# AAFL: Asynchronous-Adaptive Federated Learning in Edge-Based Wireless Communication Systems for Countering Communicable Infectious Diseasess

Jieren Cheng, *Member, IEEE*, Ping Luo, *Student Member, IEEE*, N. Xiong, *Senior Member, IEEE*, and Jie Wu, *Fellow, IEEE*

***Abstract*— With the rapid growth of the coronavirus disease of 2019 (COVID-19) cases, massive amounts of relevant data are being trained on machine learning models for countering communicable infectious diseases. Federated Learning (FL) is a paradigm of distributed machine learning to deal with the individual COVID-19 data, and enable the protection of data privacy. However, FL has low efficiency in Edge-Based wireless communication systems with system heterogeneity. In this paper, we propose an "*Asynchronous-Adaptive FL*" (AAFL) scheme. Specifically, we allow that medical devices with different performances have a heterogeneous number of local SGD iterations in each communication round, called asynchronous iteration strategy which is balanced under adaptive control. We theoretically analyze the convergence of the AAFL scheme under a given time budget and obtain a mathematical relationship between the heterogeneous number of local SGD iterations and the optimal model parameters. Based on the mathematical relationship, we design an algorithm for parameter server and work nodes to adaptively control the heterogeneous number of local SGD iterations. Subsequently, we build a prototype heterogeneous system and conduct experiments on various scenarios for analyzing the general properties of our algorithm, and then apply our algorithm to public COVID-19 databases. The experimental results and application performance demonstrate the effectiveness and efficiency of our AAFL scheme.**

***Index Terms*— Edge computing, infectious diseases, COVID-19, machine learning, federated learning, heterogeneous systems.**

## I. Introduction

THE coronavirus disease of 2019 (COVID-19) instigated a global pandemic of viral pneumonia. To counter the communicable infectious disease of COVID-19, there is a growing urgency for speedy and accurate detection in likely patients with COVID-19 [1]. The above detection needs efficient decision rules from a huge amount of relevant data, e.g., chest X-ray image [2], and machine learning (ML) helps address it by refining diagnosis capacity and modelling techniques [3]. In addition, ML can overcome the scarcity of individual COVID-19 data by data integration across scattered locations in a centralized repository [4]. However, the centralized repository of COVID-19 data is expensive and limited by the inherent issue of data sharing, which has ethical, regulatory and legal complexities for the owners' privacy protection [5]. Thus, it is crucial to develop a widely available learning technique which trains a model for decentralized individual COVID-19 data while maintaining privacy.

One solution is Federated Learning (FL) [6], which is a paradigm of distributed ML and allows multiple decentralized users to train an ML model on their individual COVID-19 databases without sharing patient information. FL was initially developed for mobiles, the Internet of Things, and edge devices, and recently attained the popularity of the health domain in Edge-Based wireless communication systems [7], [8], [9].

In the context of the COVID-19 detection system, FL include medical devices (e.g., X-ray machine), called work nodes, which hold the patients' private training database. Work nodes train local models with Stochastic Gradient Descent (SGD) [10] in parallel and send the updated model parameters to a centralized parameter server. The server aggregates the models from work nodes, then updates the weighted average for the new model parameters, and pushes the new model back to all work nodes as the initial point for the next local SGD iterations. The above intervals start with the distribution of new model from the server, and end with the submission of the updated model parameters from all work nodes, which are called the communication rounds (which we refer to as 'round' in the following). After several given rounds, the trained model gives the identification of 'Normal', 'Pneumonia' or 'COVID-19' patients through chest X-ray images. However, the COVID-19 detection system consists of the work nodes which have significant variability in different devices and networks, and the heterogeneity of the work nodes results in the inefficiency of the whole training process [11], [12], [13]. Therefore, it is challenging to efficiently perform FL on the COVID-19 detection system based on heterogeneous devices and networks.

