{k \in K} D^m_k$. The objective of multi-job FL is to learn respective model parameters $\{w^m\}$ based on the decentralized datasets.

The global learning problem of multi-job FL can be expressed by the following formulation:

$$\min_w \frac{1}{M} \sum_{m=1}^{M} L_m, \text{ with } L_m = \frac{1}{K} \sum_{k=1}^{K} D^m_k F^m_k(w^m),$$

(1)

where $L_m$ is the loss value of Job $m$, $F^m_k(w^m)$ is the loss value of Job $m$ at Device $k$, $W := \{w^1, w^2, \ldots, w^K\}$ is the set of weight vectors for all jobs, and $F^m_k(w^m, x^m_{k,d}, y^m_{k,d})$ captures the error of the model parameter $w^m$ on the data pair $\{x^m_{k,d}, y^m_{k,d}\}$.

In order to solve the problem defined in Formula (1), the Server needs to continuously schedule devices for different jobs to iteratively update the global models until the training processes of the corresponding job converge or achieve the target performance requirement (in terms of accuracy or loss value). We design a multi-job FL process as shown in Fig. 1. The Server first initializes a global model for each job. The initialization can be implemented randomly or from the pretraining process with public data. To know the current status of devices, the Server sends requests to available devices in Step 1. Then, in Step 2, the Server schedules devices for the current job based on the scheduling plan generated by the
scheduling method (see details in Section 4). The scheduling plan is a set of devices selected to execute the local training process of the current job. Note that the scheduling process generates a scheduling plan for each job during the training process of multiple jobs, i.e., an online strategy, while the scheduling processes for multiple jobs are carried out in parallel. The Server distributes the latest global model for the current job to the scheduled devices in Step 3, and then the model is updated in each device based on the local data in Step 4. Afterward, each device sends the updated model to the Server after its local training in Step 5. Finally, the Server aggregates the models of scheduled devices to generate a new global model in Step 6. The combination of Steps 1 - 6 is denoted by a round, which is repeated for each job until the corresponding global model reaches the expected performance (accuracy, loss value, or convergence). Please note that multiple jobs are executed in parallel asynchronously, while a device can only be scheduled to one job at a given time. In addition, we assume that each job is of equal importance.

As an FL environment may contain GPUs or other high-performance chips, it is beneficial to train multiple jobs simultaneously to reduce training time while achieving target accuracy. Within each round, Step 6 exploits FedAvg [27] to aggregate multiple models within each job, which can ensure the optimal convergence [46], [47]. Within our framework, the sensitive raw data is kept within each device, while only the models are allowed to be transferred. Other methods, e.g., homomorphic encryption [48] and differential privacy [49], can be exploited to protect the privacy of sensitive data.

### 3.2.2 Cost Model

In order to measure the performance of each round, we exploit a cost model defined in Formula (2), which is composed of time cost and data fairness cost. The data fairness has a significant impact on convergence speed.

$$\text{Cost}_{m}^{r}(V_{m}^{r}) = \alpha \cdot T_{m}^{r}(V_{m}^{r}) + \beta \cdot F_{m}^{r}(V_{m}^{r}),$$

where $\alpha$ and $\beta$ are the weights of time cost and fairness cost respectively, $T_{m}^{r}()$ represents the execution time of the training process in Round $r$ with the set of scheduled devices $V_{m}^{r}$, and $F_{m}^{r}()$ is the corresponding data fairness cost. We choose the linear combination because of its convenience and excellent performance. In practice, we empirically set $\alpha$ and $\beta$ based on the information from previous execution and adjust them using small epochs. We increase $\alpha$ for fast convergence and increase $\beta$ mainly for high accuracy.

As defined in Formula (3), the execution time of a round depends on the slowest device in the set of scheduled devices.

$$T_{m}^{r}(V_{m}^{r}) = \max\{t_{m}^{k}\},$$

where $t_{m}^{k}$ is the execution time of Round $r$ in Device $k$ for Job $m$. $t_{m}^{k}$ is composed of the communication time and the computation time, which is complicated to estimate and differs for different devices. In this study, we assume that the execution time of each device follows the shift exponential distribution as defined in Formula (4) [50], [51]:

$$P(t_{m}^{k} < t) = \begin{cases} 1 - e^{-\frac{t}{\mu_{m}^{k}}}, & t \geq \tau_{m}^{k} a_{k} D_{m}^{k}, \\ 0, & \text{otherwise}, \end{cases}$$

where the parameters $\alpha_{k} > 0$ and $\mu_{k} > 0$ are the maximum and fluctuation of the computation and communication capability, which is combined into one quantity, of Device $k$, respectively. Moreover, we assume that the calculation time of model aggregation has little impact on the training process because of the strong computation capability of the Server and the low complexity of the model.

The data fairness of Round $r$ corresponding to Job $m$ is indicated by the deviation of the frequency of each device to be scheduled to Job $m$ defined in Formula (5).

$$F_{m}^{r}(V_{m}^{r}) = \frac{1}{|K|} \sum_{k \in K} \left( s_{k,m}^{r} - \frac{1}{|K|} \sum_{k \in K} s_{k,m}^{r} \right)^{2},$$

where $s_{k,m}^{r}$ is the frequency of Device $k$ to be scheduled to Job $m$, and $K$ and $|K|$ are the set of all devices and the size, respectively. $s_{k,m}^{r}$ is calculated by counting the total number of devices scheduled to Job $m$.
of the appearance of Device $k$ to be scheduled to Job $m$ in the set of scheduling plans for Job $m$, i.e., $\{V_m^1, \ldots, V_m^r\}$. In particular, $S_m^r = \{s_{k,m}^1, \ldots, s_{k,m}^r\}$ represents the frequency vector of Job $m$ at Round $r$. At the beginning, i.e., Round 0, each $s_{k,m}^0 \in S_m^0$ is 0. $s_{k,m}^r$ represents the frequency of Device $k$ scheduled to Job $m$ at Round $r$. Then, we can calculate $s_{k,m}^{r+1}$ using the following formula:

$$s_{k,m}^{r+1} = \begin{cases} s_{k,m}^r + 1, & \text{if Device } k \in V_m^r \\ s_{k,m}^r, & \text{otherwise} \end{cases}$$

(6)

Please note that the “data fairness” is reflected by the deviation of the frequency of each device scheduled to a job [52], which is different from the “fairness” (the bias of the machine learning models concerning certain features) in machine learning [53]. Formula (5) is inspired by [52], and we are among the first to extend this idea from distributed or network systems to FL. When the devices are non-uniformly sampled with low data fairness, the convergence is slowed down [46], [47]. In addition, data fairness is important due to the underlying data heterogeneity across the devices. Data fairness can help arbitrarily select devices without harming the learning performance.

3.3 Problem Formulation

The problem we address is how to reduce the training time when given a loss value for each job. While the execution of each job is carried out in parallel, the problem can be formulated as follows:

$$\min_{V_m} \left\{ \sum_{m=1}^M \sum_{r=1}^{R_m} \mathcal{F}_m(V_m) \right\}$$

subject to

$$\left\{ \begin{array}{l} \mathcal{L}_m(R_m) \leq l_m, \\ V_m \in \mathcal{K}, \forall m \in \{1, 2, \ldots, M\}, \forall r \in \{1, 2, \ldots, R_m\} \end{array} \right. \right.$$

where $l_m$ is the given loss value of Job $m$, $R_m$ represents the minimum number of rounds to achieve the given loss in the real execution, and $\mathcal{L}_m(R_m)$ is the loss value of the trained model at Round $R_m$ defined in Formula (1).

We assume Stochastic Gradient Descent (SGD) is utilized to train models, which converges at a rate of $O(r)$ with $r$ representing the number of rounds [54]. Inspired by [46], we exploit Formula (8) to roughly estimate the loss value of the global model for Job $m$ at Round $r$.

$$\text{Loss}_m(r) = \frac{1}{\gamma_m^0 + \gamma_m^1 + \gamma_m^2},$$

(8)

where $\gamma_m^0$, $\gamma_m^1$, and $\gamma_m^2$ represent non-negative coefficients of the convergence curve of Job $m$. $\gamma_m^0$, $\gamma_m^1$, and $\gamma_m^2$ can be calculated based on previous execution. In addition, we assume that the real number of rounds corresponding to the same loss value has 30% error compared with $r$ (from the observation of multiple executions). Given a loss value of a model, we exploit this loss estimation method to calculate the maximum rounds for each job. Given a loss value of a model, we utilize this loss estimation method to calculate the number of rounds as $R_m$ and take $(1 + 0.3) \times R_m$ as $R_m$ defined in Table 1. Please note that this estimation is different from the loss value during the real execution; i.e., $R_m$ can be different from $R_m$.

