FedSA: A Semi-Asynchronous Federated Learning Mechanism in Heterogeneous Edge Computing

Qianpiao Ma, Yang Xu, *Member, IEEE*, Hongli Xu, *Member, IEEE*, Zhida Jiang, Liusheng Huang, *Member, IEEE*, and He Huang, *Member, IEEE*

***Abstract*— Federated learning (FL) involves training machine learning models over distributed edge nodes (*i.e.*, workers) while facing three critical challenges, edge heterogeneity, Non-IID data and communication resource constraint. In the synchronous FL, the parameter server has to wait for the slowest workers, leading to significant waiting time due to edge heterogeneity. Though asynchronous FL can well tackle the edge heterogeneity, it requires frequent model transfers, resulting in massive communication resource consumption. Moreover, the different relative frequency of workers participating in asynchronous updating may seriously hurt training accuracy, especially on Non-IID data. In this paper, we propose a semi-asynchronous federated learning mechanism (FedSA), where the parameter server aggregates a certain number of local models by their arrival order in each round. We theoretically analyze the quantitative relationship between the convergence bound of FedSA and different factors, *e.g.*, the number of participating workers in each round, the degree of data Non-IID and edge heterogeneity. Based on the convergence bound, we present an efficient algorithm to determine the number of participating workers to minimize the training completion time. To further improve the training accuracy on Non-IID data, FedSA deploys adaptive learning rates for workers by their relative participation frequency. We extend our proposed mechanism to the dynamic and multiple learning tasks scenarios. Experimental results on the testbed show that our proposed mechanism and algorithms address the three challenges more effectively than the state-of-the-art solutions.**

***Index Terms*— Edge computing, federated learning, semi-asynchronous mechanism, heterogeneity, non-IID.**

I. Introduction

With the increasing popularity of the Internet of Things (IoT), significant amounts of data are generated from the physical world per second [1]–[3]. Traditionally, these huge amounts of data are forwarded to the remote cloud for processing and training, which may cause a significant delay due to long-distance transmission, and potential privacy leakage. To this end, *edge computing* is proposed to shift more computation to the network edge, enabling efficient data processing locally. Besides, it also promotes the landing of federated learning (FL), which performs distributed machine learning over edge nodes [4]–[6].

An FL system usually consists of one or multiple parameter servers and a large number of edge nodes (*i.e.*, workers), following the typical parameter server (PS) architecture [7]. For simplicity, we assume one parameter server in the system. Nevertheless, our solution can be easily extended to the situation of multiple parameter servers. The FL procedure usually consists of a certain number of *rounds* until model convergence. In each round, workers perform the local updating, and the parameter server aggregates the local models from workers. Since the workers expose their updated local models to the parameter server instead of their raw data, FL can efficiently protect workers' privacy [8].

To implement effective FL, we should take the following factors and challenges into considerations.

- **Edge Heterogeneity:** Workers may be various devices, with diverse CPU capacities, data size and network connections. As a result, the required time to perform local updating and receive/upload models may vary significantly. For example, if a device has poor computation capacity or is under weak wireless channel conditions, it will take a long time to perform local updating or model delivering.
- **Non-IID Data:** A worker collects data from its physical location directly. As a result, its local data often cannot be regarded as the samples drawn uniformly from the overall distribution. In other words, the data among workers usually are not independent-and-identically-distributed (*i.e.*, Non-IID) [9]. It has been pointed out that the performance of FL will be significantly degraded in the presence of Non-IID data, in terms of the convergence rate and the model accuracy [10]–[12].
- **Communication Resource Constraint:** The communication resource between workers and the parameter server is usually constrained in edge computing [13]–[15]. Therefore, the parameter server may easily become a communication bottleneck if it frequently distributes/receives models to/from workers [16].

Many well-known FL mechanisms have been proposed since the first work in 2016 [8]. We summarize the advantages and disadvantages of some typical FL mechanisms in Table I.

Manuscript received March 1, 2021; revised September 13, 2021; accepted September 22, 2021. Date of publication October 6, 2021; date of current version November 22, 2021. This work was supported in part by the National Science Foundation of China (NSFC) under Grant 62132019, Grant 61936015, Grant 62102391, and Grant U1709217. *(Corresponding authors: Yang Xu; Hongli Xu.)*

Qianpiao Ma, Yang Xu, Hongli Xu, Zhida Jiang, and Liusheng Huang are with the School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China, and also with the Suzhou Institute for Advanced Study, University of Science and Technology of China, Suzhou, Jiangsu 215123, China (e-mail: maqiu@mail.ustc.edu.cn; xuyangcs@ustc.edu.cn; xuhongli@ustc.edu.cn; zdjiang@mail.ustc.edu.cn; lshuang@ustc.edu.cn).

He Huang is with the School of Computer Science and Technology, Soochow University, Suzhou 215123, China (e-mail: huangh@suda.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2021.3118435.

Digital Object Identifier 10.1109/JSAC.2021.3118435

TABLE I
COMPARISON OF DIFFERENT FL APPROACHES

FL Mechanisms		Participating Workers	Round Duration	Training Time	Handling Non-IID	Scalability	Communication Consumption
Synchronous	FedAvg [9]	Full&Partial	Long	Long	Poor	Poor	Low
	AdaptFL [17]	Full	Long	Long	Good	Poor	Low
	FAVOR [18]	Partial	Long	Long	Good	Poor	Low
Asynchronous	FedAsync [19]	One	Short	Long	Poor	Good	High
	ASO-fed [20]	One	Short	Long	Poor	Good	High
Semi-asynchronous	CSAFL [21]	One	Short	Long	Poor	Good	High
	SAFA [22]	Partial	Short	Long	Poor	Good	Low
	Ours	Partial	Short	Short	Good	Good	Low

There are three main types of FL mechanisms. The first type is synchronous FL [9], [17], [18], [23], [24]. Specifically, after the parameter server receives the local models from all workers or a *pre-selected* subset of workers, it aggregates these local models into a new global model for the next round. There are two cases. 1) All workers participate in the global updating, called full participation in each round [17], [23]. Due to edge heterogeneity, the completion time of all workers varies significantly, which is known as the *straggler* problem [25]. Since the single-round duration depends on the maximum completion time of all workers, it may lead to a long or even unacceptable training time. 2) A pre-selected subset of workers participates in the global updating, called partial participation [18], [24]. Nevertheless, only partial workers participate in each round of training, while the others are idle, which makes the computation resource in the system under low utilization and the global model needs more time until convergence. Furthermore, in practical scenarios, since a worker, that participates in the global updating, may fail due to mobility, power-off or weak connection, the parameter server should wait a relatively long time even until the current round times out, leading to poor scalability.

The second type is asynchronous FL, in which the parameter server performs global updating as soon as it receives a local model from any worker, and then sends the updated global model back [19], [20], [26]. Since there is no need to wait for other local models on the parameter server, this mechanism has a short single-round duration. Besides, when a worker participates in the global updating, the remaining workers are still training and uploading their local models, so there are no idle workers. However, the existing asynchronous mechanisms have two main drawbacks. Firstly, the asynchronous mechanisms may cause significant communication resource consumption as they require frequent model transfers between workers and the parameter server [27]. Secondly, the existence of stragglers leads to the different relative frequency of workers participating in asynchronous updating, which leads to the large staleness of their local models relative to the global model [28], and causes lower training accuracy, especially on Non-IID data.

To overcome the detrimental effects of stragglers in synchronous FL (*i.e.*, long single-round duration) and asynchronous FL (*i.e.*, staleness), the third type is semi-asynchronous FL, which forces to synchronize stale local models while performing global updating asynchronously [21], [22], [29]. However, these semi-asynchronous FL mechanisms assign the number of workers participating in the global updating without quantitatively considering any factors, such as the data distribution among workers and edge heterogeneity (*e.g.*, the various completion time of workers), which may vastly affect the training performance. For example, the authors [21], [29] propose that only one worker participates in the global updating of each round. However, it may easily cause to synchronize some slow workers' local models due to their large staleness before they complete local training. As a result, these slow workers may never participate in the global updating, which decreases the training accuracy, especially on Non-IID data. The authors of SAFA [22] claim that the training performance is closely associated with the number (*e.g.*, M) of workers participating in the global updating. However, they do not give the quantitative analysis between training performance and the number of participating workers, data distribution, workers' completion time, nor do they propose the method to determine the optimal value of M. Actually, it is a major challenge to determine the optimal M for different data distributions and edge heterogeneity scenarios.

To relieve these disadvantages, we propose a novel semi-asynchronous federated learning (FedSA) mechanism. Specifically, in each round, after the parameter server receives the local models from a certain number (*e.g.*, M) of workers, it aggregates those local models, depending on workers' arrival order at the parameter server. Furthermore, we theoretically analyze the quantitative relationship between training performance and several factors, such as the number M of participating workers, data distribution, edge heterogeneity and the communication budget. Based on this, we propose an efficient algorithm to determine the optimal value of M according to edge heterogeneity and data distribution among workers, so as to minimize the training time given the communication budget. The main contributions are summarized as follows:

- We design a semi-asynchronous FL mechanism, called FedSA, for heterogeneous edge computing. We analyze the relationship between the convergence bound of FedSA and different factors, *e.g.*, parameter M, the degree of data Non-IID among workers and edge heterogeneity. The convergence analysis of FedSA is also provided.

TABLE II
KEY NOTATIONS

Symbol	Semantics
$\mathcal{V}$	The set of workers $\{v_1, v_2, ..., v_N\}$
$\mathcal{V}_k$	The workers participating in the global updating in round k
$\mathcal{D}_i$	The local dataset on worker v_i
D_i	The size of $\mathcal{D}_i$
$F(\mathbf{w})$	The global loss function
$F_i(\mathbf{w})$	The local loss function of v_i
$F(\mathbf{w}^*)$	The optimal value of $F(\mathbf{w})$
$\mathbf{w}_k$	The global model in round k
$\mathbf{w}_k^i$	The local model derived by local updating $\mathbf{w}_{k-1}$ on v_i
$\mathbf{x}_k^i$	The received local model from v_i in round k
τ_k^i	The staleness of v_i in round k
η_i	The learning rate of worker v_i
f_i	The relative frequency of v_i participating in the global updating
λ	The global learning rate
p_i	The model preparation time of v_i
r_i	The duration from the current time to the arrival of v_i's model
t_k	The completion time of round k

- We propose an efficient algorithm to determine the optimal M so as to minimize the training time given the communication budget. Furthermore, we extend the proposed algorithm to the dynamic and multiple learning tasks scenarios.
- To further improve the training accuracy on Non-IID data, we deploy adaptive learning rates for workers by the relative frequency of their participating in the global updating.
- We implement our proposed mechanism and algorithm on a testbed. Experimental results show that the proposed solution is more efficient than the state-of-the-art solutions on the datasets with different degrees of Non-IID.

II. FEDSA MECHANISM AND PROBLEM FORMULATION

A. Federated Learning (FL)

For ease of expression, some key notations in this paper are listed in Table II. We consider a federated learning system with a set of workers $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$. Each worker v_i trains a model on its local dataset $\mathcal{D}_i$, with the size of $D_i \triangleq |\mathcal{D}_i|$. The total amount of data is $D = \sum_{v_i \in \mathcal{V}} D_i$. Suppose that $\mathcal{D}_i$ holds the training data: $d_{i,1}$, $d_{i,2}$, ..., d_{i,D_i}. The loss function of worker v_i is defined as

$$F_i(\mathbf{w}) \triangleq \frac{1}{D_i} \sum_{d_{i,j} \in \mathcal{D}_i} f(\mathbf{w}; d_{i,j}), \tag{1}$$

Algorithm 1 Semi-Asynchronous Federated Learning (FedSA)

1: $k = 0$
2: **while** $F(\mathbf{w}_k) - F(\mathbf{w}^*) > \varepsilon$ **do**
3: **Processing at Each Worker** v_i
4: **if** Receive $\mathbf{w}_k$ from the server **then**
5: Update local model by Eq. (3)
6: Upload local model $\mathbf{w}_{k+1}^i$
7: **Processing at the Parameter Server**
8: $\mathcal{V}_k = \varnothing$
9: **while** $|\mathcal{V}_k| < M$ **do**
10: Receive local model $\mathbf{x}_k^i$ from worker v_i
11: $\mathcal{V}_k = \mathcal{V}_k \cup \{v_i\}$
12: Update global model by Eq. (5)
13: **for** each $v_i \in \mathcal{V}$ **do**
14: **if** $v_i \in \mathcal{V}_k$ or $\tau_k^i > \tau^0$ **then**
15: Distribute updated model $\mathbf{w}_k$ and learning rate η_i
16: $k = k + 1$
17: **Return** the final global model $\mathbf{w}_k$

where $\mathbf{w}$ is the parameter vector and $f(\cdot)$ is a user-specified loss function, *e.g.*, linear regression, logistic regression, support vector machine (SVM). The global loss function on all the distributed datasets is defined as

$$F(\mathbf{w}) \triangleq \frac{\sum_{v_i \in \mathcal{V}} D_i F_i(\mathbf{w})}{D}. \tag{2}$$

The learning problem is to obtain the optimal vector $\mathbf{w}^*$ so as to minimize $F(\mathbf{w})$, *i.e.*, $\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} F(\mathbf{w})$.

B. Semi-Asynchronous Federated Learning (FedSA)

We propose the semi-asynchronous FL mechanism, called FedSA, which is formally described in Alg. 1. Since the parameter server architecture consists of a parameter server and a set of workers, we describe the algorithm on both the worker side and the server side.

On the worker side (Lines 3-6), after a worker v_i receives a global model $\mathbf{w}_k$, it performs several iterations for local updating by

$$\mathbf{w}_{k+1}^i = \mathbf{w}_k - \eta_i \nabla F_i(\mathbf{w}_k), \tag{3}$$

where η_i is the learning rate (*i.e.*, step size) of worker v_i. Due to edge heterogeneity, we adopt the adaptive learning rate for model training, which will be discussed in Section II-D. Then worker v_i uploads its updated local model $\mathbf{w}_{k+1}^i$ to the parameter server. For ease of interpretation, we assume that each worker performs only one iteration in each local updating. Nevertheless, the proposed mechanism and analyses are still applicable to the case with multiple iterations in each local updating, as long as changing the definition of $\mathbf{w}_{k+1}^i$ to the model after multiple iterations in Eq. (3).