As it requires the global information of the whole training process, which is hard to predict, to solve the problem, we transform the problem to the following one, which can be solved with limited local information of each round. In addition, in order to achieve the given loss value of Job $m$ within a short time (the first constraint in Formula (7)), we need to consider the data fairness within the total cost in Formula (9), within which the data fairness can help reduce $R_m$ to minimize the total training time.

$$\min_{V_m} \left\{ \text{TotalCost}(V_m) \right\}$$

(9)

subject to

$$\text{TotalCost}(V_m) = \sum_{m=1}^M \text{Cost}_m(V_m),$$

where $\text{Cost}_m(V_m)$ can be calculated based on Formula (2) with a set of scheduled devices $V_m$ to be generated using a scheduling method for Job $m$. Since the scheduling results of one job may potentially influence the scheduling of other jobs, we consider the cost of other jobs when scheduling devices to the current job in this problem. As the search space is $O(2^K)$, this scheduling problem is a combinatorial optimization problem [55] and NP-hard [25].

4 DEVICE SCHEDULING FOR MULTI-JOB FL

In this section, we propose two original scheduling methods, i.e., BO-based and RL-based, and a novel intelligent scheduling approach, i.e., meta-greedy, to address the problem defined in Formula (9). The scheduling plan generated by a scheduling method is defined in Formula (10):

$$V_m^r = \arg\min_{V_m \in \mathcal{K}(V_m)} \text{TotalCost}(V_m),$$

(10)

where $V_m^r$ is a scheduling plan, $\mathcal{K}(V_m)$ represents the set of available devices to schedule, $\text{TotalCost}(V_m)$ is defined in Formula (9), and $\mathcal{K}$ and $\mathcal{V}$ are the set of all devices and the set of occupied devices in Round $r$, respectively.

The BO-based and RL-based methods are designed for different model complexities, and we choose the better one based on known profiling information with small tests (a few epochs) to avoid possible limitations. RLDS favors complex jobs, as it can learn the influence among diverse devices. The influence refers to the concurrent, complementary, and latent impacts of the data in multiple devices for diverse jobs. However, BODS favors simple jobs, while it relies on simple statistical knowledge. The complexity of jobs is determined by the number of parameters of models and the size of the training dataset. We consider the probability to release the devices in $V_m$ in BODS and RLDS, and possible concurrent occupation of other devices for other jobs, which is not explained in the paper to simplify the explanation. During the execution, we assume that a fraction of the devices is sampled for each job and some devices can be unavailable for the execution. In order to optimize the scheduling process for diverse types of models, we further propose a meta-greedy scheduling approach, which takes advantage of multiple scheduling
methods and chooses the most appropriate scheduling plan from the results of the scheduling methods.

**Algorithm 1. Bayesian Optimization-Based Scheduling**

**Input:**
- \( \mathcal{V}_o \): A set of occupied devices
- \( S_m \): A vector of the frequency of each device scheduled to Job \( m \)
- \( R_m \): The maximum round of the current Job \( m \)
- \( l_m \): The desired loss value for Job \( m \).

**Output:**
- \( V_m = \{V_{m1}^r, \ldots, V_{mL}^r\} \): a set of scheduling plans, each with the size \( |K| \times C_m \)
1. \( \Pi_L \leftarrow \) Randomly generate a set of observation points and calculate the cost
2. for \( r \in \{1, \ldots, R_m\} \) and \( l_m \) is not achieved do
3. \( \Pi' \leftarrow \) Randomly generate a set of observation points with the devices within \( K \setminus \mathcal{V}_o \)
4. \( V_m^r \leftarrow \arg \max \ E_{V} \{V; \Pi'\} \)
5. FL training of Job \( m \) with \( V_m^r \) and update \( S_m, \mathcal{V}_o \)
6. \( C_r = \text{TotalCost}(V_m^r) \)
7. \( \Pi_{L+1} \leftarrow \Pi_{L+1} \cup \{V_m^r, C_r\} \)
8. end for

**4.1 Bayesian Optimization-Based Scheduling**

As the Gaussian Process (GP) [56] can well represent linear and non-linear functions, BO-based methods [57] can exploit a GP to find a near-optimal solution for the problem defined in Formula (10). In this section, we propose a Bayesian Optimization-based Device Scheduling method (BODS).

We adjust a GP to fit the cost function \( \text{TotalCost}(\cdot) \). The GP is composed of a mean function \( \mu \) defined in Formula (11) and a covariance function \( K \) defined in Formula (12) with a Matern kernel [58].

\[
\mu(V_m^r) = \frac{E}{V_m^r \in K \setminus \mathcal{V}_o} \{\text{TotalCost}(V_m^r)\}
\]

(11)

\[
K(V_m^r; V_m^{r'}) = \frac{E}{V_m^r \in K \setminus \mathcal{V}_o, V_m^{r'} \in K \setminus \mathcal{V}_o} \left\{([\text{TotalCost}(V_m^r) - \mu(V_m^r)]([\text{TotalCost}(V_m^{r'}) - \mu(V_m^{r'}))]\right\}
\]

(12)

The BODS is explained in **Algorithm 1**. First, we generate a random set of observation points and calculate the cost according to the Formula (2) (Line 1). Each observation point is a pair of scheduling plan and cost for the estimation of mean function and the covariance function. Then, in each round, we randomly sample a set of scheduling plans (Line 3), within which we use updated \( \mu \) and \( K \) based on \( \Pi_{L+1} \) (Line 4) to select the one with the largest reward. Afterward, we perform the FL training for Job \( m \) with the generated scheduling plan (Line 5). In the meanwhile, we calculate the cost corresponding to the actual execution (Line 6) according to Formula (9) and update the observation point set (Line 7).

Let \( (V_r, C) \) denote an observation point for Job \( m \) in Round \( r \), where \( V_r = \{V_{r1}, \ldots, V_{rL}\} \) and \( C \) is the cost value of \( \text{TotalCost}(V_r) \) while the scheduling plans of other jobs are updated with the ones in use in Round \( r \). At a given time, we have a set of observations \( \Pi_{L-1} = \{(V_{r1}, C_1), \ldots, (V_{rL-1}, C_{L-1})\} \) composed of \( L-1 \) observation points. We denote the minimum cost value within the \( L-1 \) observations by \( C_{L-1}^+ \). Then, we exploit Expected Improvement (EI) [59] to select a new scheduling plan \( V_m^r \) in Round \( r \) what improves \( C_{L-1}^- \) the most, which is the utility function. Please note that this is not an exhaustive search since we randomly select several observation points (a subset of the whole search space) at the beginning and add new observation points using the EI method. The utility function is defined in Formula (13).

\[
\alpha_{EI}(\mathcal{V}; \Pi_{L-1}) = E[u(V) | \mathcal{V}; \Pi_{L-1}]
\]

\[
= (C_{L-1}^- - \mu(V)) \Phi(C_{L-1}^+ ; \mu(V), K(V, V))
+ K(V, V) \mathcal{N}(C_{L-1}^+ ; \mu(V), K(V, V)),
\]

(14)

where \( K \) is the Cumulative Distribution Function (CDF) of the standard Gaussian distribution. Finally, we can choose the scheduling plan with the largest reward as the next observation point, i.e., \( V_{Lm}^r \).

**4.2 Reinforcement Learning-Based Scheduling**

In order to learn more information about the near-optimal scheduling patterns for complex jobs, we further propose a Reinforcement Learning-based Device Scheduling (RLDS) method as shown in Fig. 2. In addition, the method is inspired by [39], [60]. The scheduler of RLDS consists of a policy network and a policy converter. During the process of device scheduling, RLDS collects the status information of jobs as the input of the policy network. Afterwards, the policy network generates a list of probabilities on all devices as the output. Finally, the policy converter converts the list into a scheduling plan.

**4.2.1 Policy Network**

The policy network is implemented using a Long Short-Term Memory (LSTM) network followed by a fully connected layer, which can learn the device sharing relationship among diverse jobs. We take the computation and communication capability of available devices to be used in Formula (4), and the data fairness of each job defined in Formula (5) as the input. The network calculates the probability of each available device to be scheduled for a job.
4.2.2 Policy Converter
The Policy Converter generates a scheduling plan based on the probability of each available device which is calculated by the policy network with the $\epsilon$-greedy strategy [61].

4.2.3 Training
In the training process of RLDS, we define the reward as $R^m = -1 \times TotalCost(V_n)$. Inspired by [62], [63], we exploit Formula (15) to update the policy network:

$$
\theta' = \theta + \frac{\eta}{N} \sum_{n=1}^{N} \sum_{k \in V_{n,m}} \nabla \log P(\mathcal{A}_{k}^{m} | \mathcal{A}^{m}_{(k-1):1}; \theta) (\mathcal{A}^{m}_{n} - b_{m}),
$$

where $\theta'$ and $\theta$ represent the updated parameters and the current parameters of the policy network, respectively, $\eta$ represents the learning rate, $N$ is the number of scheduling plans to update the model in Round $r$ ($N > 1$ in the pre-training process and $N = 1$ during the execution of multiple jobs), $P$ represents the probability calculated based on the RL model, $\mathcal{A}_{k}^{m} = 1$ represents that Device $k$ is scheduled to Job $m$, and $b_{m}$ is the baseline value for reducing the variance of the gradient.

Algorithm 2. Reinforcement Learning Based Pre-Training

**Input:**
- $V_n$: A set of occupied devices
- $S_m$: A vector of the frequency of each device scheduled to Job $m$
- $N$: The number of scheduling plans used to train the network for each round
- $R_m$: The maximum round of the current Job $m$
- $l_m$: The desired loss value for Job $m$

**Output:**
- $\theta$: Parameters of the pre-trained policy network
- $\mathcal{A}_{n,m}^{m}$: a scheduling plan

1. $\theta \leftarrow$ pre-trained policy network
2. $\Delta \theta \leftarrow 0, b_{m} \leftarrow 0$
3. for $r \in \{1,2, \ldots, R_m\}$ and $l_m$ is not achieved do
4. $V_n \leftarrow$ generate a set of $N$ scheduling plans
5. $\mathcal{A}_{n,m}^{m} \leftarrow -1 \times TotalCost(V_{n,m})$
6. end for
7. Update $\theta$ according to Formula (15)
8. $b_{m} \leftarrow (1 - \gamma) \times b_{m} + \gamma \times \mathcal{A}_{n,m}^{m}$
9. $V_{n,m}^{r} \leftarrow \arg\min_{V_{n,m} \in V_n} TotalCost(V_{n,m})$
10. Update $S_m, V_n$ with $V_{n,m}^{r}$
11. end for

We pre-train the policy network using Algorithm 2. First, we randomly initialize the policy network (Line 1). We use the latest policy network and the $\epsilon$-Greedy method to generate $N$ scheduling plans (Line 5). The parameters are updated based on the Formula (15) (Line 7), and the baseline value $b_{m}$ is also updated with the consideration of the historical value (Line 8). Afterward, we choose the best scheduling plan that corresponds to the minimum total cost, i.e., the maximum reward (Line 9). Finally, we update the frequency matrix $S_m$ and the set of occupied devices $V_n$ while assuming that the best scheduling plan is used for the multi-job FL (Line 10).

Algorithm 3. Reinforcement Learning-Based Scheduling

**Input:**
- $V_n$: A set of occupied devices
- $S_m$: A vector of the frequency of each device scheduled to Job $m$
- $R_m$: The maximum round of the current Job $m$
- $l_m$: The desired loss value for Job $m$

**Output:**
- $V_m = \{V_{m}^{1}, \ldots, V_{m}^{R_m}\}$: a set of scheduling plans, each with the size $|K| \times C_m$
1. $\theta \leftarrow$ pre-trained policy network, $\Delta \theta \leftarrow 0, b_{m} \leftarrow 0$
2. for $r \in \{1,2, \ldots, R_m\}$ and $l_m$ is not achieved do
3. $V_{n}^{r} \leftarrow$ generate a scheduling plan using the policy network
4. FL training of Job $m$ and update $S_m, V_n$
5. Compute $\mathcal{A}_{n,m}^{m}$
6. Update $\theta$ according to Formula (15)
7. $b_{m} \leftarrow (1 - \gamma) \times b_{m} + \gamma \times \mathcal{A}_{n,m}^{m}$
8. end for

After the pre-training, we exploit RLDS during the training process of multiple jobs within the MJ-FL framework as shown in Algorithm 3. First, we load the pre-trained policy network and initialize the parameters $\Delta \theta, b_{m}$ (Line 1). When generating a scheduling plan for Job $m$, the latest policy network is utilized (Line 3). We perform the FL training for Job $m$ with the generated scheduling plan and update the frequency matrix $S_m$ and the set of occupied devices $V_n$ (Line 4). Afterward, we calculate the reward corresponding to the real execution (Line 5). The parameters are updated based on the Formula (15) (Line 6), while the baseline value $b_{m}$ is updated with the consideration of the historical value (Line 7).

4.3 Meta-Greedy Scheduling
Multiple jobs of diverse structures and layers exist within the multi-job federated learning environment. Some scheduling methods, e.g., BODS, favor simple jobs, while some other scheduling methods, e.g., RLDS, prefer complex jobs. In addition, heuristic scheduling methods, e.g., Greedy and Genetic, can be exploited to schedule devices, as well. In this case, we take advantage of the existing scheduling methods, and further propose a meta-greedy scheduling approach as shown in Fig. 3. In Round $r$, the Meta-Greedy executes six scheduling methods in parallel, appends the solution generated by each method to the candidate scheduling solutions set $\Theta_{m}^{r}$, and selects a scheduling plan from the set of candidate device scheduling solutions according to Formula (16) as the solution for the $r$-th round under the current job. The meta-greedy scheduling approach chooses the most appropriate one from the scheduling plans generated by multiple methods, which can be closer to the optimal solution compared with that of a single method.

The Meta-Greedy algorithm is shown in Algorithm 4. First, for each round (Line 2), we exploit diverse scheduling methods, e.g., BODS, RLDS, Random [27], FedCS [28], Genetic [30], and Greedy [40], to generate scheduling plan candidates (Line 3). As each scheduling method may have superior performance in a specific environment (Random corresponds to high final accuracy; Genetic corresponds to high accuracy at the beginning of the training process for the jobs of moderate complexity; Greedy corresponds to
high accuracy at the beginning of the training process of simple jobs; FedCS corresponds to high accuracy at the middle of the training process; see details in Section 6), we take these 6 methods in Meta-Greedy. Then, we choose a scheduling plan that corresponds to the smallest total cost according to Formula (16) (Line 4). Finally, we utilize the selected scheduling plan for the training process (Line 5).

We propose a dynamic cost model for the selection in Line 4. While Formula (2) can also be exploited for the selection, the influence of the data fairness becomes smaller in the later round of FL as the frequency of the participation has little change with a small increase of one. We reconstruct the cost model using Formula (16) to replace Formula (2).

$$ReCost(V^r) = \alpha \cdot \mathcal{F}^r_m(V^r_m) + \beta^r \cdot \mathcal{F}^r_m(V^r_m),$$  

(16)

where $\beta^r$ dynamically changes according to $r$ and is defined in Formula (17).

$$\beta^r = \beta \cdot \Omega(r), \ r \geq 1$$  

(17)

where $\Omega(r)$ is a function based on Round $r$. $\Omega(r)$ should enhance the impact of data fairness when $r$ becomes significant. We take $\Omega(r) = \sqrt{r}$ in our algorithm because of its excellent performance (see details in Section 6).

5 CONVERGENCE ANALYSIS

In this section, we present the theoretical convergence proof for our multi-job FL with arbitrary scheduling methods. We first introduce the assumptions and then present the convergence theorem and corollary.

Assumption 1. Lipschitz gradient: The function $F_k^m$ is $L$-smooth for each device $k \in \mathcal{N}$, i.e.,

$$\| \nabla F_k^m(x) - \nabla F_k^m(y) \| \leq L \| x - y \| .$$

Assumption 2. Unbiased stochastic gradient:

$$E_{\xi_k,h} \sim P_i [ \nabla F_k^m(w^m, \xi_{k,h}) ] = \nabla F_k^m(w^m).$$

Assumption 3. Bounded local variance: For each device $k \in \mathcal{N}$, the variance of its stochastic gradient is bounded:

$$E_{\xi_k,h} \sim P_i \| \nabla F_k^m(w^m, \xi_{k,h}) - \nabla F_k^m(w^m) \|^2 \leq \sigma^2.$$

Assumption 4. Bounded local gradient: For each device $k \in \mathcal{N}$, the expected squared of stochastic gradient is bounded:

$$E_{\xi_k,h} \sim P_i \| \nabla F_k^m(w^m, \xi_{k,h}) \|^2 \leq G^2.$$  

Algorithm 4. Meta-Greedy Scheduling

Input:

$\mathcal{V}_o$ : A set of occupied devices
$S_m$ : A vector of the frequency of each device scheduled to Job $m$
$R_m$ : The maximum round of the current Job $m$
$l_m$ : The desired loss value for Job $m$.