On the parameter server side (Lines 7-15), let $\mathcal{V}_k$ be the set of workers participating in the global updating in round k, with $|\mathcal{V}_k| = M$. The received local model from worker v_i in round k is denoted as $\mathbf{x}_k^i$. Since the workers participate in the global updating asynchronously, $\mathbf{x}_k^i$ is not necessarily equal to

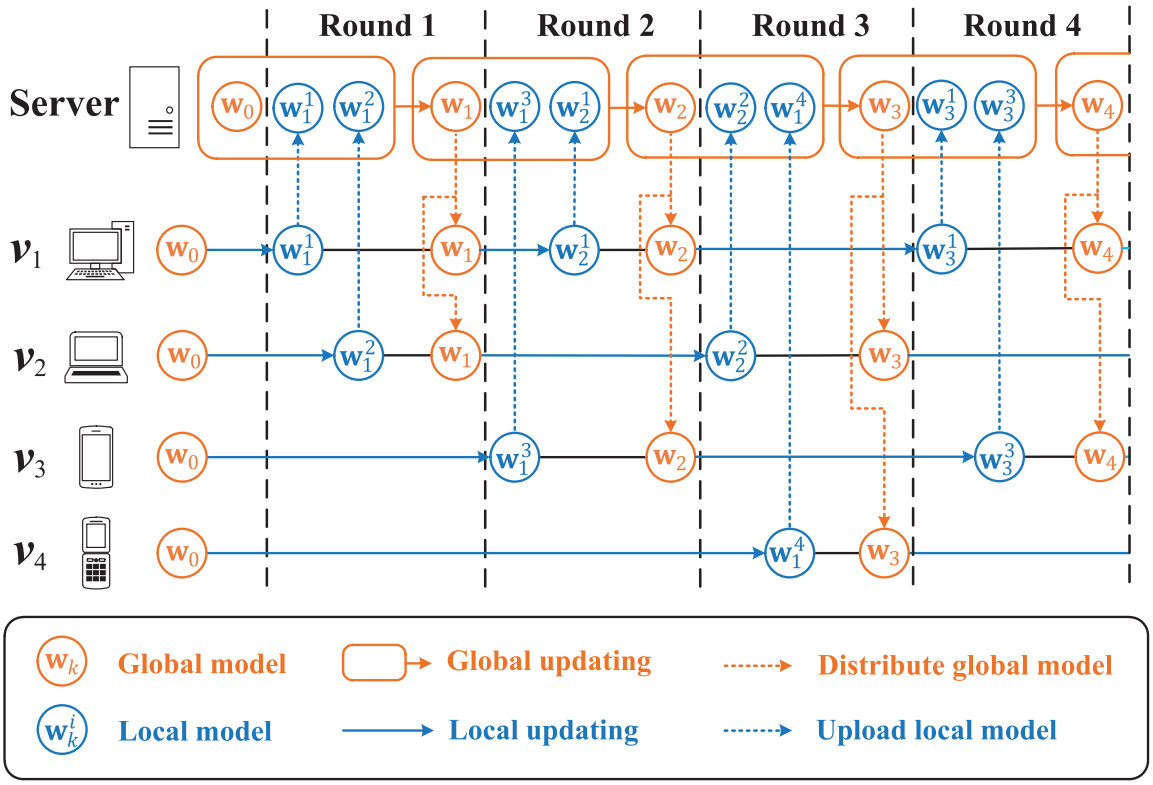


Fig. 1. Illustration of the FedSA mechanism when $M = 2$.

$\mathbf{w}_k^i$. Let τ_k^i be the interval between the current round k and the last received global model version by worker v_i, called the *staleness*. In fact, $\mathbf{x}_k^i$ is equal to $\mathbf{w}_{k-\tau_k^i}^i$, and $\mathbf{w}_{k-\tau_k^i}^i$ is trained from a previous version of the global model on v_i, *i.e.*,

$$\mathbf{x}_k^i = \mathbf{w}_{k-\tau_k^i}^i = \mathbf{w}_{k-\tau_k^i-1} - \eta_i \nabla F_i(\mathbf{w}_{k-\tau_k^i-1}). \tag{4}$$

On receiving M local models, the parameter server performs the global updating by

$$\mathbf{w}_k = (1 - \sum_{v_i \in \mathcal{V}_k} \frac{D_i}{D})\mathbf{w}_{k-1} + \sum_{v_i \in \mathcal{V}_k} \frac{D_i}{D}\mathbf{x}_k^i. \tag{5}$$

Then, the parameter server sends the updated global model back to each worker in $\mathcal{V}_k$ or whose staleness is greater than the staleness threshold τ^0 (Line 15).

We illustrate the procedure of FedSA through an example in Fig. 1 with $M = 2$. For instance, after worker v_1 receives the global model $\mathbf{w}_1$, it performs local updating, derives the local model $\mathbf{w}_2^1$, and sends $\mathbf{w}_2^1$ to the parameter server in round 2. That is, $\mathbf{x}_2^1 = \mathbf{w}_2^1$ and the staleness is $\tau_2^1 = 2-2 = 0$. On the other hand, worker v_3 updates the initial model $\mathbf{w}_0$ to the local model $\mathbf{w}_1^3$, which is sent to the parameter server in round 2. Then, $\mathbf{x}_2^3 = \mathbf{w}_1^3$ and the staleness is $\tau_2^3 = 2 - 1 = 1$. On receiving two first arrival models from v_3 and v_1 in round 2, the parameter server aggregates global model $\mathbf{w}_1$, the received local models $\mathbf{x}_2^1$ (*i.e.*, $\mathbf{w}_2^1$) and $\mathbf{x}_2^3$ (*i.e.*, $\mathbf{w}_1^3$) to derive the new global model $\mathbf{w}_2$.

C. Effect of Staleness Threshold τ^0

Due to the edge heterogeneity, the parameter server may receive some local models with large staleness relative to the global model, or even encounter failure. Besides, workers may suffer from the decline of model training capability due to unpredictable accidents (*e.g.*, instability of wireless connections, random errors, resource occupation, and battery exhaustion) in practice, which aggravates the straggler problem. Too stale local models will greatly affect the training performance. To address this challenge, the parameter forces to synchronize the local models with staleness greater than a staleness threshold τ_0 [21], [22]. Specifically, the parameter server checks the staleness τ_k^i of each worker v_i in each epoch k. If τ_k^i exceeds the staleness threshold τ^0, the parameter server distributes the current global model $\mathbf{w}_k$ to v_i (Lines 14-15). In this way, too stale local models will be excluded to participate in the global updating, but it may cause additional communication resource consumption.

D. Adaptive Learning Rate

In Section II-B, the arrival frequency (*i.e.*, how often workers participate in the global updating) of different workers in FedSA may be diverse due to a host of practical reasons, such as the data size, the CPU capacities and network connections. We assume that worker v_i participates in the global updating with a relative frequency f_i, which satisfies $\sum_{v_i \in \mathcal{V}} f_i = 1$. How to measure the relative frequency will be described in Section IV-B. We deploy the adaptive learning rate η_i for each

worker v_i according to its relative frequency of participating in the global updating for better learning performance. Intuitively, if a worker frequently participates in the global updating due to its larger computation capacity or shorter communication time, the corresponding learning rate should be small [20], [30]. The adaptive learning rate η_i can be set as:

$$\eta_i = \frac{\lambda}{N \cdot f_i}, \tag{6}$$

where λ is the *global learning rate*. It is obvious that λ is equal to the weighted average learning rate of workers: $\sum_{v_i \in \mathcal{V}} f_i \eta_i = N \cdot \frac{\lambda}{N} = \lambda$. Eq. (6) represents that the learning rate of v_i is inversely proportional to its relative participation frequency f_i. As a result, given λ and f_i, the learning rate η_i of worker v_i can be determined accordingly.

E. Problem Formulation

In FedSA, different numbers (*i.e.*, M) of workers participating in the global updating of each round will lead to various performance (*e.g.*, communication resource consumption and completion time) of a single round. We denote b as the amount of transmitted data for model exchanging between a worker and the parameter server once, including model distributing and uploading. Let t_k denote the completion time of round k. We formulate our problem as follows:

$$\textbf{(P1)}: \min \sum_{k=1}^{K} t_k \tag{7a}$$

$$\textbf{s.t. } F(\mathbf{w}_K) \leq F(\mathbf{w}^*) + \varepsilon \tag{7b}$$

$$K \cdot M \cdot b \leq B \tag{7c}$$

$$M \in \{1, 2, \ldots, N\}. \tag{7d}$$

The first inequality (7b) represents that the global model will converge after K rounds, where ε is the convergence threshold representing the training accuracy. The second inequality (7c) represents that the communication consumption during K training rounds does not exceed a communication budget B. Our target is to minimize the training time, *i.e.*, $\min \sum_{k=1}^{K} t_k$.

III. Convergence Analysis

A. Assumptions

We make the following assumptions on the loss functions $F_i, \forall v_i \in \mathcal{V}$.

Assumption 1 (Smoothness): F_i is L-smooth with $L > 0$, i.e., for $\forall \mathbf{w}_1, \mathbf{w}_2$, $F_i(\mathbf{w}_2) - F_i(\mathbf{w}_1) \leq \langle \nabla F_i(\mathbf{w}_1), \mathbf{w}_2 - \mathbf{w}_1 \rangle + \frac{L}{2} \|\mathbf{w}_2 - \mathbf{w}_1\|^2$.

Assumption 2 (Strong Convexity): F_i is μ-strongly convex with $\mu \geq 0$, i.e., for $\forall \mathbf{w}_1, \mathbf{w}_2$, $F_i(\mathbf{w}_2) - F_i(\mathbf{w}_1) \geq \langle \nabla F_i(\mathbf{w}_1), \mathbf{w}_2 - \mathbf{w}_1 \rangle + \frac{\mu}{2} \|\mathbf{w}_2 - \mathbf{w}_1\|^2$.

Note that models with convex loss functions, such as linear regression and support vector machines, satisfy Assumption 2. The evaluation results in Section VI show that our mechanism can also work well for models (*e.g.*, CNN) whose loss functions are non-convex.

Assumption 3 (Global Optimal): Assume that the learning problem has at least one solution $\mathbf{w}^*$, that minimizes the global loss function $F(\mathbf{w})$, i.e., $\nabla F(\mathbf{w}^*) = \mathbf{0}$.

B. Analysis of Convergence Bound

We analyze the convergence bound of our FedSA mechanism in this section. Though the existing convergence analysis in [28] takes into account the staleness, it only focuses on one worker participating in each global updating and assumes the uniform relative frequency for all workers, which is not applicable to the edge heterogeneity scenarios. Not only do we have multiple workers participating in each global updating, but also we consider the complicated impact of workers' relative frequency on training performance.

Before convergence analysis, we first state a key lemma that is beneficial for our statement. Let $l_k^i = k - \tau_k^i - 1 \geq 0$ denote the version of the global model on v_i before round k. For any $k \geq 1$ and $v_i \in \mathcal{V}_k$, we construct three sequences of nonnegative constants, x_k, y_k^i and z_k, satisfying $\theta_k = x_k + \sum_{v_i \in \mathcal{V}_k} y_k^i < 1$. For ease of expression, we define $z_{max} = \max_k \{z_k\}$, $\tau_{max} = \max_{i,k} \{\tau_k^i\}$ and $\theta_{max} = \max_k \theta_k$. Let $S(k)$ be a sequence of real numbers for $k \geq 0$.

Lemma 1: For arbitrary $k > 0$, if

$$S(k) \leq x_k S(k-1) + \sum_{v_i \in \mathcal{V}_k} y_k^i S(l_k^i) + z_k,$$

then

$$S(k) \leq \rho^k S(0) + \delta, \tag{8}$$

where $\rho = {\theta_{max}}^{\frac{1}{1+\tau_{max}}}$ and $\delta = \frac{z_{max}}{1-\theta_{max}}$.

Proof: Since $\theta_{max} < 1$, ${\theta_{max}}^{-\frac{\tau_{max}}{1+\tau_{max}}} > 1$. It follows that

$$\begin{aligned} & x_k + \sum_{v_i \in \mathcal{V}_k} y_k^i \rho^{-\tau_{max}} \\ &= x_k + \sum_{v_i \in \mathcal{V}_k} y_k^i {\theta_{max}}^{-\frac{\tau_{max}}{1+\tau_{max}}} \\ &\leq (x_k + \sum_{v_i \in \mathcal{V}_k} y_k^i) {\theta_{max}}^{-\frac{\tau_{max}}{1+\tau_{max}}} \\ &\leq \theta_{max} \cdot {\theta_{max}}^{-\frac{\tau_{max}}{1+\tau_{max}}} = \rho. \end{aligned}$$

It is obvious that Eq. (8) is true when $k = 0$. We assume that the induction hypothesis holds for all k from 0 to $k' - 1$, *i.e.*,

$$S(k) \leq \rho^k S(0) + \delta, \quad \forall k \in \{0, 1, \ldots k' - 1\}.$$

When $k = k'$, we deduce that

$$\begin{aligned} S(k') &\leq x_{k'} S(k'-1) + \sum_{v_i \in \mathcal{V}_{k'}} y_{k'}^i S(l_{k'}^i) + z_{k'} \\ &\leq x_{k'} (\rho^{k'-1} S(0) + \delta) + \sum_{v_i \in \mathcal{V}_{k'}} y_{k'}^i (\rho^{l_{k'}^i} S(0) + \delta) + z_{k'} \\ &= x_{k'} \rho^{k'-1} S(0) + x_{k'} \delta \\ &\quad + \sum_{v_i \in \mathcal{V}_{k'}} (y_{k'}^i \rho^{k' - \tau_{k'}^i - 1} S(0) + y_{k'}^i \delta) + z_{k'} \\ &\leq (x_{k'} + \sum_{v_i \in \mathcal{V}_{k'}} y_{k'}^i \rho^{-\tau_{max}}) \rho^{k'-1} S(0) \\ &\quad + (x_{k'} + \sum_{v_i \in \mathcal{V}_{k'}} y_{k'}^i) \delta + z_{k'} \end{aligned}$$

$$\leq \rho \cdot \rho^{k'-1} S(0) + \theta_{max}\delta + z_{max}$$
$$= \rho^{k'} S(0) + \delta.$$

Thus, we complete the induction and prove the correctness of Eq. (8). ∎

Next, we derive the specific values of x_k, y_k^i and z_k in our FedSA mechanism. For ease of expression, we define $\tilde{\eta} = \max_i \eta_i$ as the maximum learning rate among workers, $\alpha = \frac{M}{N}$ as the fraction of workers participating in each global updating, and $\beta_i = \frac{D_i}{D}$ as the proportion of v_i's data size to the total amount of data. Besides, we denote $\tilde{\mathcal{V}}^M$ as the set of M workers with the smallest data size among all workers. Then $\tilde{\beta} = \min\{\sum_{v_i \in \mathcal{V}_k} \beta_i\} = \sum_{v_i \in \tilde{\mathcal{V}}^M} D_i/D$ is the proportion of the sum of the data size of workers in $\mathcal{V}^M$ to the total amount of data.