Output:

$\mathcal{V}_m = \{ V^r_m, \ldots, V_{R_m}^m \}$ : a set of scheduling plans, each with the size $|\mathcal{K}| \times C_m$

1: Initialize $\mathcal{V}_m \leftarrow \emptyset$
2: for $r \in \{1, 2, \ldots, R_m\}$ and $l_m$ is not achieved do
3: \hspace{1cm} $\Theta_m \leftarrow \Theta_m \cup \mathcal{V}_m^r$
4: \hspace{1cm} $\mathcal{V}_m^* \leftarrow \arg\min_{\mathcal{V} \in \mathcal{V}_m} \text{ReCost}(\mathcal{V})$
5: \hspace{1cm} $\mathcal{V}_m \leftarrow \mathcal{V}_m \cup \mathcal{V}_m^*$
6: end for

Assumption 1 has been made in [46], [64], while Assumptions 2, 3, 4, have been exploited in [65]. When Assumptions 1, 2, 3, and 4 hold, we get the following theorem.

**Theorem 5.1.** Suppose that Assumptions 1 to 4 hold, and consider that $F_k^m$ is a non-convex function. When the learning rate satisfies $0 < \eta^m \leq \frac{1}{R}$, then for all $R \geq 1$ we have:

$$\frac{1}{R} \sum_{r=1}^R \sum_{h=1}^H \mathbb{E} \| \nabla F^m_m(\bar{w}_{r,h}^m) \|^2$$

$$\leq \frac{2}{\eta^m_R RH} (F^m_m(\bar{w}_{1,1}^m) - F^{m^*}) + L^2 \sigma^2 + L^2 Q^2 (H - 1)^2 \eta^2 m^2 G^2$$  

(18)

where $F^{m^*}$ is the local optimal value, $R$ refers to the number of rounds for Job $m$ during the execution, $H$ represents the number of local iterations, and $Q$ is the upper bound ratio between the learning rate at local iteration 0 and $h'$, i.e., $\eta_{r,h'} \leq Q \eta_{r,h}$.

**Proof.** The proof can be found in Appendix, available online.

Then, we can get the following corollary:

**Corollary 5.1.1.** When we choose $\eta^m = \frac{1}{\sqrt{RH}}$ and $Q \leq (RH)^{1/4}$, we have:

$$\frac{1}{R} \sum_{r=1}^R \sum_{h=1}^H \mathbb{E} \| \nabla F^m_m(\bar{w}_{r,h}^m) \|^2$$

$$\leq \frac{2 L}{\sqrt{RH}} (F^m_m(\bar{w}_{1,1}^m) - F^{m^*}) + \frac{1}{\sqrt{RH}} \sigma^2 + \frac{1}{\sqrt{RH}} (H - 1)^2 G^2$$

$$= O\left(\frac{1}{\sqrt{RH}}\right)$$  

(19)

We can find that, the training process of multi-job FL converges to a stationary point of $f(w^*)$ with a convergence rate of $O\left(\frac{1}{\sqrt{RH}}\right)$ for each job.
6 EXPERIMENTS

In this section, we present the experimental results to show the efficiency of our proposed scheduling methods within MJ-FL. We compared the performance of Meta-Greedy, RLDS, and BODS with six baseline methods, i.e., Random [27], FedCS [28], Genetic [30], Greedy [40], Deep Neural Network (DNN), and Simulated Annealing (SA) [66].

6.1 Federated Learning Setups

In the experiment, we take three jobs as a group to be executed in parallel. We carry out the experiments with two groups, i.e., Group A with VGG-16 (VGG) [67], CNN (CNN-A-IID and CNN-A-non-IID) [68], and LeNet-5 (LeNet) [68], and Group B with Resnet-18 (ResNet) [69], CNN (CNN-B) [68], and Alexnet [70], while each model corresponds to one job. The complexity of the models is as follows: AlexNet < CNN-B < ResNet and LeNet < CNN (CNN-A-IID and CNN-A-non-IID) < VGG. We exploit the datasets of Cifar-10 [71], emnist-letters [72], emnist-digitals [72], Fashion-MNIST [73], and MNIST [68] in the training process.

CNN-A-IID comprises of two $3 \times 3$ convolution layers, one with 32 channels and the other with 64 channels. Each layer is followed by one batch normalization layer and $2 \times 2$ max pooling. Then, there are oneflatten layer and three fully-connected layers (1568, 784, and 26 units) after the two convolution layers. In addition, we make a simple modification of CNN-A-IID to CNN-A-non-IID since the convergence behavior of CNN on non-IID in Group A is not satisfiable. CNN-A-non-IID consists of three $3 \times 3$ convolution layers (32, 64, 64 channels, each of them exploits ReLU activations, and each of the first two convolution layers is followed by $2 \times 2$ max pooling), followed by one flatten layer and two fully-connected layers (64, 26 units). CNN-B consists of two $2 \times 2$ convolution layers (64, 32 channels, each of them exploits ReLU activations) followed by a flatten layer and a fully-connected layer, and each convolution layer is followed by a dropout layer with 0.05. In addition, the other parameters are shown in Tables 2 and 3.

DNN comprises of a flatten layer and 40 hidden layers before an Alpha Dropout layer and a fully-connected layer. The hidden layers consist of 20 units of a fully-connected layer and a batch normalization layer.

For the non-IID setting of each dataset, the training set is classified by category, and the samples of each category are divided into 20 parts. Each device randomly selects two categories and then selects one part from each category to produce its local training set. For the IID setting, each device randomly samples a specified number of images from each training set. In addition, we use 12 Tesla V100 GPUs to simulate an FL environment composed of a parameter server and 100 devices. We exploit the execution time, including the training time and communication time, measured in real edge devices as the parameters in Formula (4) to simulate the real execution. We use Formula (4) to simulate the capabilities of devices in terms of training time with the uniform sampling strategy, while the accuracy is the results from the actual training processes. In the experiment, we use corresponding target accuracy (for ease of comparison) in the place of target loss value.

6.2 Experimental Results

We first present the comparison between MJ-FL and Single-Job FL (SJ-FL). Then, we compare the proposed methods with baselines, e.g., Random [27], FedCS [28], Genetic [30], Greedy [40], DNN and SA, in both IID and non-IID settings. Afterward, we present the ablation experiments to show the impact of execution time and the data fairness in the cost model and the influence of $\Omega(r)$.

6.2.1 Comparison With Single-Job FL

In order to demonstrate the effectiveness of our proposed framework, i.e., MJ-FL, over the SJ-FL, we execute each group of jobs sequentially with FedAvg, which is denoted the Random method when adapted to multi-job FL. As shown in Tables 4, 5, and 6, MJ-FL outperforms SJ-FL (up to 1.68 times faster) with Random and the same accuracy. RLDS with MJ-FL outperforms Random with SJ-FL up to 15.38 times faster, and the advantage of BODS can be up to 8.83 times faster. Furthermore, Meta-Greedy can achieve much better performance, i.e., 19 times faster.

| TABLE 2 | Experimental Setup of Group A |
| Features | Cifar10 | Emnist-L | Emnist-D |
| Network model | VGG16 | CNN | LeNet5 |
| Parameters | 26,233K | 3,785K | 62K |
| Size | 50K/10K | 124.8K/20.8K | 240K/40K |
| Local epochs | 5 | 5 | 5 |
| Mini-batch size | 30 | 10 | 64 |

| TABLE 3 | Experimental Setup of Group B |
| Features | Fashion_mnist | Cifar10 | Mnist |
| Network model | CNN | ResNet18 | AlexNet |
| Parameters | 225K | 598K | 3,275K |
| Size | 60K/10K | 50K/10K | 60K/10K |
| Local epochs | 5 | 5 | 5 |
| Mini-batch size | 10 | 30 | 64 |

Size represents the training size and test samples (number of training samples/number of test samples). "Emnist-L" represents "Emnist-Letters" and "Emnist-D" represents "Emnist-Digitals".
in IID setups, the final accuracy of our proposed methods, i.e., BODS (up to 15.52% compared to Greedy), RLDS (up to 15.90% compared to Greedy), and Meta-Greedy (up to 15.33% compared to Greedy), significantly surpasses that of the other methods. In addition, the training time for a target accuracy of our proposed methods is significantly shorter than baseline methods, including the time for an individual job, i.e., the training time of each job (up to 8.19 times faster for BODS, 12.6 times faster for RLDS, and 12.73 times faster for Meta-Greedy), and the time for the whole training process, i.e., the total time calculated according to Formula (7), (up to 4.04 times for BODS, 5.81 times for RLDS and 8.16 times for Meta-Greedy). Slightly different from Non-IID setups results, RLDS performs better than BODS for both complex and simple jobs in terms of the convergence speed, while Meta-Greedy still converges the fastest to the target accuracy.