Theorem 1: If $\tilde{\eta} < \frac{\mu}{L^2}$ *and* $\lambda < \frac{1}{2\alpha\tilde{\beta}(\mu - L^2\tilde{\eta})}$*, after the initial global model is updated by Eq.* (5) *for* K *rounds, the trained model* $\mathbf{w}_K$ *satisfies*

$$\mathbb{E}[F(\mathbf{w}_K)] - F(\mathbf{w}^*) \leq \rho^K (F(\mathbf{w}_0) - F(\mathbf{w}^*)) + \delta,$$

where $\rho = [1 - 2\alpha\lambda\tilde{\beta}(\mu - \tilde{\eta}L^2)]^{\frac{1}{1+\tau_{max}}}$ *and* $\delta = \frac{\tilde{\eta}L}{2\tilde{\beta}(\mu - \tilde{\eta}L^2)} \sum_{v_i \in \mathcal{V}} \beta_i \|\nabla F_i(\mathbf{w}^*)\|^2$.

Proof: First, we analyze how the difference between $F(\mathbf{w}_k)$ and $F(\mathbf{w}^*)$ changes in each round by the global updating in Eq. (5). Since F is convex and $\sum_{v_i \in \mathcal{V}_k} \beta_i \in (0, 1]$, we deduce that

$$\begin{aligned}
&F(\mathbf{w}_k) - F(\mathbf{w}^*) \\
&= F((1 - \sum_{v_i \in \mathcal{V}_k} \beta_i)\mathbf{w}_{k-1} + \sum_{v_i \in \mathcal{V}_k} \beta_i \mathbf{x}_k^i) - F(\mathbf{w}^*) \\
&\leq (1 - \sum_{v_i \in \mathcal{V}_k} \beta_i) F(\mathbf{w}_{k-1}) + \sum_{v_i \in \mathcal{V}_k} \beta_i F(\mathbf{x}_k^i) - F(\mathbf{w}^*) \\
&= (1 - \sum_{v_i \in \mathcal{V}_k} \beta_i)(F(\mathbf{w}_{k-1}) - F(\mathbf{w}^*)) \\
&\quad + \sum_{v_i \in \mathcal{V}_k} \beta_i (F(\mathbf{x}_k^i) - F(\mathbf{w}^*)).
\end{aligned} \tag{9}$$

Let $l_k^i = k - \tau_k^i - 1$. By Eq. (4), we have $\mathbf{x}_k^i - \mathbf{w}_{l_k^i} = -\eta_i \nabla F_i(\mathbf{w}_{l_k^i})$. According to Assumption 1, it is not hard to prove that F_i is L-smooth. It follows

$$\begin{aligned}
&F(\mathbf{x}_k^i) - F(\mathbf{w}^*) \\
&\leq F(\mathbf{w}_{l_k^i}) - F(\mathbf{w}^*) + \langle \nabla F(\mathbf{w}_{l_k^i}), \mathbf{x}_k^i - \mathbf{w}_{l_k^i} \rangle \\
&\quad + \frac{L}{2} \|\mathbf{x}_k^i - \mathbf{w}_{l_k^i}\|^2 \\
&= F(\mathbf{w}_{l_k^i}) - F(\mathbf{w}^*) - \eta_i \langle \nabla F(\mathbf{w}_{l_k^i}), \nabla F_i(\mathbf{w}_{l_k^i}) \rangle \\
&\quad + \frac{\eta_i^2 L}{2} \|\nabla F_i(\mathbf{w}_{l_k^i})\|^2 \\
&\leq F(\mathbf{w}_{l_k^i}) - F(\mathbf{w}^*) - \eta_i \langle \nabla F(\mathbf{w}_{l_k^i}), \nabla F_i(\mathbf{w}_{l_k^i}) \rangle \\
&\quad + \eta_i^2 L (\|\nabla F_i(\mathbf{w}_{l_k^i}) - \nabla F_i(\mathbf{w}^*)\|^2 + \|\nabla F_i(\mathbf{w}^*)\|^2) \\
&\quad (\triangleright\ \|x + y\|^2 \leq 2(\|x\|^2 + \|y\|^2)) \\
&\leq F(\mathbf{w}_{l_k^i}) - F(\mathbf{w}^*) - \eta_i \langle \nabla F(\mathbf{w}_{l_k^i}), \nabla F_i(\mathbf{w}_{l_k^i}) \rangle \\
&\quad + \eta_i \tilde{\eta} L (\|\nabla F_i(\mathbf{w}_{l_k^i}) - \nabla F_i(\mathbf{w}^*)\|^2 + \|\nabla F_i(\mathbf{w}^*)\|^2).
\end{aligned} \tag{10}$$

According to Eq. (6), the expectation of $\sum_{v_i \in \mathcal{V}_k} \beta_i \eta_i \|\nabla F_i(\mathbf{w}^*)\|^2$ can be calculated as follows

$$\begin{aligned}
&\mathbb{E}[\sum_{v_i \in \mathcal{V}_k} \beta_i \eta_i \|\nabla F_i(\mathbf{w}^*)\|^2] \\
&= M \sum_{v_i \in \mathcal{V}} f_i \cdot \beta_i \eta_i \|\nabla F_i(\mathbf{w}^*)\|^2 \\
&= \alpha\lambda \sum_{v_i \in \mathcal{V}} \beta_i \|\nabla F_i(\mathbf{w}^*)\|^2.
\end{aligned} \tag{11}$$

According to Eq. (2) and Assumption 3, we have

$$\begin{aligned}
\mathbb{E}[\sum_{v_i \in \mathcal{V}_k} \beta_i \eta_i \nabla F_i(\mathbf{w}^*)] &= \alpha\lambda \sum_{v_i \in \mathcal{V}} \beta_i \nabla F_i(\mathbf{w}^*) \\
&= \alpha\lambda \nabla (\sum_{v_i \in \mathcal{V}} \beta_i F_i(\mathbf{w}^*)) \\
&= \alpha\lambda \nabla F(\mathbf{w}^*) = \mathbf{0}.
\end{aligned} \tag{12}$$

Similarly, it follows

$$\mathbb{E}[\sum_{v_i \in \mathcal{V}_k} \beta_i \eta_i \nabla F_i(\mathbf{w}_{l_k^i})] = \alpha\lambda \sum_{v_i \in \mathcal{V}} \beta_i \nabla F(\mathbf{w}_{l_k^i}). \tag{13}$$

By [31], the L-smoothness of function F_i can also be expressed as

$$\begin{aligned}
&\|\nabla F_i(\mathbf{w}_{l_k^i}) - \nabla F_i(\mathbf{w}^*)\|^2 \\
&\qquad \leq L \langle \nabla F_i(\mathbf{w}_{l_k^i}) - \nabla F_i(\mathbf{w}^*), \mathbf{w}_{l_k^i} - \mathbf{w}^* \rangle.
\end{aligned} \tag{14}$$

Furthermore, F is μ-strongly convex. Combining with Eqs. (12)-(14), we have

$$\begin{aligned}
&\mathbb{E}[\sum_{v_i \in \mathcal{V}_k} \beta_i \eta_i \|\nabla F_i(\mathbf{w}_{l_k^i}) - \nabla F_i(\mathbf{w}^*)\|^2] \\
&\leq \alpha\lambda L \sum_{v_i \in \mathcal{V}} \beta_i \langle \nabla F(\mathbf{w}_{l_k^i}), \mathbf{w}_{l_k^i} - \mathbf{w}^* \rangle \\
&\leq \frac{\alpha\lambda L}{\mu} \sum_{v_i \in \mathcal{V}} \beta_i \|\nabla F(\mathbf{w}_{l_k^i})\|^2.
\end{aligned} \tag{15}$$

Then combining Eqs. (9)-(13) and (15), it follows

$$\begin{aligned}
&\mathbb{E}[F(\mathbf{w}_k)] - F(\mathbf{w}^*) \\
&\leq (1 - \sum_{v_i \in \mathcal{V}_k} \beta_i)(F(\mathbf{w}_{k-1}) - F(\mathbf{w}^*)) \\
&\quad + \sum_{v_i \in \mathcal{V}_k} \beta_i (F(\mathbf{w}_{l_k^i}) - F(\mathbf{w}^*)) \\
&\quad - \alpha\lambda \sum_{v_i \in \mathcal{V}} \beta_i \|\nabla F(\mathbf{w}_{l_k^i})\|^2 + \frac{\alpha\lambda\tilde{\eta}L^2}{\mu} \sum_{v_i \in \mathcal{V}} \beta_i \|\nabla F(\mathbf{w}_{l_k^i})\|^2 \\
&\quad + \alpha\lambda\tilde{\eta}L \sum_{v_i \in \mathcal{V}} \beta_i \|\nabla F_i(\mathbf{w}^*)\|^2.
\end{aligned} \tag{16}$$

Due to the convex of F, we have

$$\begin{aligned}
\|\nabla F(\mathbf{w}_{l_k^i})\|^2 &= \|\nabla F(\mathbf{w}_{l_k^i}) - \nabla F(\mathbf{w}^*)\|^2 \\
&\geq 2\mu (F(\mathbf{w}_{l_k^i}) - F(\mathbf{w}^*)).
\end{aligned} \tag{17}$$

Thus, we can obtain that

$$\begin{aligned}
&-\alpha\lambda \sum_{v_i\in\mathcal{V}} \beta_i \|\nabla F(\mathbf{w}_{l_k^i})\|^2 + \frac{\alpha\lambda\tilde{\eta}L^2}{\mu} \sum_{v_i\in\mathcal{V}} \beta_i \|\nabla F(\mathbf{w}_{l_k^i})\|^2 \\
&\leq -\alpha\lambda(1-\frac{\tilde{\eta}L^2}{\mu}) \sum_{v_i\in\mathcal{V}_k} \beta_i \|\nabla F(\mathbf{w}_{l_k^i})\|^2 \\
&\leq -2\alpha\lambda(\mu-\tilde{\eta}L^2) \sum_{v_i\in\mathcal{V}_k} \beta_i (F(\mathbf{w}_{l_k^i}) - F(\mathbf{w}^*)). \qquad (18)
\end{aligned}$$

Then we take Eq. (18) into Eq. (16) and obtain that

$$\begin{aligned}
&\mathbb{E}[F(\mathbf{w}_k)] - F(\mathbf{w}^*) \\
&\leq (1 - \sum_{v_i\in\mathcal{V}_k} \beta_i)(F(\mathbf{w}_{k-1}) - F(\mathbf{w}^*)) \\
&\quad + \sum_{v_i\in\mathcal{V}_k} \beta_i [1 - 2\alpha\lambda(\mu-\tilde{\eta}L^2)](F(\mathbf{w}_{l_k^i}) - F(\mathbf{w}^*)) \\
&\quad + \alpha\lambda\tilde{\eta}L \sum_{v_i\in\mathcal{V}} \beta_i \|\nabla F_i(\mathbf{w}^*)\|^2. \qquad (19)
\end{aligned}$$

Let $S(k) \triangleq \mathbb{E}[F(\mathbf{w}_k)] - F(\mathbf{w}^*)$. Then $F(\mathbf{w}_{k-1}) - F(\mathbf{w}^*) = S(k-1)$ and $F(\mathbf{w}_{l_k^i}) - F(\mathbf{w}^*) = S(l_k^i)$. The recursive relation is transformed into

$$\begin{aligned}
S(k) \leq\ & \underbrace{(1 - \sum_{v_i\in\mathcal{V}_k} \beta_i)}_{x_k} S(k-1) \\
&+ \sum_{v_i\in\mathcal{V}_k} \underbrace{\beta_i [1 - 2\alpha\lambda(\mu-\tilde{\eta}L^2)]}_{y_i^k} S(l_k^i) \\
&+ \underbrace{\alpha\lambda\tilde{\eta}L \sum_{v_i\in\mathcal{V}} \beta_i \|\nabla F_i(\mathbf{w}^*)\|^2}_{z_k}. \qquad (20)
\end{aligned}$$

Then, we can obtain that

$$\begin{aligned}
\theta_{max} &= \max_k \theta_k = \max_k \{x_k + \sum_{v_i\in\mathcal{V}_k} y_k^i\} \\
&= \max_k \{1 - 2\alpha\lambda(\mu-\tilde{\eta}L^2) \sum_{v_i\in\mathcal{V}_k} \beta_i\} \\
&= 1 - 2\alpha\lambda\tilde{\beta}(\mu-\tilde{\eta}L^2). \qquad (21)
\end{aligned}$$

According to Lemma 1, if $\alpha\lambda\tilde{\beta}(\mu-\tilde{\eta}L^2) \in (0, \frac{1}{2})$, then

$$\mathbb{E}[F(\mathbf{w}_K)] - F(\mathbf{w}^*) \leq \rho^K (F(\mathbf{w}_0) - F(\mathbf{w}^*)) + \delta,$$

where $\rho = [1 - 2\alpha\lambda\tilde{\beta}(\mu - \tilde{\eta}L^2)]^{\frac{1}{1+\tau_{max}}} \in (0,1)$ is called the convergence factor, and represents the convergence rate of the loss function in one round. $\delta = \frac{\tilde{\eta}L}{2\tilde{\beta}(\mu-\tilde{\eta}L^2)} \sum_{v_i\in\mathcal{V}} \beta_i \|\nabla F_i(\mathbf{w}^*)\|^2$ called the residual error, represents that the loss function can converge to a δ-neighborhood of the optimal value. ■

C. Discussions

In this section, we can draw some meaningful corollaries from Theorem 1.

Corollary 1: The residual error δ depends partly on the data distribution. The greater the degree of data Non-IID among workers, the larger the value of $\|\nabla F_i(\mathbf{w}^*)\|^2$ for each worker v_i, and the higher residual error δ.

Corollary 2: Given IID data among workers, $\|\nabla F_i(\mathbf{w}^*)\|^2 = 0$ for $\forall v_i \in \mathcal{V}$, and $\delta = 0$. It means that the global loss function will converge to the optimal value with enough rounds.

Corollary 3: When the global learning rate λ is constant, the value of $\tilde{\eta}$ decreases as the value of M increases (because the learning rates among all workers will be more balanced). In addition, $\tilde{\beta}$ increases as the value of M increases. Therefore, the residual error δ can be effectively reduced by increasing the value of M.

Corollary 4: The upper bound of staleness τ_{max} decreases as the value of M increases. e.g., if $M = N$, then $\tau_{max} = 0$. Therefore, the convergence factor ρ decreases as M increases, too.