Evaluation With the non-IID Setting. When the decentralized data is of non-IID, the data fairness defined in Formula (5) has a significant impact on the accuracy. As shown in Figs. 6 and 7, the convergence speed of our proposed methods, i.e., RLDS, BODS and Meta-Greedy, is significantly faster than other methods. RLDS shows a significant advantage for complex jobs (VGG in Fig. 6a), while BODS shows advantage for relatively simple jobs in Groups A and B (please see details in Figs. 6 and 7). This is reasonable as we have much more parameters to adjust for complex jobs with RLDS, e.g., the learning rate, the $\epsilon$ decay rate, the structure of the LSTM etc, which can fit into a complex jobs. However, it may be complicated to fine-tune the parameters for a simple job with RLDS. In contrary, the execution of simple jobs can be directly well addressed by the Bayesian optimization. Meta-Greedy can lead to a good performance for both simple and complex jobs. As shown in Tables 4 and 5, the final accuracy of RLDS, BODS and Meta-Greedy significantly outperforms other methods (up to 44.6% for BODS, 39.4% for RLDS and 46.4% for Meta-Greedy), as well. Given a target accuracy, our proposed methods can achieve the accuracy within a shorter time, compared

| Table 4 | Convergence Accuracy and the Time Required to Achieve the Target Accuracy for Different Methods in Group A |
| --- | --- |
| Random | Genetic | FedCS | Greedy | BODS | RLDS | Meta-Greedy |
| VGG | 0.55 | 0.54 | 0.55 | 0.43 | 0.57 | 0.56 | **0.577** (0.55) | 2486 | 1164.3 | 1498.5 | / | 455.1 | 406.8 | **291.3** |
| Cnn | **0.90** | 0.80 | 0.80 | 0.83 | **0.90** | 0.897 | 0.90 | Cnn (0.80) | 44.25 | 95.85 | 27.39 | 43.04 | 15.88 | 17.6 | **11.9** |
| LeNet | 0.990 | 0.988 | 0.990 | 0.986 | **0.991** | **0.991** | **0.991** | LeNet (0.984) | 43.81 | 30.15 | 33.37 | 43.76 | 28.84 | 22.54 | **12.37** |
| Non-IID | | | | | | |
| VGG | **0.610** | 0.558 | 0.603 | 0.522 | 0.603 | 0.605 | 0.602 | VGG (0.55) | 126.9 | 231.4 | 87.5 | / | 57.7 | 43.81 | **30.34** |
| Cnn | **0.943** | 0.928 | 0.942 | 0.928 | **0.943** | 0.935 | 0.936 | Cnn (0.930) | 52.05 | 176.85 | 27.45 | 26.48 | 19.25 | 13.0 | **12.88** |
| LeNet | 0.9945 | 0.9928 | 0.9934 | 0.990 | **0.9946** | **0.9946** | 0.9936 | LeNet (0.99) | 14.94 | 6.03 | 7.12 | 17.98 | 5.18 | 4.02 | **2.02** |
| IID | | | | | | |
| VGG | 0.610 | 0.558 | 0.603 | 0.522 | 0.603 | 0.605 | 0.602 | VGG (0.55) | 126.9 | 231.4 | 87.5 | / | 57.7 | 43.81 | **30.34** |
| Cnn | 0.943 | 0.928 | 0.942 | 0.928 | 0.943 | 0.935 | 0.936 | Cnn (0.930) | 52.05 | 176.85 | 27.45 | 26.48 | 19.25 | 13.0 | **12.88** |
| LeNet | 0.9945 | 0.9928 | 0.9934 | 0.990 | **0.9946** | **0.9946** | 0.9936 | LeNet (0.99) | 14.94 | 6.03 | 7.12 | 17.98 | 5.18 | 4.02 | **2.02** |

The bold values represent the best accuracy or the shortest time. The numbers in parentheses represent the target accuracy, and "/" represents that the target accuracy is not achieved.

| Table 5 | The Convergence Accuracy and the Time Required to Achieve the Target Accuracy for Different Methods in Group B |
| --- | --- |
| Random | Genetic | FedCS | Greedy | BODS | RLDS | Meta-Greedy |
| ResNet | 0.546 | 0.489 | 0.523 | 0.403 | 0.583 | 0.562 | **0.590** | ResNet (0.50) | 852.9 | 621.5 | 402.3 | / | 219.8 | 168.9 | **149.8** |
| Cnn | 0.824 | 0.767 | 0.823 | 0.764 | 0.836 | 0.830 | 0.602 | Cnn (0.73) | 47.1 | 22.0 | 18.5 | 70.8 | 13.8 | 15.5 | **13.0** |
| AlexNet | 0.989 | 0.986 | 0.987 | 0.871 | 0.990 | 0.990 | **0.990** | AlexNet (0.976) | 140.57 | 60.0 | 82.87 | 181.2 | 59.2 | 53.87 | **45.0** |
| Non-IID | | | | | | |
| ResNet | 0.787 | 0.754 | 0.782 | 0.743 | 0.791 | 0.785 | **0.799** | ResNet (0.740) | 65.93 | 32.51 | 31.4 | 52.93 | 15.9 | 12.82 | **10.6** |
| Cnn | 0.868 | 0.867 | 0.868 | 0.868 | 0.869 | 0.869 | **0.871** | Cnn (0.867) | 120.19 | 38.99 | 89.6 | 36.13 | 32.83 | 19.7 | **16.13** |
| AlexNet | 0.9938 | 0.9938 | 0.9935 | 0.9935 | 0.9935 | 0.9935 | **0.9943** | AlexNet (0.9933) | 35.08 | 19.44 | 20.97 | / | 21.65 | 12.9 | **10.8** |
| IID | | | | | | |
| ResNet | 0.787 | 0.754 | 0.782 | 0.743 | 0.791 | 0.785 | **0.799** | ResNet (0.740) | 65.93 | 32.51 | 31.4 | 52.93 | 15.9 | 12.82 | **10.6** |
| Cnn | 0.868 | 0.867 | 0.868 | 0.868 | 0.869 | 0.869 | **0.871** | Cnn (0.867) | 120.19 | 38.99 | 89.6 | 36.13 | 32.83 | 19.7 | **16.13** |
| AlexNet | 0.9938 | 0.9938 | 0.9935 | 0.9935 | 0.9935 | 0.9935 | **0.9943** | AlexNet (0.9933) | 35.08 | 19.44 | 20.97 | / | 21.65 | 12.9 | **10.8** |

The numbers in parentheses represent the target accuracy, and "/" represents that the target accuracy is not achieved.
with baseline methods, in terms of the time for a single job, i.e., the training time of each job (up to 5.04 times shorter for BODS, 5.11 times shorter for RLDS and 7.53 times shorter for Meta-Greedy), and the time for the whole training process, i.e., the total time calculated based on Formula (7) (up to 4.15 times for BODS, 4.67 times for RLDS and 7.16 times for Meta-Greedy), for Groups A and B. In addition, among our proposed methods, Meta-Greedy always has the shortest time to achieve the target accuracy. We have similar observations with IID, while the advantage of Meta-Greedy is much more significant (up to 19.0 times shorter in terms of the time for a single job) than that of non-IID as shown in Tables 4 and 5. Besides, we can find that our proposed method (i.e., Meta-Greedy) leads to a better performance in the case of non-IID data due to dynamically enhancing the impact of data fairness to decrease the imbalance of devices.

Comparison in Other Cases. As shown in Figs. 5 and 7, the convergence speed corresponding to RLDS, BODS and Meta-Greedy is much higher than baseline methods with different target accuracy. The advantage of RLDS and BODS is up to 8.77 times faster for Target 1 (0.845), 13.96 times faster for Target 2 (0.856), and 27.04 times faster for Target 3 (0.865), compared with the baselines. In addition, Meta-Greedy outperforms the other six methods (four baselines and two scheduling methods, i.e., RLDS and BODS) up to 4.42 times faster for Target 1 (0.845), 6.65 times faster for Target 2 (0.856), and 5.61 times faster for Target 3 (0.865). Furthermore, According to Fig. 8, SA [66] corresponds to much worse performance (up to 91.4% slower and 3.5% lower accuracy) compared with our methods. We carry out experiments to compare our methods with DNNs [42], the performance of which is significantly worse (up to 90.5% slower and 26.3% lower accuracy) than our methods. In addition, we test other combinations of the two costs, which correspond to worse performance (up to 37.1% slower and 3.5% lower accuracy for the sum of squared costs, and 64.4% slower and 3.3% lower accuracy for multiplication) compared to the linear one (Formula (2)).

As RLDS can learn more information through a complex neural network, RLDS outperforms BODS for complex jobs (0.008 and 0.029 in terms of accuracy with VGG19 and
ResNet18, and 46.7% and 34.8% faster for the target accuracy of 0.7 with VGG19 and 0.5 with ResNet18; see details in Appendix, available in the online supplemental material). Due to the emphasis on the combination of data fairness and device capabilities, i.e., computation and communication capabilities, BODS can lead to high convergence accuracy and fast convergence speed for simple jobs (0.018 in terms of accuracy and 38% faster for the target accuracy of 0.97 with CNN; see details in Appendix, available in the online supplemental material). Meta-Greedy reconstructs the cost model based on six scheduling schemes and dynamically adjusts the parameters so that the impact of data fairness increases with the number of rounds, which leads to high convergence accuracy and fast convergence speed in both complex and simple jobs. BODS, RLDS and Meta-Greedy significantly outperform the baseline methods, while there are also differences among the four methods. The Greedy method prefers devices with high capacity, which leads to a significant decline in terms of the final convergence accuracy. The Genetic method can exploit randomness to achieve data fairness while generating scheduling plans, and the convergence performance is better than the Greedy method. The FedCS method optimizes the scheduling plan by randomly selecting devices, which improves the fairness of the device to a certain extent, and convergences faster than the Random method. We carry out an experiment with Group A with both IID and non-IID distribution, and find that Meta-Greedy with 6 methods significantly outperforms that with 2 methods in terms of accuracy (up to 0.091) and training speed (up to 65%) (see details in Appendix, available in the online supplemental material). In addition, after analyzing the log of the training process (see details in Appendix, available in the online supplemental material), we find Greedy and Genetic are extensively exploited, RLDS is selected at the beginning of the training process, BODS is chosen at the end of the training process, FedCS participates with less frequency, and Random is seldomly utilized. As Meta-Greedy can intelligently select a proper scheduling plan based on available methods, it results in a more efficient training process.

### 6.2.3 Ablation Study

In this section, we first present the ablation study to show the impact of the execution time and the data fairness. Then, we analyze the influence of $\Omega(r)$.
Impact of Data Fairness. As shown in Figs. 10 and 9, we conduct ablation experiments and find an evident decline in convergence accuracy when the cost model is only composed of execution time cost. From Figs. 10 and 9, we can see that the accuracy of the cost model composed of execution time (i.e., BODS (Time) and RLDS (Time)) is worse than that of both data fairness and execution time (i.e., BODS and RLDS) in most cases (up to 35.83% lower for RLDS (Time)). The abnormal case, when the convergence accuracy of BODS (Time) is slightly higher (about 1%) than the convergence accuracy of BODS (please see details in Fig. 10d) should be caused by randomness. Besides, from the Figs. 10 and 9, we can find that the impact of execution time on complex jobs (VGG in Fig. 10 and Resnet in Fig. 9) is significantly greater than that on simple jobs (LeNet in Fig. 10 and AlexNet in Fig. 9).

Impact of Execution Time. As shown in Figs. 10 and 9, the data fairness improves both the convergence speed (up to 9.35 times faster) and the accuracy (up to 15.3%). From Figs. 11 and 12, we can find that the convergence speed significantly decreases (up to 25.46 times slower for RLDS (Variance) compared with RLDS) when given a target accuracy with the cost model.
composed of only data fairness. In most cases, considering data fairness reduces the convergence speed and essentially does not affect the convergence accuracy. There are few cases that the job fails to achieve the convergence accuracy. In addition, from the results shown in Figs. 10 and 9, we find that the convergence accuracy in complex jobs (VGG in Fig. 11 and ResNet in Fig. 12) are more likely to decline when the cost model is only composed of data fairness, compared with that with simple jobs. Besides, the performance corresponding to the cost model with only data fairness is close to that of the ‘Random’ method in most setups.

Influence of Ω(r). As shown in Figures of Appendix, available in the online supplemental material, Meta-Greedy with Ω(r) = \sqrt{r} outperforms other methods, i.e., Ω(r) = r and Ω(r) = \log r, in terms of both convergence accuracy and convergence speed. Although the convergence accuracy of Meta-Greedy with Ω(r) = r is slightly higher than that with Ω(r) = \sqrt{r} in a few experiments, the convergence accuracy of Meta-Greedy with Ω(r) = \sqrt{r} in complex jobs is significantly lower than that with Ω(r) = r. In addition, Meta-Greedy with Ω(r) = \sqrt{r} takes less time to reach the target accuracy in previous rounds compared with Meta-Greedy with Ω(r) = r.
This indicates that when round $r$ is linearly related to $\Omega(r)$, the data fairness in later rounds influences the cost model too much and may lead to a lower convergence accuracy. Therefore, we avoid using $\Omega(r) = r$ due to the dramatic changes of the magnitude in the linear relationship. Meta-Greedy with $\Omega(r) = \sqrt{r}$ takes less time to reach the target accuracy in previous rounds compared with Meta-Greedy with $\Omega(r) = \log r$. In the meanwhile, the convergence accuracy of Meta-Greedy with $\Omega(r) = \log r$ is lower than that of Meta-Greedy with $\Omega(r) = \sqrt{r}$, which implies that the relatively gentle variation of $\Omega(r)$ with round $r$ leads to lower convergence accuracy and slower convergence speed. By contrast, Meta-Greedy with $\Omega(r) = \sqrt{r}$ performs best among them in terms of both convergence accuracy and convergence speed. Thus, we adopt Meta-Greedy with $\Omega(r) = \sqrt{r}$, which dynamically changes the influence of data fairness to obtain relatively optimal solution in both complex and simple jobs.

7 Conclusion

In this work, we proposed a new Multi-Job Federated Learning framework, i.e., MJ-FL. The framework is composed of a system model and three device scheduling methods. The system model is composed of a process for the parallel execution of multiple jobs and a cost model based on the capability of devices and data fairness. We proposed three device scheduling methods, i.e., RLDS for complex jobs and BODS for simple jobs, while Meta-Greedy for both complex and simple jobs, to efficiently schedule proper devices for each job based on the cost model. We carried out extensive experimentation with six real-life models and four datasets with IID and non-IID distribution. The experimental results show that MJ-FL outperforms the single-job FL, and that our proposed scheduling methods, i.e., BODS and RLDS, significantly outperform baseline methods (up to 44.6% in terms of accuracy, 12.6 times faster for a single job and 5.81 times faster for the total time). In addition, Meta-Greedy, the intelligent scheduling approach based on multiple scheduling methods (including the two proposed methods, i.e., RODS and BODS) significantly outperforms other methods (up to 46.4% in terms of accuracy, 12.73 times faster for a single job and 8.16 times faster for the total time).