Corollary 4 shows that we can decrease the convergence factor ρ by increasing the number M of workers participating in each global updating. However, it does not mean a short convergence time, because the completion time of a single round depends on the arrival time of the Mth model. Consequently, a larger value of M will result in a longer completion time of a single round. Accordingly, it is a significant problem to determine the proper value of M to achieve better training performance.

IV. Algorithm Description

A. Using Convergence Bound to Convert Problem

Corollary 3 shows that the residual error δ decreases with the increase of M. So, when $M = 1$, the value of δ reaches the maximum, *i.e.*, $\delta \leq \delta_{max} = \frac{\tilde{\eta}_{M=1}L}{2\beta_{min}(\mu-\tilde{\eta}_{M=1}L^2)} \sum_{v_i\in\mathcal{V}} \beta_i \|\nabla F_i(\mathbf{w}^*)\|^2$, where $\beta_{min} = \min_{v_i\in\mathcal{V}}\{\beta_i\} = \min_{v_i\in\mathcal{V}}\{D_i\}/D$ is the proportion of the data size of worker with the smallest data size to the total amount of data, and $\tilde{\eta}_{M=1}$ is the maximum learning rate among workers when $M = 1$.

According to Theorem 1, we obtain the convergence bound of the global model after K rounds. To satisfy the constraint in Eq. (7b), we make the upper bound of $\mathbb{E}[F(\mathbf{w}_K)] - F(\mathbf{w}^*)$ less than ε, *i.e.*,

$$\begin{aligned}
&\mathbb{E}[F(\mathbf{w}_K)] - F(\mathbf{w}^*) \leq \rho^K (F(\mathbf{w}_0) - F(\mathbf{w}^*)) + \delta \\
&\leq [1 - 2\alpha\lambda\tilde{\beta}(\mu-\tilde{\eta}L^2)]^{\frac{K}{1+\tau_{max}}} (F(\mathbf{w}_0) - F(\mathbf{w}^*)) + \delta_{max} \\
&\leq \varepsilon.
\end{aligned}$$

We define that

$$\psi \triangleq 1 - 2\alpha\lambda\tilde{\beta}(\mu-\tilde{\eta}L^2) \in (0,1), \qquad (22)$$

then it holds that

$$K \geq (1+\tau_{max}) \log_\psi \phi, \qquad (23)$$

where the control parameter

$$\phi \triangleq \frac{\varepsilon - \delta_{max}}{F(\mathbf{w}_0) - F(\mathbf{w}^*)} \qquad (24)$$

represents the target training accuracy. By Eq. (7c), we can deduce that

$$K \leq \frac{B}{Mb}. \qquad (25)$$

If the value of M is chosen such that $(1+\tau_{max})\log_\psi \phi > \frac{B}{Mb}$, it means that FL system cannot reach the target training loss under the communication budget. Otherwise, when we set $K = \lceil (1+\tau_{max})\log_\psi \phi \rceil$, the value of $\sum_{k=1}^{K} t_k$ reaches the minimum. To simplify the analysis, we regard

$$K \approx (1+\tau_{max})\log_\psi \phi. \tag{26}$$

Thus the new objective can be transformed as:

$$\min \sum_{k=1}^{K} t_k = K\overline{t}, \tag{27}$$

where $\overline{t}$ is the average completion time per round. So we convert the original problem **P1** to the following optimization problem:

$$\textbf{(P2)}: \min \overline{t}(1+\tau_{max})\log_\psi \phi \tag{28a}$$

$$\textbf{s.t.}\ (1+\tau_{max})\log_\psi \phi \le \frac{B}{Mb} \tag{28b}$$

$$M \in \{1,2,\ldots,N\}. \tag{28c}$$

Remark 1: The convergence rate depends on the convergence factor ρ and the completion time of a single round, independent of the residual error δ, which represents that the loss function can converge to a δ-neighborhood of the optimal value. Therefore, it is reasonable to expand the residual error to the maximum δ_{max} for facilitating the implementation of Alg. 2.

B. Algorithm Description

The key challenge of the algorithm design is to estimate the value of some parameters in the objective of **P2**. The values of some parameters depend on the data distribution among workers, such as μ and L, and we employ the method in [32] for estimation. The values of other parameters are determined by the time sequence of model arrivals, such as $\overline{t}$, τ_{max} and $\tilde{\eta}$, and we propose an algorithm to estimate their values as formally described in Alg. 2. First, we introduce the concept of *model preparation time* to describe the edge heterogeneity. Then we design a prediction process to estimate the values of these parameters for each given value of M (*i.e.*, $M = 1, 2, \ldots N$). The optimal value of M is determined to minimize the objective at last.

1) Model Preparation Time: It is defined as the average duration from the parameter server distributes its global model to worker v_i to the next time it receives v_i's local model, *i.e.*, the duration of model distribution, local model updating and local model uploading. We assume that the full knowledge about model preparation time $\mathcal{P} = \{p_1, p_2, \ldots p_N\}$ can be obtained according to the parameter server's previous measurements. When the network environment is relatively stable, the element values in $\mathcal{P}$ are expected to be stable and can be easily estimated. When the edge environment is dynamic, the element values in $\mathcal{P}$ change over time, even so, our mechanism is still applicable by performing Alg. 2 in real-time (detail in Section V-A).

Remark 2: Since the global updating on the parameter server contains only linear operations, its duration is negligible compared with the model preparation time and can be ignored [33]. For example, our evaluation on the testbed shows that each global updating of the AlexNet model only takes 0.03s, which is significantly less than the model preparation time (15s).

Algorithm 2 Parameter Estimation

Input: The set of model preparation time $\mathcal{P}$, global learning rate λ, communication budget B, staleness threshold τ^0, each worker's data size D_i, $v_i \in \mathcal{V}$

Output: The optimal M

1: $T_{min} = +\infty$
2: **for** each $M \in \{1,2,\ldots,N\}$ **do**
3: **for** each $v_i \in V$ **do**
4: $r_i = p_i, \tau_i = 0, c_i = 0$
5: $\tau_{max} = 0$
6: **for** $k \in \{1,2,\ldots,K^*\}$ **do**
7: $t_k = r_{(M)}$
8: $\overline{t} = \frac{\sum_{0<k'<k} t_{k'}}{k}$
9: $\mathcal{V}_k = \varnothing$
10: **for** each $v_i \in \mathcal{V}$ **do**
11: **if** $r_i \le t_k$ **then**
12: $\mathcal{V}_k = \mathcal{V}_k \cup \{v_i\}$
13: **for** each $v_i \in \mathcal{V}$ **do**
14: $\tau_i = \tau_i + 1$
15: **if** $v_i \in \mathcal{V}_k$ or $\tau_i > \tau^0$ **then**
16: $r_i = p_i, \tau_i = 0, c_i = c_i + 1$
17: **else**
18: $r_i = r_i - t_k$
19: $\tau_{max} = \max\{\tau_{max}, \tau_i\}$
20: $c_{total} = \sum_{v_i \in \mathcal{V}} c_i$
21: **for** each $v_i \in \mathcal{V}$ **do**
22: $f_i = \frac{c_i}{c_{total}}$
23: $\eta_i = \frac{\lambda}{N f_i}$
24: $\tilde{\eta} = \max_{v_i \in \mathcal{V}}\{\eta_i\}$
25: $\tilde{\beta} = \sum_{v_i \in \tilde{\mathcal{V}}^M} D_i / D$
26: $K = (1+\tau_{max})\log_\psi \phi$
27: **if** $K > \frac{B}{Mb}$ **then**
28: **continue**
29: $T = K\overline{t}$
30: **if** $T < T_{min}$ **then**
31: $T_{min} = T$
32: $M' = M$
33: $M = M'$
34: **return** M

2) A Prediction Process: The values of $\overline{t}$, τ_{max} and $\tilde{\eta}$ are determined by vector $\mathcal{P}$ and the value of M. Since they are difficult to express explicitly in a mathematical formula, we design a prediction process to estimate their values (Lines 3-24). The main idea of the prediction process is to simulate the process of the model arrivals and global updating given different model preparation time $p_i \in \mathcal{P}$ for each $M = 1, 2, \ldots, N$. The prediction process will run K^* rounds.

Let r_i denote the duration from the current time to the arrival of worker v_i's model at the parameter server. Since the parameter server is aware of all workers' model preparation

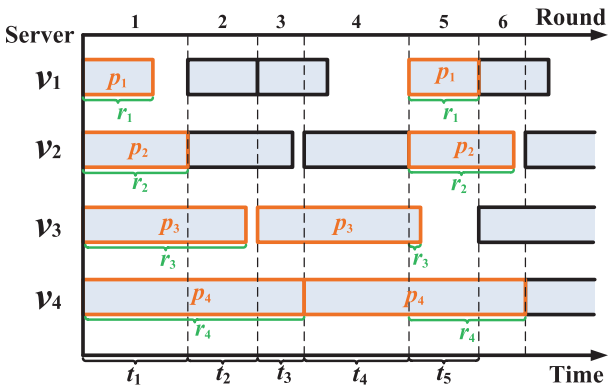


Fig. 2. Illustration of the prediction process when $M = 2$.

time $\mathcal{P} = \{p_1, p_2, \dots p_N\}$ in advance, it can also estimate the required time $\mathcal{R} = \{r_1, r_2, \dots r_N\}$ for each model to arrive and predict the arrival order of their local models. The elements in $\mathcal{R}$ are rearranged in increasing order by $r_{(1)} \leq r_{(2)} \leq \dots \leq r_{(N)}$, which represents their arrival order. The elements in $\mathcal{P}$ are also rearranged as $p_{(1)} \leq p_{(2)} \leq \dots \leq p_{(N)}$.

For each value of M, we first initialize the required arrival time r_i for each worker v_i as its model preparation time p_i. The participation frequency c_i of worker v_i, v_i's staleness τ_i and τ_{max} are initialized as 0 (Lines 3-5).

Then we run the prediction process for K^* rounds. The completion time of round k is determined by the Mth smallest number in $\mathcal{R}$, *i.e.*, $t_k = r_{(M)}$ (Line 7). Then the M models whose required arrival time is less than t_k are involved in the set $\mathcal{V}_k$ (Lines 10-12). If v_i is in set $\mathcal{V}_k$ or its staleness exceeds the threshold τ^0, its required arrival time r_i is reset to p_i, its staleness τ_i is set to 0, and its participation frequency c_i increases by 1. Otherwise, r_i is reduced by t_k. During this process, if the staleness of a model is greater than τ_{max}, it is truncated to τ_{max} (Lines 13-19). After K^* rounds, each worker v_i's participation frequency c_i is recorded and the corresponding relative participation frequency f_i is updated. Then each worker v_i's learning rate η_i is obtained by Eq. (6), and their maximum value $\tilde{\eta}$ is also derived (Lines 20-24).

To better illustrate the prediction process in our algorithm, the diagram is shown in Fig. 2, with four workers $v_1 - v_4$ and a parameter server in the system, and the workers' model preparation time satisfies $p_1 < p_2 < p_3 < p_4$. Note that r_i is a time-varying value, and we mark each r_i at the beginning of the first and fifth rounds as examples in Fig. 2. Let $M = 2$, *i.e.*, after the parameter server receives the local models from two workers, it performs global updating. At the beginning of round 1, $r_1 = p_1$, $r_2 = p_2$, $r_3 = p_3$ and $r_4 = p_4$, respectively. Since r_1 and r_2 are the two smallest among them, v_1 and v_2 participate in the global updating of round 1 and $t_1 = r_{(2)} = r_2$. Similarly, at the beginning of round 5, the expected time for workers' next arrival is $r_1 = p_1$, $r_2 = p_2$, $r_3 = p_3 - t_3 - t_4$, and $r_4 = p_4 - t_4$, respectively. It is obvious that $r_3 < r_1 < r_2 < r_4$, thus v_3 and v_1 participate in the global updating in round 5, and $t_5 = r_{(2)} = r_1$. Estimates of $\overline{t}$, τ_{max} and $\tilde{\eta}$ can be obtained after multiple rounds of such prediction.

3) Determine the Optimal Value of M: After all parameters are estimated, we calculate the number of expected rounds K for each chosen M. If the communication budget is not satisfied, the corresponding M is directly excluded (Lines 26-28). Otherwise, the objective $T = K\overline{t}$ is calculated for each M that satisfies the communication budget. Finally, The optimal M that minimizes the value of the objective is determined (Lines 29-32).

Remark 3: Alg. 2 mainly determines the optimal value of M so as to minimize the objective $T = \overline{t}(1 + \tau_{max}) \log_{\psi} \phi$, where $\psi = 1 - 2\alpha\lambda\tilde{\beta}(\mu - \tilde{\eta}L^2)$ and $\phi = (\varepsilon - \delta_{max})/(F(\mathbf{w}_0) - F(\mathbf{w}^*))$. However, the optimal parameter vector $\mathbf{w}^*$ is hard to be obtained in real scenarios. Obviously, the value of M affects the values of $\overline{t}$, τ_{max} and ψ, but does not affect the value of ϕ. Accordingly, the value of ϕ does not affect the relative size of the objective $T = \overline{t}(1 + \tau_{max}) \log_{\psi} \phi = \frac{\overline{t}(1+\tau_{max})}{\lg \psi} \lg \phi$ under different M. In fact, ϕ, also called the target training accuracy, can be regarded as a manually chosen control parameter. We can appropriately set $\phi \in (0, 1)$ in Alg. 2, which is similar to the method in [32]. For example, without loss of generality, we set ϕ as 0.05 in our experiments.

Remark 4: The choice of the rounds K^* of the prediction process is based on the following facts. 1) It is obvious that when only one worker is chosen in each round, i.e., $M = 1$, the prediction process needs the most number of rounds to obtain stable parameters. 2) Our evaluation in Section VI-B shows that the estimate of τ_{max} needs much more rounds than $\overline{t}$. 3) the maximum staleness τ_{max} corresponds to the worker with the maximum model preparation time. Therefore, we estimate the staleness of the worker with the largest model preparation time $P_{(N)}$ when $M = 1$ as $\hat{\tau} = \lceil \sum_{v_i \in \mathcal{V}} \frac{P_{(N)}}{P_{(i)}} \rceil$. Our evaluation shows that predicting $K^* \approx \hat{\tau}$ rounds can obtain stable estimates of $\overline{t}$ and τ_{max}, so it is more than enough to set $K^* = 10\hat{\tau}$.

Remark 5: As shown in our evaluation in Section VI-B, the prediction process in Alg. 2 takes a very short time, which is negligible compared with the model training and transmission time. Therefore, our algorithm is practical.

V. Extension

A. Extending to Dynamic Scenarios

In the previous sections, we assume that the network environment is relatively stable, such that the model preparation time p_i of each worker v_i is expected to be stable and a fixed number M of workers participating in the global updating is determined. However, due to the mobility of workers and uncertain network environment, their model preparation time may be varied over time. To this end, we propose the dynamic semi-asynchronous FL mechanism, in which the varied number of participating workers is adopted. The dynamic FedSA mechanism is formally described in Alg. 3.

In order to cope with the time-varying model preparation time caused by dynamic scenarios (*e.g.*, worker mobility, network uncertainty), we log each worker's historical model preparation time and determine the number of participating workers in the next round using the average of their historical model preparation time. In addition to the updating

Algorithm 3 Dynamic FedSA

1: $k = 0$
2: **for** each $v_i \in \mathcal{V}$ **do**
3: $p_i = 0, \tilde{p}_i = 0, \mathcal{Q}_i = \varnothing$
4: **while** $F(\mathbf{w}_k) - F(\mathbf{w}^*) > \varepsilon$ **do**
5: **Processing at Each Worker** v_i
6: **if** Receive $\mathbf{w}_k$ from the server **then**
7: Update local model by Eq. (3)
8: Upload local model $\mathbf{w}_{k+1}^i$
9: **Processing at the Parameter Server**
10: ***Updating Thread:***
11: $\mathcal{V}_k = \varnothing$
12: **while** $|\mathcal{V}_k| < M_k$ **do**
13: Receive local model $\mathbf{x}_k^i$ from worker v_i
14: $\mathcal{V}_k = \mathcal{V}_k \cup \{v_i\}$
15: Update global model by Eq. (5)
16: **for** each $v_i \in \mathcal{V}$ **do**
17: **if** $v_i \in \mathcal{V}_k$ or $\tau_k^i > \tau^0$ **then**
18: Distribute updated model $\mathbf{w}_k$ and learning rate η_i
19: ***Logging Thread:***
20: Measure the completion time t_k of round k
21: **for** each $v_i \in \mathcal{V}$ **do**
22: $\tilde{p}_i = \tilde{p}_i + t_k$
23: **if** $v_i \in \mathcal{V}_k$ **then**
24: $EnQueue(\mathcal{Q}_i, \tilde{p}_i), p_i = \frac{\sum_{x \in \mathcal{Q}_i} x}{|\mathcal{Q}_i|}$
25: $\tilde{p}_i = 0$
26: Use $\mathcal{P} = \{p_1, p_2, \ldots, p_N\}$ to estimate M_{k+1} by Alg. 2
27: $k = k + 1$
28: **Return** the final model and loss function

thread, the parameter server also maintains a logging thread (Lines 19-26), which uses a variable $\tilde{p}_i$ to record the duration from worker v_i last participated in the global updating to the current time. The value of $\tilde{p}_i$ increases with time (Line 22). Once worker v_i participates in the global updating, $\tilde{p}_i$ is logged and added to a queue $\mathcal{Q}_i$. The parameter server calculates the average of all elements in the queue $\mathcal{Q}_i$, *i.e.*, $p_i = \frac{\sum_{x \in \mathcal{Q}_i} x}{|\mathcal{Q}_i|}$ as the estimated model preparation time of v_i (Line 24). Then, $\tilde{p}_i$ is reset to 0 (Line 25). The parameter server uses all workers' model preparation time $\mathcal{P} = \{p_1, p_2, \ldots p_N\}$ to estimate the optimal number of participating workers for the next round by Alg. 2 (Line 26).

B. Extending to Multiple Learning Tasks Scenarios

1) Problem Formulation: In this section, we consider a more practical scenario, where multiple learning tasks are performed simultaneously in the FL system.

Assume that there are L multiple learning tasks $\mathcal{S} = \{s_1, s_2, \ldots, s_L\}$ in the FL system, and b_j is the amount of transmitted data for one time of s_j's model exchanging between a worker and the parameter server. In order to extend our algorithm to multiple learning tasks scenario, the parameter server needs to maintain a set of vectors $\mathcal{P}_j = \{p_{1,j}, p_{2,j}, \ldots, p_{N,j}\}$ for each task $s_j \in S$, where $p_{i,j}$ is the model preparation time of worker v_i for task s_j. The function of each $\mathcal{P}_j$ is similar to that of $\mathcal{P}$ in Section IV-B.

Algorithm 4 Algorithm for Multi-Task Scenario

Input: Sets of the model preparation time $\mathcal{P}_j$, $\forall s_j \in S$, communication budget B
Output: Final $M_j, \forall s_j \in S$
1: **for** each $s_j \in S$ **do**
2: Obtain initial M_j by Alg. 2
3: Estimate B_j and T_j according to M_j
4: $S_c = \varnothing$
5: **while** $\sum_{s_j \in S} B_j > B$ **do**
6: $s_j = \text{argmin}_{s_{j'} \in S \setminus S_c} \{T_{j'}\}$
7: $M_j = M_j + 1$
8: Update B_j and T_j according to M_j
9: **if** $M_j = N$ **then**
10: $S_c = S_c \cup \{s_j\}$
11: **return** $M_j, \forall s_j \in S$

Our problem is to determine the number M_j of workers participating in the global updating for each task s_j, so as to minimize the maximum training time of all tasks given communication budget. Let $F_j(\mathbf{w}_j)$ denote the global loss function of task s_j, and $t_{j,k}$ denote the completion time of task s_j in round k. We formulate our multi-task learning problem as follows:

$$\textbf{(P3)}: \min \max_{s_j \in S} \sum_{k=1}^{K_j} t_{j,k} \tag{29a}$$

$$\textbf{s.t.}\ F_j(\mathbf{w}_{j,K_j}) \leq F_j(\mathbf{w}_{j,*}) + \varepsilon \quad \forall s_j \in S \tag{29b}$$

$$\sum\nolimits_{s_j \in S} K_j M_j b_j \leq B \tag{29c}$$

$$M_j \in \{1, 2, \ldots, N\} \quad \forall s_j \in S. \tag{29d}$$

The first set of inequalities (29b) represents that the global model reaches convergence after K_j rounds for each task s_j. The second inequality (29c) represents that the communication consumption of all tasks does not exceed a communication budget.

2) Algorithm Description: Our multi-task scenario algorithm is formally described in Alg. 4.

First, we obtain the optimal M_j for each task $s_j \in \mathcal{S}$ without considering the communication budget by Alg. 2. In terms of Alg. 2, the communication consumption $B_j = K_j M_j b_j$ and the total training time $T_j = K_j \bar{t}_j$ can also be estimated, where K_j is the number of training rounds and $\bar{t}_j$ is the single round training time of s_j (Lines 1-3). Then we select the task s_j with the minimum training time, and gradually increase the value of M_j to reduce its communication consumption (along with the increase of training time) (Lines 5-8). If the value of M_j has been increased to N, s_j will never be selected again (Lines 9-10). This process continues until the whole system satisfies the communication budget.

VI. Performance Evaluation

A. System Setup

We implement an experimental testbed comprising of one parameter server and 10 workers (labeled from v_1 to v_{10}) to evaluate the effectiveness of our proposed mechanism. Fig. 3

Fig. 3. Hardware devices of our testbed. *Left*: Parameter Server; *Right*: TX2.

TABLE III
MAIN PERFORMANCE MODES OF TX2

Mode	Denver 2	Freq.	A57	Freq.	GPU Freq.
0	2	2.0 GHz	4	2.0 GHz	1.30 GHz
1	0	×	4	1.2 GHz	0.85 GHz
2	2	1.4 GHz	4	1.4 GHz	1.12 GHz
3	0	×	4	2.0 GHz	1.12 GHz

shows the hardware devices of our testbed. The parameter server is equipped with an 8-core Intel(R) Xeon(R) CPU (E5-2620v4) and 4 NVIDIA GeForce RTX 2080Ti GPUs with 11GB RAM. We use ten NVIDIA Jetson TX2 devices as workers, each of which has an NVIDIA Pascal GPU with 256 CUDA capable cores and a CPU cluster consisting of a 2-core Denver2 and a 4-core ARM CortexA57. Our experiments are based on Ubuntu 18.04, CUDA v10.0, cuDNN v7.5.0.

1) Implementation of Edge Heterogeneity: The Jetson TX2 device can be powered by different performance modes, which are listed in Table III. Specifically, modes 0 and 2 work with 2 Denver CPUs, a quad-core A57 CPU and a pascal architecture GPU, while modes 1 and 3 work with only an A57 CPU and a GPU. The frequencies of Denver CPUs are 2.0 GHz and 1.4 GHz for modes 0 and 2, respectively. The frequencies of A57 CPUs are 2.0 GHz, 1.2 GHz, 1.4 GHz and 2.0 GHz for modes 0, 1, 2 and 3, respectively. The frequencies of GPUs are 1.30 GHz, 0.85 GHz, 1.12 GHz and 1.12 GHz for modes 0, 1, 2 and 3, respectively. To realize the device heterogeneity, we set the TX2 devices in different modes, at different distances from the parameter server, so that their model preparation time is different.

2) Models and Datasets: The experiments are conducted with three classical models (*i.e.*, LR [34], CNN [35] and AlexNet [36]) and on two datasets (MNIST [37] and CIFAR-10 [38]). LR is trained on MNIST, consisting of 60,000 handwritten digits for training and 10,000 for the test. AlexNet is trained on CIFAR-10, which includes 50,000 images for training and 10,000 for the test, and has ten different types of objects. CNN is trained on both MNIST and CIFAR-10. The detailed CNN network architectures. 1) For MNIST: The CNN consists of two 5×5 convolution layers (20, 50 channels), each of which is followed by 2×2 max pooling, two fully-connected layers with 800 and 500 units, and a softmax layer with 10 units. 2) For CIFAR-10: The CNN consists of two 5×5 convolution layers (32, 64 channels), each of which is also followed by 2×2 max pooling, two fully-connected layers with 1600 and 512 units, and a softmax layer with 10 units. We adopt the same mini-batch size (*i.e.*, 64) for all workers.

For our FedSA mechanism, the global learning rate is set as $\lambda = 0.01$ for CNN and LR on MNIST, and $\lambda = 0.05$ for CNN and AlexNet on CIFAR-10. Then the learning rate η_i of each v_i is set according to Eq. (6). The weighted average of the learning rates of all workers satisfies $\overline{\eta} = \sum_{v_i \in V} f_i \eta_i = \lambda$. In addition, the learning rates of all workers in all benchmarks are set to $\overline{\eta}$.

3) Data Partition: To analyze training performance under different data distributions, we study three ways of partitioning the MNIST and CIFAR-10 data among workers: 1) IID. All data in MNIST (or CIFAR-10) are shuffled, and then evenly partitioned into 10 subsets, one for each worker. 2) Non-IID. All data in the dataset are divided into two groups according to the parity of the labels. Then the data in the odd group are distributed evenly to workers v_1-v_5, and the data in the even group are distributed evenly to workers v_6-v_{10}. 3) Non-IID-0.9. 10% of the data in the dataset are distributed to 10 workers as in IID, and the other 90% of the data are distributed to workers as in Non-IID.

4) Benchmarks and Performance Metrics: We adopt four typical mechanisms as benchmarks for performance comparison. Two of them belong to the synchronous FL mechanisms and are variants of FedAvg [9]. The first is called FedAvg-full (FedAF), in which all workers participate in the global updating in each round. The second is called FedAvg-partial (FedAP), in which the parameter server selects partial workers randomly to participate in the global updating in each round. The third one, called FedAsync (FedASY) [19], is an asynchronous FL mechanism, where the parameter server performs the global updating as soon as it receives a local model from any worker. The fourth one is a semi-asynchronous FL mechanism, called SAFA [22]. Since FedAP and SAFA do not specify the number of workers participating in the global

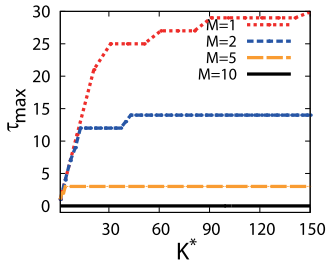


Fig. 4. Value of τ_{max} vs. Value of K^*.

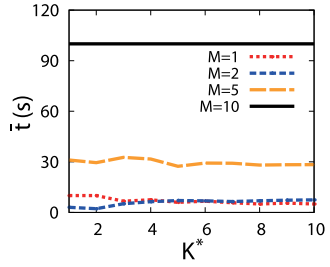


Fig. 5. Value of $\overline{t}$ vs. Value of K^*.

TABLE IV
THE VALUE OF $\hat{\tau}$ IN RELATION TO γ AND N

$\hat{\tau}$ N / γ	10	20	50	100
10	28	53	130	258
50	72	108	224	422
100	124	162	291	518

updating in each round, we naturally set it to half of the total number (*i.e.*, 5) of workers in our experiments.

To evaluate the training performance, we adopt three performance metrics. 1) *Loss Function* reflects the training process of the model and whether convergence has been achieved. 2) *Accuracy* is the most common performance metric in classification problems, which is defined as the proportion of right data classified by the model to all test data. 3) *Training Time* is adopted to measure the training speed.

B. Parameter Estimation

1) Prediction Process: To know the specific prediction rounds K^* to obtain stable parameters (*e.g.*, τ_{max} and $\overline{t}$) in Alg. 2, we simulate the different numbers of workers N with different model preparation time $\mathcal{P}$ in the system. Let $\gamma = \frac{p_{(N)}}{p_{(1)}}$ be the quotient of the largest and the smallest elements in $\mathcal{P}$. All the elements in $\mathcal{P}$ are evenly distributed between $p_{(1)}$ and $p_{(N)}$. As shown in Figs. 4 and 5, the values of τ_{max} and $\overline{t}$ tend to be stable after a certain rounds, and predicting τ_{max} takes more number of rounds than predicting $\overline{t}$. Besides, The smaller the value of M is, the more rounds it needs.

The value of $\hat{\tau}$ is calculated by $\hat{\tau} = \lceil \sum_{v_i \in \mathcal{V}} \frac{P_{(N)}}{P_{(i)}} \rceil$ according to Section IV-B. We calculate the value of $\hat{\tau}$ under different γ and N as shown in Table IV. The test results are shown in Figs. 6 ($\gamma = 10$) and 7 ($N = 10$). It indicates that the actual test result of appropriate prediction rounds K^* is

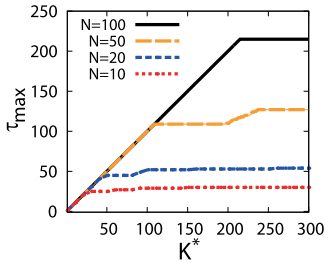


Fig. 6. Value of τ_{max} vs. Value of K^* when $\gamma = 10$.