References

[1] J. Liu et al., “From distributed machine learning to federated learning: A survey,” Knowl. Inf. Syst., vol. 64, no. 4, pp. 885–917, 2022.
[2] Official Journal of the European Union, “General data protection regulation,” 2016. Accessed: Feb. 12, 2021. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679
[3] Cybersecurity law of the people’s Republic of China, 2018. Accessed: Feb. 22, 2021. [Online]. Available: https://www.newamerica.org/cybersecurity-initiative/digitchina/blog/translation-cybersecurity-law-peoples-republic-china/
[4] California consumer privacy Act home page, 2018. Accessed: Feb. 14, 2021. [Online]. Available: https://www.ca.gov.
[5] W. B. Chik, “Communication-efficient learning of deep networks from decentralized data,” in Proc. Int. Conf. Artif. Intell. Statist., 2017, pp. 1273–1282.
[6] P. Kairouz et al., “Advances and open problems in federated learning,” 2019, arXiv:1912.04977.
[7] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning: Concept and applications,” ACM Trans. Intell. Syst. Technol., vol. 10, no. 2, pp. 1–19, 2019.
[8] A. Smola and S. Narayanamurthy, “An architecture for parallel topic models,” Proc. VLDB Endowment, vol. 3, no. 1/2, pp. 703–710, 2010.
[9] J. Liu et al., “Large-scale knowledge distillation with elastic heterogeneous computing resources,” Concurrency Comput. Pract. Exper., vol. e7272, pp. 1–16, 2022.
[10] J. Liu et al., “HeterPS: Distributed deep learning with reinforcement learning based scheduling in heterogeneous environments,” 2021, arXiv:2111.10365.
[11] L. L. Pilla, “Optimal task assignment for heterogeneous federated learning devices,” in Proc. IEEE Int. Parallel Distrib. Process. Symp., 2021, pp. 661–670.
[12] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, “Federated multi-task learning,” in Proc. Annu. Conf. Neural Inf. Process. Syist., 2017, pp. 4427–4437.
[13] J. Han, M. M. Rafique, L. Xu, A. R. Butt, S.-H. Lim, and S. S. Vazhkudai, “MARBLE: A multi-GPU aware job scheduler for deep learning data systems,” in Proc. IEEE/ACM Int. Symp. Cluster Cloud Internet Comput., 2020, pp. 272–281.
[14] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, “Federated learning: Challenges, methods, and future directions,” IEEE Signal Process. Mag., vol. 37, no. 3, pp. 50–60, May 2020.
[15] L. Li, D. Shi, R. Hou, H. Li, M. Pan, and Z. Han, “To talk or to work: Flexible communication compression for energy efficient federated learning over heterogeneous mobile edge devices,” in Proc. IEEE Conf. Comput. Commun., 2021, pp. 1–10.
[16] W. Zhao, J. Zhang, D. Xie, Y. Qian, R. Jia, and P. Li, “AllBox: CTR prediction model training on a single node,” in Proc. ACM Int. Conf. Inf. Knowl. Manage., 2019, pp. 319–328.
[17] H. Guo, W. Guo, Y. Gao, R. Tang, X. He, and W. Liu, “ScaleFreeCTR: MixCache-based distributed training system for CTR models with huge embedding tables,” in Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Infor. Retrieval, 2021, pp. 1269–1278.
[18] A. Hard et al., “Federated learning for mobile keyboard prediction,” 2018, arXiv:1811.03604.
[19] X. Fang, J. Huang, F. Wang, L. Liu, Y. Sun, and H. Wang, “SMLM: Self-supervised meta-learner for en route travel time estimation at Baidu maps,” in Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2021, pp. 2840–2848.
[20] C. Qin et al., “DuerQuiz: A personalized question recommender system for intelligent job interview,” in Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2019, pp. 2165–2173.
[21] M. Masterson, “Baidu’s deep speech recognition bests Google, Apple, and Bing,” Speech Technol. Mag., vol. 20, no. 1, pp. 12–13, 2015.
[22] T. Noura and P. Triantafillou, “Load distribution fairness in P2P data management systems,” in Proc. IEEE Int. Conf. Data Eng., 2007, pp. 396–405.
[23] A. Finkelstein, M. Harman, S. A. Mansouri, J. Ren, and Y. Zhang, “Fairness analysis in requirements assignments,” in Proc. IEEE Int. Requirements Eng. Conf., 2008, pp. 115–124.
[24] J. Du and J. Y.-T. Leung, “Complexity of scheduling parallel task systems,” SIAM J. Discrete Math., vol. 2, no. 4, pp. 473–487, 1989.
[25] L. Liu, H. Xu, G. Sun, L. Luo, Q. Jin, and S. Luo, “Job scheduling for distributed machine learning in optical wan,” Future Gener. Comput. Syst., vol. 112, pp. 549–560, 2020.
[26] T. Nishio and Y. Sato, “AutoML for HPC systems,” in Proc. Int. Conf. Artif. Intell. Statist., 2017, pp. 1273–1282.
[27] T. Nishio and Y. Sato, “Client selection for federated learning with heterogeneous resources in mobile edge,” in Proc. IEEE Int. Conf. Commun., 2019, pp. 1–7.
[28] S. Abdulrahman, H. Toot, A. Mourad, and C. Talhi, “FedMCCS: Multicriterion client selection model for optimal IoT federated learning,” IEEE Internet of Things J., vol. 8, no. 6, pp. 4723–4735, Mar. 2021.
[29] Y. Nishita, S. Garg, A. Chan, and R. Calheiros, “Scheduling algorithms for efficient execution of stream workflow applications in multicloud environments,” IEEE Trans. Serv. Comput., vol. 15, no. 2, pp. 860–875, Mar./Apr. 2022.
[30] C. Zhou et al., “Efficient device scheduling with multi-job federated learning,” in Proc. AAAI Conf. Artif. Intell., 2022, pp. 9971–9979.
[32] K. Bonawitz et al., “Towards federated learning at scale: System design,” in Proc. Int. Conf. Mach. Learn. Syst., 2019, pp. 374–388.

[33] Y. Liu et al., “FedVision: An online visual object detection platform powered by federated learning,” in Proc. AAAI Conf. Artif. Intell., 2020, pp. 13172–13179.

[34] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, N. Hoang, and Y. Khazaeni, “Baysian nonparametric federated learning of neural networks,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 7252–7261.

[35] H. Wang, M. Yurochkin, Y. Sun, D. Papapioupolous, and Y. Khazaeni, “Federated learning with matched averaging,” in Proc. Int. Conf. Learn. Representations, 2020, pp. 1–16.

[36] H. Zhang, J. Liu, J. Jia, Y. Zhou, and H. Dai, “FedDUAP: Federated learning with dynamic update and adaptive pruning using shared progressive server,” in Proc. Int. Joint Conf. Artif. Intell., 2022, pp. 1–7.

[37] J. Jin, R. Zhou, L. Xu, J. Liu, and D. Dou, “Accelerated federated learning with decoupled adaptive optimization,” in Proc. Int. Conf. Mach. Learn., 2022, pp. 10298–10322.

[38] D. Chen et al., “Matching theory based low-latency scheme for multi-task federated learning in MEC networks,” IEEE Internet Things J., vol. 8, no. 14, pp. 11415–11426, Jul. 2021.

[39] P. Sun, Z. Guo, J. Jiang, J. Li, J. Lan, and Y. Hu, “DeepWeave: Accelerating job completion time with deep reinforcement learning-based coflow scheduling,” in Proc. Int. Joint Conf. Artif. Intell., 2020, pp. 3314–3320.

[40] W. Shi, S. Zhou, and Z. Niu, “Device scheduling with fast convergence for wireless federated learning,” in Proc. IEEE Int. Conf. Commun., 2020, pp. 1–6.

[41] K. K. Kim, Y. Sung, and S. Park, “A probabilistic machine learning approach to scheduling parallel loops with Bayesian optimization,” IEEE Trans. Parallel Distrib. Syst., vol. 32, no. 7, pp. 1815–1827, Jul. 2021.

[42] Z. Zang et al., “Hybrid deep neural network scheduler for job shop problem based on convolution versus two-dimensional transformation,” Comput. Intel. Neurosci., vol. 2019, pp. 1–20, 2019.

[43] Q. Chen, Z. Zheng, C. Hu, D. Wang, and F. Liu, “On-edge multi-task transfer learning: Model and practice with data-driven task allocation,” IEEE Trans. Parallel Distrib. Syst., vol. 31, no. 6, pp. 1357–1371, Jun. 2020.

[44] M. K. Emani and M. O’Boyle, “Celebrating diversity: A mixture of experts approach for runtime mapping in dynamic environments,” in Proc. ACM SIGPLAN Conf. Program. Lang. Des. Implement., 2015, pp. 499–508.

[45] M. Li et al., “Solving distributed machine learning with the parameter server,” in Proc. USENIX Symp. Oper. Syst. Des. Implementation, 2014, pp. 583–598.

[46] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, “On the convergence of FedAvg on non-IID data,” in Proc. Int. Conf. Learn. Representations, 2020, pp. 1–26.

[47] F. Zhou and G. Cong, “On convergence properties of a k-step averaging stochastic gradient descent algorithm for nonconvex optimization,” in Proc. Int. Joint Conf. Artif. Intell., 2018, pp. 3219–3227.

[48] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” in Proc. Int. Conf. Theory Appl. Cryptographic Techn., 1999, pp. 223–238.

[49] C. Dwork, “Differential privacy: A survey of results,” in Proc. Int. Conf. Theory Appl. Models Computat., 2008, pp. 1–19.

[50] W. Shi, S. Zhou, Z. Niu, M. Jiang, and L. Gou, “Joint device scheduling and resource allocation for latency constrained wireless federated learning,” IEEE Trans. Wireless Commun., vol. 20, no. 1, pp. 453–467, Jan. 2021.

[51] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, “Speeding up distributed machine learning using codes,” IEEE Trans. Inf. Theory, vol. 64, no. 3, pp. 1514–1529, Mar. 2018.

[52] S. Petrangeli, M. Claeyys, S. Latre, J. Famaey, and F. De Turck, “A multi-agent Q-learning-based framework for achieving fairness in HTTP adaptive streaming,” in Proc. IEEE Netw. Operations Manage. Symp., 2014, pp. 1–9.

[53] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, “A survey on Bias and fairness in machine learning,” ACM Comput. Surv., vol. 54, no. 6, pp. 1–35, 2021.

[54] M. H. Cheon, J. Mao, R. Guo, “Optimis: An efficient dynamic resource scheduler for deep learning clusters,” in Proc. 13th EuroSys Conf, 2018, pp. 1–14.

[55] P. Toth, “Optimization engineering techniques for the exact solution of np-hard combinatorial optimization problems,” Eur. J. Oper. Res., vol. 125, no. 2, pp. 222–238, 2000.

[56] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, “Gaussian process optimization in the bandit setting: No regret and experimental design,” in Proc. Int. Conf. Mach. Learn., 2010, pp. 1015–1022.

[57] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, “Taking the human out of the loop: A review of Bayesian optimization,” Proc. IEEE, vol. 104, no. 1, pp. 148–175, Jan. 2016.

[58] C. K. Williams and C. E. Rasmussen, Gaussian Processes for Machine Learning, 2nd ed., Cambridge, MA, USA: MIT Press, 2006.

[59] D. R. Jones, M. Schonlau, and W. J. Welch, “Efficient global optimization of expensive black-box functions,” J. Glob. Optim., vol. 13, no. 4, pp. 455–492, 1998.

[60] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, “Learning scheduling algorithms for data processing clusters,” in Proc. ACM Special Int. Group Data Commun., 2019, pp. 270–288.

[61] Z. Xia and D. Zhao, “Online reinforcement learning by Bayesian inference,” in Proc. Int. Joint Conf. Neural Netw., 2015, pp. 1–6.

[62] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” Mach. Learn., vol. 8, pp. 229–256, 1992.

[63] B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning,” in Proc. Int. Conf. Learn. Representations, 2017, pp. 1–16.

[64] G. Li et al., “FedHiSyn: A hierarchical synchronous federated learning framework for resource and data heterogeneity,” in Proc. Int. Conf. Parallel Process., 2022, pp. 1–10.

[65] S. Reddi et al., “Adaptive federated optimization,” 2020, arXiv:2003.00295.

[66] P. J. V. Laarhoven and H. E. Aarts, “Simulated annealing,” in Simulated Annealing: Theory and Applications. Berlin, Germany: Springer, 1987, pp. 7–15.

[67] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in Proc. Int. Conf. Learn. Representations, 2015, p. 14.

[68] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[69] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 770–778.

[70] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Proc. Annu. Conf. Neural Inf. Process. Syst., 2012, pp. 1106–1114.

[71] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” University of Toronto, Toronto, ON, Canada, Tech. Rep., 2009, pp. 1–60.

[72] G. Cohen, S. Atshar, J. Tapson, and A. Van Schaik, “EMNIST: Extending MNIST to handwritten letters,” in Proc. Int. Joint Conf. Neural Netw., 2017, pp. 2921–2926.

[73] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms,” 2017, arXiv:1708.07747.
Juncheng Jia received the BEng degree in computer science and technology from Zhejiang University, in 2005, and the PhD degree in computer science from the Hong Kong University of Science and Technology, in 2009. He worked as a postdoctoral fellow with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. He is currently working as an associate professor with the School of Computer Science and Technology, Soochow University, China. His research interests include federated learning and wireless networks.

Beichen Ma is currently working toward the BSc degree with the School of Advanced Technology, Xi’an Jiaotong-Liverpool University. His research interests include distributed machine learning and federated learning.

Chendi Zhou received the BEng degree from the School of Computer Science, Henan University of Engineering, in 2019, and the MEng degree from the School of Computer Science and Technology, Soochow University, in 2022. Her research interests include federated learning and mobile computing.

Jingbo Zhou received the BE degree from Shan-dong University, in 2009, and the PhD degree from the National University of Singapore, in 2014. He is a staff research scientist with Business Intelligent Lab, Baidu Research, working on machine learning problems for both scientific research and business applications, with a focus on spatial temporal data mining, knowledge graphs, and geometric deep learning. He has published more than 30 research papers in top venues, such as SIGMOD, KDD, VLDB, ICDE, IEEE Transactions on Knowledge and Data Engineering and AAAI. He has served regularly on the program committees among top conferences in the data mining and artificial intelligence communities, like KDD, AAAI, IJCAI, ACL, CIKM, and SDM.

Yang Zhou received the PhD degree from the College of Computing, Georgia Institute of Technology. He is currently an assistant professor with the Department of Computer Science and Software Engineering, Auburn University. His current research interests lie in the areas of trustworthy machine learning, parallel, distributed, and federated learning, graph machine learning, and natural language processing. He has published more than 80 research papers in top venues of machine learning (ICML, NeurIPS), data mining (KDD, ICDM, ACM Transactions on Knowledge Discovery and Data Mining), artificial intelligence (AAAI, IJCAI, ACM Transactions on Intelligent Systems and Technology), natural language processing (ACL, EMNLP), Web (WWW, ACM Transactions on the Web), high performance computing (HPDC, SC), database systems (VLDB, ICDE, IEEE Transactions on Knowledge and Data Engineering, The International Journal on Very Large Data Bases), networking (IEEE Journal on Selected Areas in Communications, ACM Transactions on Internet Technology), web services (ICWS, IEEE Transactions on Services Computing), and software engineering (ISSSTA). The developed models and frameworks have been widely used by many research groups and six papers have been included in reading lists and taught in courses at universities worldwide. He was named as KDD Rising Star by Microsoft Academic Search and Microsoft Research Asia in 2016.

Huaiyu Dai (Fellow, IEEE) received the BE and MS degrees in electrical engineering from Tsinghua University, Beijing, China, in 1996 and 1998, respectively, and the PhD degree in electrical engineering from Princeton University, Princeton, NJ, in 2002. He was with Bell Labs, Lucent Technologies, Holmdel, NJ, in summer 2000, and with AT&T Labs-Research, Middletown, NJ, in summer 2001. He is currently a professor of electrical and computer engineering with NC State University, Raleigh, holding the title of University Faculty Scholar. His research interests include the general areas of communications, signal processing, networking, and computing. His current research focuses on machine learning and artificial intelligence for communications and networking, multilayer and interdependent networks, dynamic spectrum access and sharing, as well as security and privacy issues in the above systems. He has served as an area editor of the IEEE Transactions on Communications, a member of the Executive Editorial Committee of the IEEE Transactions on Wireless Communications, and an editor of the IEEE Transactions on Signal Processing. Currently, he serves as an area editor of the IEEE Transactions on Wireless Communications. He was a co-recipient of best paper awards at 2010 IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2010), 2016 IEEE INFOCOM BIGSECURITY Workshop, and 2017 IEEE International Conference on Communications (ICC 2017). He received Qualcomm Faculty Award in 2019.

Dejing Dou (Senior Member, IEEE) received the bachelor’s degree from Tsinghua University, China, in 1996, and the PhD degree from Yale University, in 2004. He is the chief data scientist with BCG in Greater China. He was the head of Big Data Lab (BDL) and Business Intelligence Lab (BIL), Baidu Research. He was also a full professor from the Computer and Information Science Department, University of Oregon and led the Advanced Integration and Mining (AIM) Lab. He had been the director of the NSF IUCRC Center for Big Learning (CBL) from 2018 to 2020. He was a visiting Associate Professor with Stanford Center for Biomedical Informatics Research during 2012-2013. His research areas include artificial intelligence, data integration, NLP and health informatics. He has published more than 200 research papers, some of which appear in prestigious conferences and journals like AAAI, IJCAI, ICML, NeurIPS, ICLR, KDD, ICML, ACM, EMNLP, CVPR, ICCV, CIKM, ISWC, Artificial Intelligence Journal, Journal of Machine Learning Research, ACM Transactions on Knowledge Discovery from Data, Knowledge and Information Systems, and Journal of Intelligent Information Systems with more than 6000 Google Scholar citations. His DEXA’15 paper received the Best Paper Award. His KDD’07 paper was nominated for the Best Research Paper Award. His COLING’18 paper was Area Chair Favorites (excellent). He is on the editorial boards of the Journal on Data Semantics, Journal of Intelligent Information Systems, and PLoS One. He is an editor-in-chief of the AIMS Electronic Research Archive. He has been serving as program committee members for major international conferences and as program co-chairs for five of them. He has received more than 5 million PI research grants from the NSF and the NIH. He is a senior member of AAAI, and ACM.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.