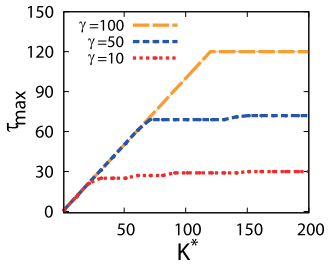


Fig. 7. Value of τ_{max} vs. Value of K^* when $N = 10$.

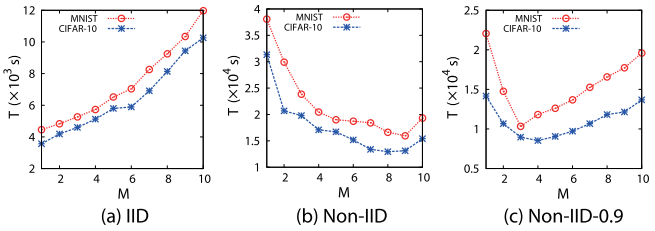


Fig. 8. Estimated completion time T vs. M. (a) IID; (b) Non-IID; (c) Non-IID-0.9.

approximately equal to the calculated $\hat{\tau}$, *i.e.*, $K^* \approx \hat{\tau}$. For example, when $N = 10$, and $\gamma = 50$ in Fig. 7, the value of τ_{max} stays at 69 after the 70th round, while our calculated $\hat{\tau}$ is 72 in Table IV. Although the value of τ_{max} may increase slightly in a few rounds after the $\hat{\tau}$th round, setting $K^* = 10\hat{\tau}$ is more than enough in Alg. 2.

2) Estimation of M Under Different Data Distributions: In this section, we estimate the optimal value of M under different data distributions in FedSA.

We calculate the value of our objective $T = \overline{t}(1 + \tau_{max}) \log_{\psi} \phi$ by Alg. 2 when the value of M is set as 1 to N. Without loss of generality, we set $\phi = 0.05$ in our experiments. The results are shown in Fig. 8. Fig. 8(a) shows that, under IID data distribution, it takes the shortest time to achieve target training performance when $M = 1$, and more time is required with the larger value of M. Fig. 8(b) shows that when the data distribution is Non-IID, the optimal value of M is predicted as 7-9. Fig. 8(c) shows that when the data distribution is Non-IID-0.9, the optimal value of M is predicted as 3-4.

Besides, the total execution time of the prediction process and the estimation of M is 0.11s when $N = 10$, and 13.66s when $N = 100$, which is negligible compared with the massive amounts of model training time.

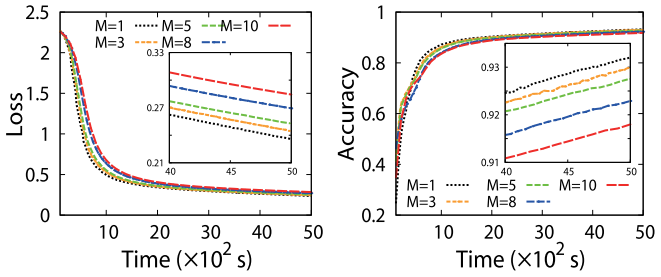


Fig. 9. The influence of M (LR on MNIST, IID). *Left*: Loss; *Right*: Accuracy.

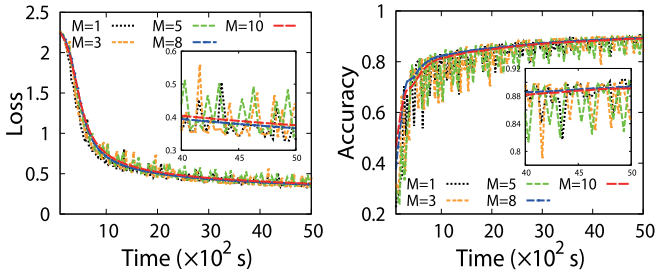


Fig. 10. The influence of M (LR on MNIST, Non-IID). *Left*: Loss; *Right*: Accuracy.

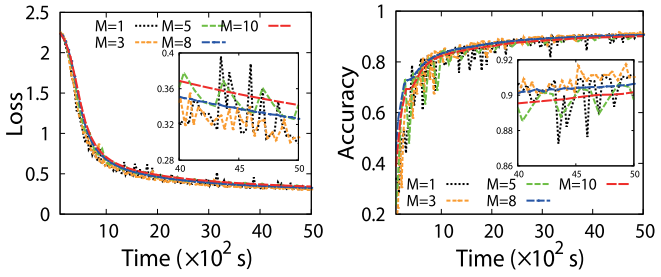


Fig. 11. The influence of M (LR on MNIST, Non-IID-0.9). *Left*: Loss; *Right*: Accuracy.

3) Verification for the Optimal M: To verify the effectiveness of the previous parameter estimation, we conduct experiments on the testbed to explore the influence of M on model training performance under different data distributions.

We train LR on MNIST for 5000s under different data distributions as shown in Figs. 9-11. Fig. 9 shows that when the data distribution is IID, it takes more time to achieve convergence with the larger value of M. For example, when M is set as 1, 3, 5, 8 and 10, after 5000s of training, the accuracy is 93.13%, 92.98%, 92.75%, 92.32% and 91.18%, respectively. The time required to reach 91% accuracy is 2489s, 2680s, 2862s, 3381s and 3903s, respectively. The situation is complicated when the data distribution is Non-IID. As shown in Fig. 10, from the overall trend, loss decreases and accuracy increases with the increase of training time. However, when M is set as 1, 3 and 5, loss and accuracy curves jitter in different degrees. For example, the accuracy jitters between 82% and 90% for $M = 1$, 3 and 5 after 5000s of training. On the other hand, when M is set as 8 and 10, the accuracy is stable over 89.5% and 89.1%, respectively. When M is set as 1, 3, 5, 8 and 10, the time required to stabilize above 80% accuracy is 4961s, 4908s, 3272s, 740s and 905s, respectively. The results

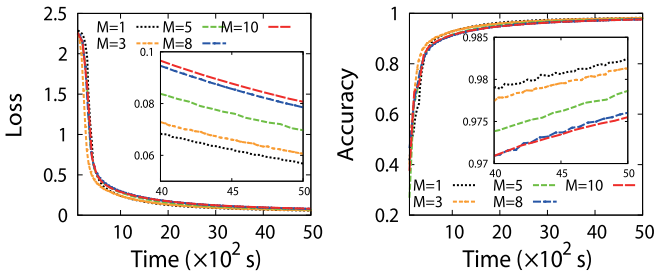


Fig. 12. The influence of M (IID). *Left*: Loss; *Right*: Accuracy.

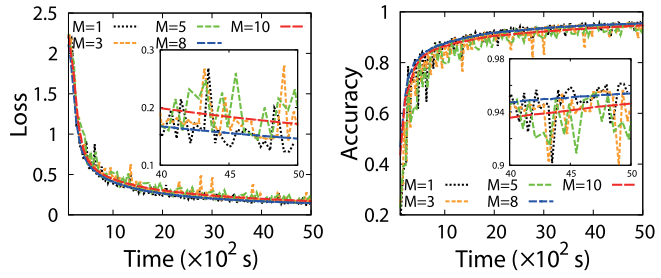


Fig. 13. The influence of M (Non-IID). *Left*: Loss; *Right*: Accuracy.

for Non-IID-0.9 are shown in Fig. 11, when M is set as 1, 3, 5, 8 and 10, the time required to stabilize above 88% accuracy is 4598s, 2489s, 2625s, 2739s and 2896s, respectively.

Figs. 9-11 show that the experimental results of training LR can match with the theoretical convergence bound when the loss function is convex, and the determination of the optimal M is effective in Alg. 2. To verify the effectiveness of Alg. 2 when the loss function is non-convex, we further training CNN on MNIST under different data distributions. Figs. 12-14 show that Alg. 2 can still determine the optimal M when the loss function does not satisfy the convex assumption. For example, as shown in Fig. 13, when the data distribution is Non-IID, the accuracy jitters between 90% and 96% for $M = 1$, 3 and 5 after 5000s of training. On the other hand, when M is set as 8 and 10, the accuracy is stable over 95.39% and 94.63%, respectively. When M is set as 1, 3, 5, 8 and 10, the time required to stabilize above 90% accuracy is 4961s, 4908s, 3272s, 1524s and 1897s, respectively.

Figs. 9-14 show that the jitter can be reduced (*i.e.*, the residual error δ in Section III is decreased) with the increasing value of M. At the same time, the convergence rate is different given different M under different data distribution, which is jointly determined by the convergence factor ρ and the completion time of a single round. In fact, both the convergence rate and jitter degree together determine the training performance. The evaluation results in Fig.4-14 show that our algorithm can estimate the optimal value of M to minimize the training time while achieving target training performance on the datasets with different degrees of Non-IID.

C. Evaluation Results

1) Training Performance Comparison: In this section, we compare the benchmarks with our FedSA mechanism by training the models for 5000s on different datasets.

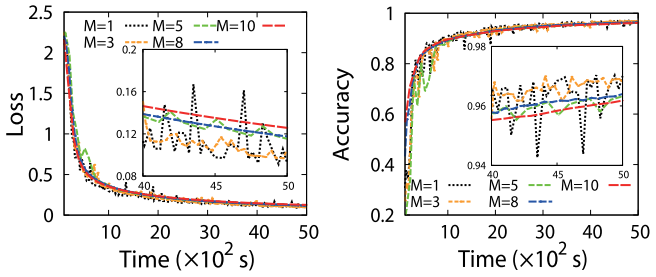


Fig. 14. The influence of M (Non-IID-0.9). *Left*: Loss; *Right*: Accuracy.

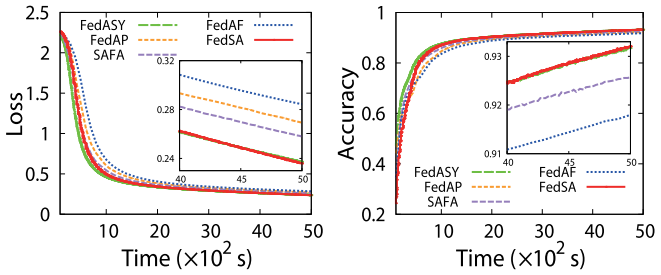


Fig. 15. Loss/Accuracy vs. Time (LR on MNIST, IID). *Left*: Loss; *Right*: Accuracy.

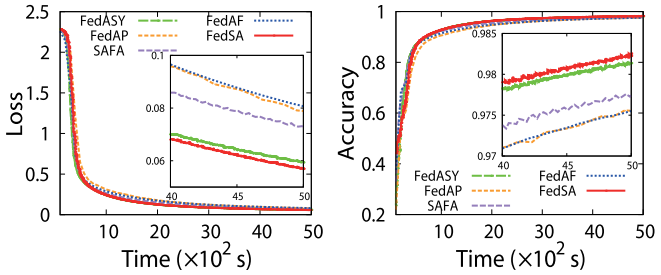


Fig. 16. Loss/Accuracy vs. Time (CNN on MNIST, IID). *Left*: Loss; *Right*: Accuracy.

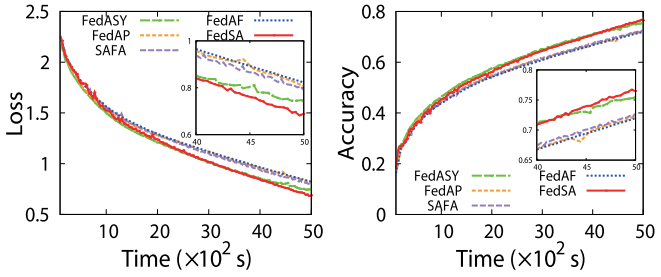


Fig. 17. Loss/Accuracy vs. Time (CNN on CIFAR-10, IID). *Left*: Loss; *Right*: Accuracy.

We observe that FedSA always achieves better training performance than FedAF, FedAP, FedASY and SAFA, but the degree of performance varies with different datasets and data distributions.

Figs. 15-18 show that, when the data distribution is IID, the training performance of FedSA is a little worse than that of FedASY at the beginning, but gradually catches up with and exceeds that of FedASY as time goes on because of the adaptive learning rate (Section VI-C.5). Besides, the training

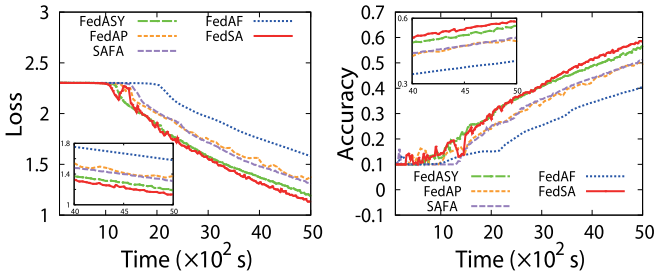


Fig. 18. Loss/Accuracy vs. Time (AlexNet on CIFAR-10, IID). *Left*: Loss; *Right*: Accuracy.

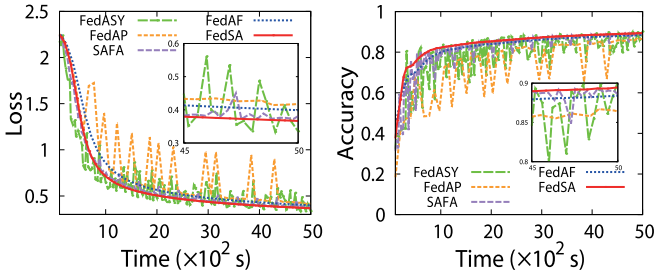


Fig. 19. Loss/Accuracy vs. Time (LR on MNIST, Non-IID). *Left*: Loss; *Right*: Accuracy.

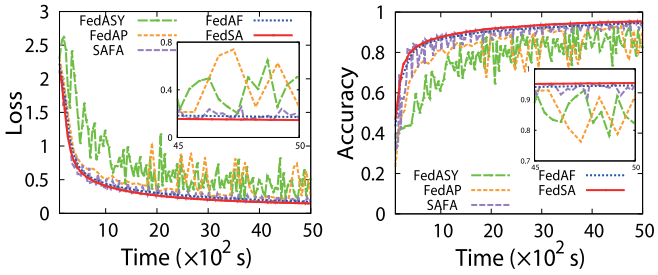


Fig. 20. Loss/Accuracy vs. Time (CNN on MNIST, Non-IID). *Left*: Loss; *Right*: Accuracy.

performance of FedSA can work better than the other three mechanisms. For example, as shown in Fig. 16, after 5000s of training, the accuracy of FedSA, FedASY, SAFA, FedAF and FedAP is 98.25%, 98.13%, 97.71%, 97.55% and 97.48%, respectively. The time required to reach 97% accuracy is 1840s, 2612s, 3359s, 3894s and 3842s, respectively. FedSA reduces training time by about 29.56%, 45.22%, 52.75% and 52.11% compared to FedASY, SAFA, FedAF and FedAP.

In addition, when the data distribution is Non-IID, the performance of different mechanisms is very complicated. Figs. 19-22 show that, FedSA and FedAF jitter a little. The training performance of FedAF and SAFA is worse than that of FedSA, but the degree is diverse across different training models on different datasets. The performance of FedAP and FedASY is not satisfying that they cannot even converge within the limited training time. For example, as shown in Fig. 20, after 5000s of training, the accuracy of FedSA, FedAF, SAFA, FedASY, and FedAP is 95.39%, 94.64%, 91.31%, 76.61% and 60.62%, respectively. The time required to reach 90% accuracy is 1520s, 1912s and 3950s for FedSA,

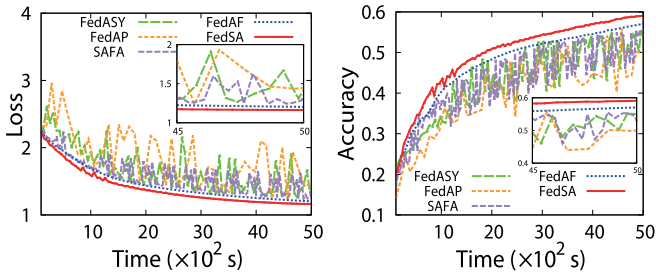


Fig. 21. Loss/Accuracy vs. Time (CNN on CIFAR-10, Non-IID). *Left*: Loss; *Right*: Accuracy.

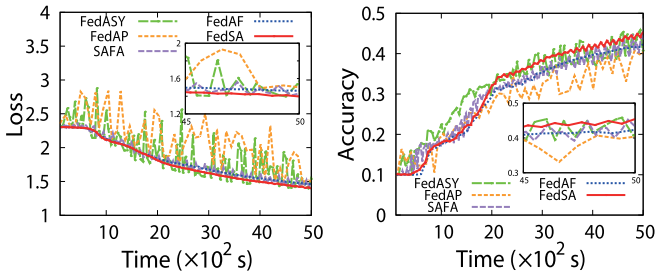


Fig. 22. Loss/Accuracy vs. Time (AlexNet on CIFAR-10, Non-IID). *Left*: Loss; *Right*: Accuracy.

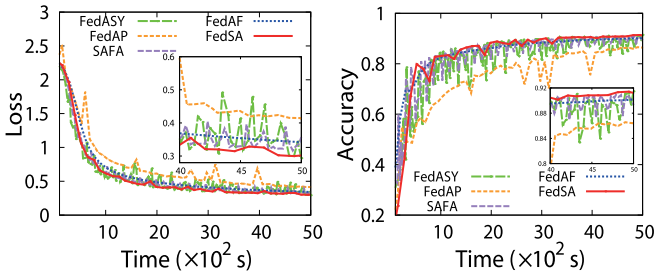


Fig. 23. Loss/Accuracy vs. Time (LR on MNIST, Non-IID-0.9). *Left*: Loss; *Right*: Accuracy.

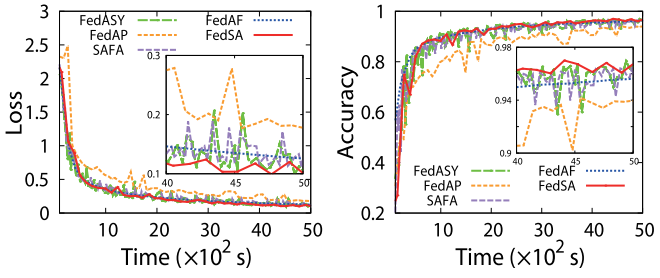


Fig. 24. Loss/Accuracy vs. Time (CNN on MNIST, Non-IID-0.9). *Left*: Loss; *Right*: Accuracy.

FedAF and SAFA, respectively. FedSA reduces training time by about 20.50% and 61.52% compared to FedAF and SAFA, respectively. However, FedASY and FedAP fail to converge given the time constraint. It indicates that FedASY and FedAP are more sensitive to data distribution. For LR on MNIST, CNN on CIFAR-10 and AlexNet on CIFAR-10, FedSA reduces the training time by about 10.71%, 18.92% and 21.33% compared to FedAF as shown in Figs. 19, 21 and 22.

When the data distribution is Non-IID-0.9, all mechanisms jitter smaller compared to the case of Non-IID. The reason lies in that as long as workers have a small amount of data (relative to the total amount of data) uniformly sampled from the overall distribution, the performance of model training on Non-IID data can be greatly improved [10]. However, the performance of FedAP and FedASY is still not satisfying. For example, as shown in Fig. 24, after 5000s of training, the accuracy of FedSA, FedAF, SAFA, FedASY, and FedAP is 96.49%, 96.17%, 94.21%, 92.51% and 71.42%, respectively. The time required to reach 92% accuracy is 1389s, 1650s, 2429s and 4798s for FedSA, FedAF, SAFA and FedASY, respectively. FedSA reduces training time by about 15.81%, 42.81% and 71.05% compared to FedAF, SAFA and FedASY.

More detailed training performance of CNN on MNIST is listed in Table V. When the data distribution is IID, our algorithm can work better than FedAF, FedAP and SAFA. Besides, the performance gap between FedSA and FedASY is widening as the target accuracy gets higher and higher. For example, to reach 98% accuracy, FedSA reduces training time by about 16.41%, 42.84%, 42.03% and 35.87% compared to FedASY, FedAF, FedAP and SAFA, respectively. When the data distribution is Non-IID, FedASY and FedAP cannot converge to over 90% accuracy. FedSA reduces training time

TABLE V
TRAINING TIME REQUIRED TO ACHIEVE TARGET ACCURACY

Accuracy		FedSA	FedASY	FedAF	FedAP	SAFA
IID	70%	320s	270s	298s	279s	258s
	80%	339s	338s	388s	359s	369s
	90%	672s	697s	811s	946s	720s
	95%	1489s	1486s	2102s	2140s	1763s
	97%	1840s	2612s	3894s	3842s	3355s
	98%	3780s	4522s	6613s	6521s	5894s
Non -IID	70%	261s	2073s	302s	3517s	454s
	80%	488s	9483s	552s	13495s	880s
	90%	1522s	-	1991s	-	3624s
	95%	4682s	-	5703s	-	13541s
	97%	11457s	-	14123s	-	-
	98%	-	-	-	-	-
Non -IID -0.9	70%	241s	479s	220s	450s	362s
	80%	627s	1417s	680s	3424s	463s
	90%	1010s	2559s	1112s	8710s	1738s
	95%	2829s	6412s	3389s	13412s	5192s
	97%	5439s	12883s	7512s	-	10293s
	98%	14312s	-	15110s	-	-

by 15-30% compared to FedAF and 40%-65% compared to SAFA. When the data distribution is Non-IID-0.9, our algorithm can work better than the benchmarks. For example, to reach 95% accuracy, FedSA reduces training time by about 55.88%, 16.52%, 78.91% and 45.51% compared to FedASY, FedAF, FedAP and SAFA, respectively.

2) The Impact of Communication Budgets: Table VI shows the impact of adjusting the communication budget B for CNN on MNIST. The amount of the CNN model data transmitted each time between a worker and the parameter server is 1.64MB. When the data distribution is IID, we set the target

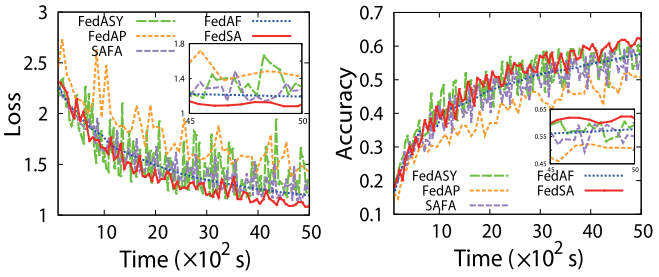


Fig. 25. Loss/Accuracy vs. Time (CNN on CIFAR-10, Non-IID-0.9). *Left*: Loss; *Right*: Accuracy.

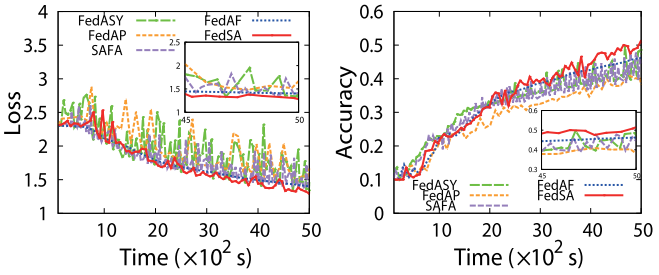


Fig. 26. Loss/Accuracy vs. Time (AlexNet on CIFAR-10, Non-IID-0.9). *Left*: Loss; *Right*: Accuracy.

TABLE VI
THE TRAINING TIME UNDER DIFFERENT COMMUNICATION BUDGETS

Communication Budgets(MB)		FedSA M	FedSA T	FedAF	SAFA	FedAP	Fed -ASY
IID (95%)	800	1	1489s	2088s	1759s	2140s	1499s
	600	4	1549s				
	400	9	2070s		–	–	–
	200	–					
Non -IID (90%)	1200	8	4342s	5720s	×		
	1000	9	5171s				
	800	10	5720s				
	600	–					

–: Can not reach the required accuracy. ×: Can not converge.

training accuracy as 95% and observe the training time. When the data distribution is Non-IID, we set the target training accuracy as 90%. As shown in Table VI, when the communication budget is large enough, the optimal M for FedSA is set as 1 in the case of IID data. The value of optimal M increases with the decrease of the communication budget. This is because that, the smaller M is, the more communication resource it requires to achieve the same accuracy. When the communication budget is small, a small value of M makes the constraint Eq. (7c) not satisfied. In the case of Non-IID, the optimal M is first estimated as described in VI-B.2. When the choice does not satisfy the communication budget, the value of M will also increase as in the case of IID data.

3) Influence of Adaptive Learning Rate: In this section, we verify the benefits of the adaptive learning rate described in Section II-D. Figs. 27 and 28 show the comparison of training performance between FedSA ($M = 1$) (with adaptive learning rate) and FedASY (without adaptive learning rate) with CNN on MNIST. Fig. 27 shows that when the data distribution is IID, the performance of FedSA is a little worse than FedASY

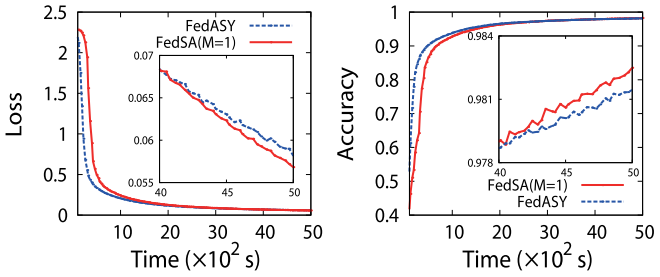


Fig. 27. Influence of adaptive learning rate (IID). *Left*: Loss; *Right*: Accuracy.

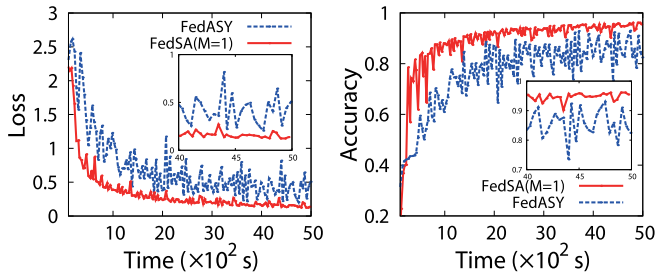


Fig. 28. Influence of adaptive learning rate (Non-IID). *Left*: Loss; *Right*: Accuracy.

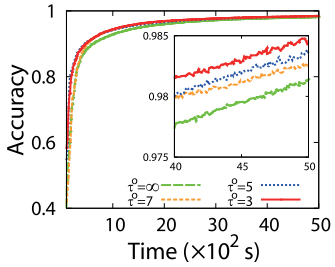


Fig. 29. Influence of staleness threshold (IID).

at the beginning, but after 3000s of training, its accuracy catches up with or even exceeds that of FedASY. For example, after 5000s of training, the accuracy of FedSA and FedASY is 98.25% and 98.14%, respectively. Fig. 28 shows that by using the adaptive learning rate, FedSA can greatly reduce the jitter of the training curve and improve training performance when the data distribution is Non-IID. For example, after 5000s of training, the accuracy of FedSA varies from 92%-96%, while the accuracy of FedASY varies from 75%-90%.

4) Influence of Staleness Threshold: In this section, we observe the effect of the staleness threshold τ^0 on training performance and communication consumption. Figs. 29-31 show the comparison of training accuracy of CNN on MNIST when staleness threshold τ^0 is set to 3, 5, 7 and infinity (*i.e.*, no forced synchronization of stale models), respectively. As shown in Fig. 29, when the data is IID, the accuracy increases with the decreasing of staleness threshold τ^0 under the same training time. For example, after 5000s of training, the accuracy will reach 98.41%, 98.35%, 98.25% and 98.14% when τ^0 is set to 3, 5, 7 and infinity, respectively. However, τ^0 will not significantly impact the accuracy when the data distribution is Non-IID or Non-IID-0.9. For example, the accuracy

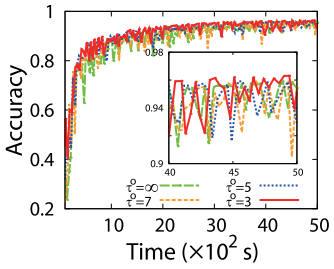


Fig. 30. Influence of staleness threshold (Non-IID).

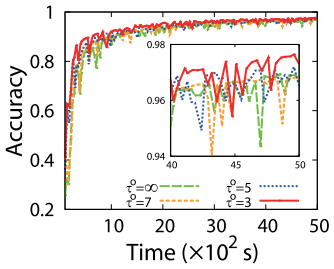


Fig. 31. Influence of staleness threshold (Non-IID-0.9).

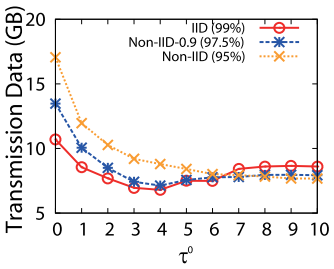


Fig. 32. Transmission Data vs. Staleness Threshold τ^0.

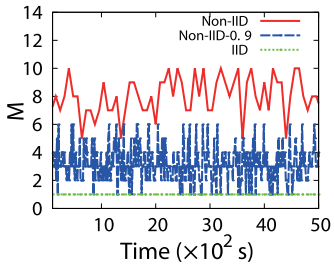


Fig. 33. M vs. Time.

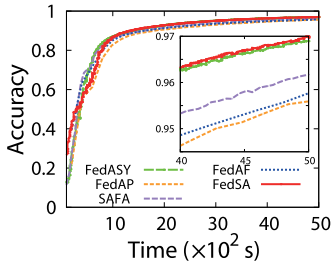


Fig. 34. Accuracy in dynamic scenario (IID).

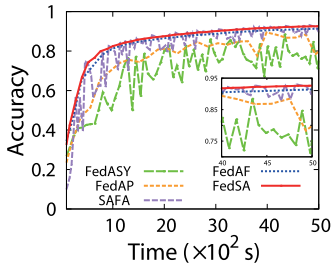


Fig. 35. Accuracy in dynamic scenario (Non-IID).

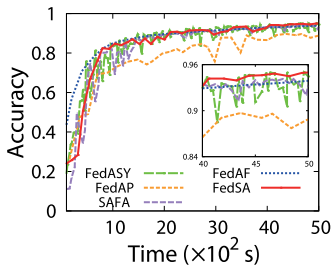


Fig. 36. Accuracy in dynamic scenario (Non-IID-0.9).

jitters between 90% and 95% for all values of τ^0 in Fig. 30. That is because forcing slow workers to synchronize may cause the global model more biased towards the local models that participate in the global updating more frequently, which offsets the beneficial effect of τ^0 on bounding the staleness. Fig. 32 shows the amount of transmission data when the value of τ^0 ranges from 0 to 10 under different data distributions. As shown in this figure, when the data distribution is Non-IID, the amount of transmission data increases as the value of τ^0 decreases. For example, when $\tau^0 = 10$, it requires transmission data of 7.67GB to reach 95% accuracy, while τ^0 is 0, it requires 17.06GB. When the data distribution is IID, it minimizes the amount of transmission data with the target accuracy requirement by setting τ^0 to 4. For example, when $\tau^0 = 4$, it consumes transmission data of 6.79GB to reach 99% accuracy, while 10.70GB when $\tau^0 = 0$ and 8.61GB when $\tau^0 = 10$.

5) Dynamic Scenario: In this section, we train CNN on MNIST to verify that our FedSA can still work well in the dynamic scenario by dynamically determining the optimal value of M. In order to realize the dynamic scenario on our testbed, we take the trained local model on worker v_i into the cache for a period of time κp_i before it is uploaded to the parameter server, where κ is a random value uniformly sampled from [0,2], and p_i is the current model preparation time of v_i.

As shown in Fig. 33, the optimal value of M is changed under different data distributions. For example, the optimal value of M is 1 under IID data distribution. While under the Non-IID data distribution, the optimal value of M fluctuates between 5 and 10. When the data distribution is Non-IID-0.9, the optimal value of M fluctuates between 1 and 6. Furthermore, we compare FedSA with other benchmarks in the dynamic scenario in Figs. 34-36. As shown in these figures, the training performance of FedSA is better than the other mechanisms under different distributions.

6) Multi-Learning Tasks Scenario: In this section, we evaluate the performance of three learning tasks simultaneously existing in the system with total communication budgets,

TABLE VII
THE MAXIMUM TRAINING TIME FOR MULTI-TASK SCENARIO UNDER DIFFERENT COMMUNICATION BUDGETS

Communication Budgets (MB)		FedSA							FedAF	SAFA	FedAP	FedASY
		M_1	T_1	M_2	T_2	M_3	T_3	$\max T_j$				
IID	5000	1	3000s	1	3000s	1	3000s	3000s	4910s	4722s	3458s	3210s
	4500	3	3183s	2	3152s	3	3281s	3281s				–
	4000	6	3526s	4	3332s	8	3684s	3684s				
	3500	8	3985s	6	3912s	10	3730s	3985s		–	–	
	3000	10	4650s	9	4703s	10	3730s	4703s				
	2500	–										
Non-IID	2500	8	3000s	8	3000s	8	3000s	3000s	4200s	×		
	2200	9	3255s	8	3000s	9	3620s	3620s				
	2000	10	3295s	10	3216s	9	3620s	3620s				
	1800	–										

–: Can not reach the required accuracy. ×: Can not converge.

including LR on MNIST (s_1, 2.56 MB), CNN on MNIST (s_2, 1.64MB) and CNN on CIFAR-10 (s_3, 3.35MB). Since the training time for different models to achieve the same accuracy is diverse on different datasets, for the sake of comparison, we refer to the results in section VI-C.1, and take the accuracy after 3000s of training as the target training accuracy of the three tasks in this section, *i.e.*, when the data distribution is IID, the target training accuracy of three tasks s_1, s_2 and s_3 is set as 91.55%, 97.42% and 64.65%, respectively. When the data distribution is Non-IID, their target training accuracy is set as 87.42%, 93.09% and 54.46%, respectively.

The results are shown in Table VII. When the data distribution is IID, as the communication budget drops from 5000MB to 2500MB, the M_j chosen by each task s_j gradually increases from 1 to 10 in FedSA, and the training time also increases gradually. When the communication budget is large enough, FedSA requires a little less training time than FedASY, but much less time than the two FedAvg mechanisms. For example, when the communication budget is 5000 MB, the training time of FedSA, FedASY, SAFA, FedAF and FedAP is 3000s, 3210s, 3458s, 4910s and 4722s, respectively. When the communication budget gets smaller, FedASY, SAFA and FedAP cannot reach the target accuracy, the training time of FedSA gradually increases to close to FedAF.

When the data distribution is Non-IID, FedASY, SAFA and FedAP cannot even converge. When the communication budget is large enough, FedSA requires much less training time than FedAF does. As the communication budget drops, the M_j chosen by each task s_j increases from 8 to 10 in FedSA.

7) Summary of Evaluation Results: We conclude the evaluation results from Figs. 15-36 and Tables V-VII as follows. Figs. 15-26 and Tables V show that our FedSA mechanism is more efficient compared with benchmarks under different data distributions. Table VI shows that FedSA can estimate the optimal M given different communication budgets. Figs. 27-28 show that our proposed adaptive learning rate can improve the performance on both IID and Non-IID data. Figs. 29-32 show the effect of staleness threshold on training performance and communication consumption. Figs. 33-36 show the effectiveness of the dynamic FedSA mechanism. Table VII shows that our mechanism and algorithms are practical in the multiple learning tasks scenario.

VII. CONCLUSION

In this paper, we propose FedSA, a novel semi-asynchronous federated learning mechanism for heterogeneous edge computing. We theoretically prove the convergence of FedSA. We then propose an efficient algorithm to determine the optimal number of participating workers in each round given communication budget to minimize the training time. We further extend our algorithm to the dynamic and multiple learning tasks scenarios in practice. The experimental results indicate the effectiveness of our proposed mechanism on the datasets with different degrees of Non-IID.

REFERENCES

[1] A. Pantelopoulos and N. G. Bourbakis, "A survey on wearable sensor-based systems for health monitoring and prognosis," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 40, no. 1, pp. 1–12, Jan. 2010.

[2] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones," in *Proc. Esann*, Apr. 2013, p. 3.

[3] G. Zhu, D. Liu, Y. Du, C. You, J. Zhang, and K. Huang, "Towards an intelligent edge: Wireless communication meets machine learning," *IEEE Commun. Mag.*, vol. 58, no. 1, pp. 19–25, Jan. 2020.

[4] H. B. McMahan and D. Ramage. (2017). *Google*. [Online]. Available: http://www.googblogs.com/federated-learning-collaborative-machine-learning-without-centralized-training-data/

[5] X. Wei, Q. Li, Y. Liu, H. Yu, T. Chen, and Q. Yang, "Multi-agent visualization for explaining federated learning," in *Proc. 28th Int. Joint Conf. Artif. Intell.* Palo Alto, CA, USA: AAAI Press, Aug. 2019, pp. 6572–6574.

[6] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7.

[7] M. Li, "Scaling distributed machine learning with the parameter server," in *Proc. 11th USENIX Symp. Oper. Syst. Design Implement. (OSDI)*, 2014, pp. 583–598.

[8] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," 2016, *arXiv:1610.02527*. [Online]. Available: https://arxiv.org/abs/1610.02527

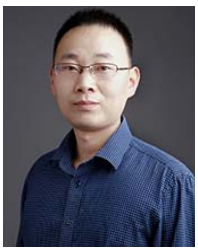
[9] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. AISTATS*, 2017, pp. 1273–1282.

[10] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," 2018, *arXiv:1806.00582*. [Online]. Available: https://arxiv.org/pdf/1806.00582

[11] F. Sattler, S. Wiedemann, K.-R. Muller, and W. Samek, "Robust and communication-efficient federated learning from non-I.I.D. data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3400–3413, Sep. 2020.
[12] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 4615–4625.
[13] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.
[14] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.
[15] M. Li, D. G. Andersen, A. J. Smola, and K. Yu, "Communication efficient distributed machine learning with the parameter server," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 19–27.
[16] N. Wang, B. Varghese, M. Matthaiou, and D. S. Nikolopoulos, "ENORM: A framework for edge node resource management," *IEEE Trans. Services Comput.*, vol. 13, no. 6, pp. 1086–1099, Sep. 2019.
[17] S. Wang *et al.*, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Jun. 2019.
[18] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-IID data with reinforcement learning," in *Proc. INFOCOM*, Jul. 2020, pp. 1698–1707.
[19] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," 2019, *arXiv:1903.03934*. [Online]. Available: http://arxiv.org/abs/1903.03934
[20] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala, "Asynchronous online federated learning for edge devices with non-IID data," 2019, *arXiv:1911.02134*. [Online]. Available: http://arxiv.org/abs/1911.02134
[21] Y. Zhang *et al.*, "CSAFL: A clustered semi-asynchronous federated learning framework," 2021, *arXiv:2104.08184*. [Online]. Available: https://arxiv.org/abs/2104.08184
[22] W. Wu, L. He, W. Lin, R. Mao, C. Maple, and S. Jarvis, "SAFA: A semi-asynchronous protocol for fast federated learning with low overhead," *IEEE Trans. Comput.*, vol. 70, no. 5, pp. 655–668, May 2021.
[23] J. Wang and G. Joshi, "Cooperative SGD: A unified framework for the design and analysis of communication-efficient SGD algorithms," 2018, *arXiv:1808.07576*, [Online]. Available: https://arxiv.org/abs/1808.07576
[24] W. Xia, T. Q. S. Quek, K. Guo, W. Wen, H. H. Yang, and H. Zhu, "Multi-armed bandit-based client scheduling for federated learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 11, pp. 7108–7123, Nov. 2020.
[25] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2018, *arXiv:1812.06127*. [Online]. Available: https://arxiv.org/abs/1812.06127
[26] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Differentially private asynchronous federated learning for mobile edge computing in urban informatics," *IEEE Trans. Ind. Informat.*, vol. 16, no. 3, pp. 2134–2143, Mar. 2020.
[27] Z. Chai, Y. Chen, A. Anwar, L. Zhao, Y. Cheng, and H. Rangwala, "FedAT: A communication-efficient federated learning method with asynchronous tiers under non-IID data," 2020, *arXiv:2010.05958*. [Online]. Available: https://arxiv.org/abs/2010.05958
[28] H. R. Feyzmahdavian, A. Aytekin, and M. Johansson, "A delayed proximal gradient method with linear convergence rate," in *Proc. IEEE Int. Workshop Mach. Learn. Signal Process. (MLSP)*, Sep. 2014, pp. 1–6.
[29] J. Hao, Y. Zhao, and J. Zhang, "Time efficient federated learning with semi-asynchronous communication," in *Proc. IEEE 26th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2020, pp. 156–163.
[30] I. M. Baytas, M. Yan, A. K. Jain, and J. Zhou, "Asynchronous multi-task learning," in *Proc. ICDM*, Dec. 2016, pp. 11–20.
[31] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, vol. 87. Springer, 2003.
[32] S. Wang *et al.*, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, Apr. 2018, pp. 63–71.
[33] Y. Zhan and J. Zhang, "An incentive mechanism design for efficient edge learning by deep reinforcement learning approach," in *Proc. INFOCOM*, Jul. 2020, pp. 2489–2498.
[34] D. W. Hosmer, Jr., S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*, vol. 398. Hoboken, NJ, USA: Wiley, 2013.
[35] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2014.
[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.
[37] 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.
[38] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," Citeseer, 2009.

Qianpiao Ma received the B.S. degree in computer science from the University of Science and Technology of China in 2014, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Technology. His primary research interests include federated learning, mobile edge computing, and game theory.

Yang Xu (Member, IEEE) received the B.S. degree from the Wuhan University of Technology in 2014 and the Ph.D. degree in computer science and technology from the University of Science and Technology of China in 2019. He is currently an Associate Researcher with the School of Computer Science and Technology, University of Science and Technology of China. His primary research interests include ubiquitous computing, deep learning, and mobile edge computing.



Hongli Xu (Member, IEEE) received the Ph.D. degree in computer software and theory from the University of Science and Technology of China, China, in 2007. He is currently a Professor at the School of Computer Science and Technology, University of Science and Technology of China. He has published more than 100 papers in famous journals and conferences. His primary research interests are software defined networks, edge computing, and the Internet of Thing. He has won the best paper award or the best paper candidate in several famous conferences.

Zhida Jiang received the B.S. degree from the Hefei University of Technology in 2019. He is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, University of Science and Technology of China. His primary research interests include mobile edge computing and federated learning.

Liusheng Huang (Member, IEEE) received the M.S. degree in computer science from the University of Science and Technology of China in 1988. He is currently a Senior Professor and a Ph.D. Supervisor with the School of Computer Science and Technology, University of Science and Technology of China. He has authored or coauthored six books and over 300 journal/conference papers. His research interests are in the areas of the Internet of Things, vehicular *ad-hoc* networks, information security, and distributed computing.

He Huang (Member, IEEE) received the Ph.D. degree from the School of Computer Science and Technology, University of Science and Technology of China (USTC), China, in 2011. From 2019 to 2020, he was a Visiting Research Scholar with Florida University, Gainesville, USA. He is currently a Professor with the School of Computer Science and Technology, Soochow University, China. He has authored over 100 papers in related international conference proceedings and journals. His current research interests include traffic measurement, computer networks, and algorithmic game theory. He is a member of the Association for Computing Machinery (ACM). He received the best paper awards from Bigcom 2016, IEEE MSN 2018, and Bigcom 2018. He has served as the technical program committee member of several conferences, including IEEE INFOCOM, IEEE MASS, IEEE ICC, and IEEE Globecom.