Manuscript received 15 January 2022; revised 17 June 2022; accepted 21 July 2022. Date of publication 3 October 2022; date of current version 14 November 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 62162024 and Grant 62162022, in part by the Key Projects in Hainan Province under Grant ZDYF2021GXJS003 and Grant ZDYF2020040, and in part by the Major Science and Technology Project of Hainan Province under Grant ZDKJ2020012. *(Corresponding authors: Ping Luo; N. Xiong.)*

Jieren Cheng, Ping Luo, and N. Xiong are with the School of Computer Science and Technology, Hainan University, Haikou 570228, China (e-mail: cjr22@163.com; luoping@hainanu.edu.cn; nnxiong10@yahoo.com).

Jie Wu is with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122 USA (e-mail: jiewu@temple.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2022.3211564.

Digital Object Identifier 10.1109/JSAC.2022.3211564

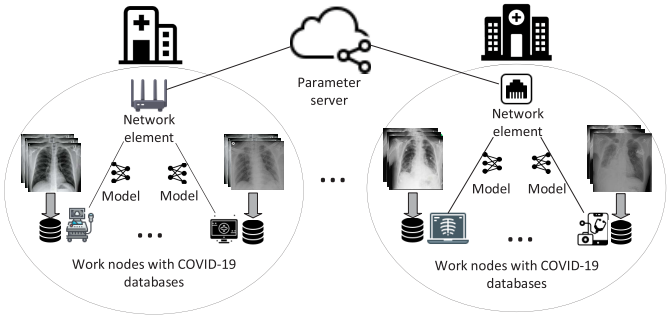



Fig. 1. FL in Edge-Based wireless communication systems for countering communicable infectious diseases.

In this paper, we address the problem of how to efficiently utilize the heterogeneous number of local SGD iterations (associated with computing speeds and networks) [14] and time resource [15] in the COVID-19 detection system based on heterogeneous devices and networks for the optimal FL performance. As illustrated in Fig. 1, we consider a typical FL architecture in Edge-Based wireless communication systems for countering communicable infectious diseases of COVID-19. Work nodes are composed of Edge-Based medical devices with different computing capabilities, and carry the local training database of COVID-19 in an environment which is composed of different network elements. According to these differences, the work nodes are divided into fast work nodes and slow work nodes, fast work nodes take a shorter time than slow work nodes to complete a fixed number of local SGD iterations and to submit the results of FL in a round. The performance of different work nodes results in the synchronization barrier on the fixed number of local SGD iterations [11], which means that the time of completing each round depends on the maximum training and submission time among the slow work nodes. To solve the FL synchronization barrier in the COVID-19 detection system based on heterogeneous devices and networks, we allow all the different work nodes have a heterogeneous number of local SGD iterations in each round, called asynchronous iteration strategy.

As for the asynchronous iteration strategy, we focus on the relationship between the heterogeneous number of local SGD iterations and the convergence of the FL final model. In asynchronous iteration strategy, fast work nodes can perform more local SGD iterations than slow work nodes within a given time resource, thus there is a large disparity between the number of local SGD iterations of work nodes, and the new model of parameter server strays towards the local model of fast work nodes, away from the optimal model [14]. Therefore, it is necessary to balance the fast work nodes and slow work nodes under time resource budget for stable performance.

We propose an Asynchronous-Adaptive FL (AAFL) scheme to adaptively control the number of local SGD iterations at each work node in each round, so that the time resource is most efficiently used to get the optimal model of FL in the COVID-19 detection system based on heterogeneous devices and networks. The main contributions of this paper are as follows:

TABLE I
SUMMARY OF MAIN NOTATIONS

| | |
|---|---|
| $F(\boldsymbol{w})$ | Global loss function |
| $F_i(\boldsymbol{w})$ | Local loss function at work node $i$ |
| $k$ | Round index |
| $\lambda$ | Local SGD iteretion index |
| $\boldsymbol{w}_k^{i,\lambda}$ | Local model parameters at work node $i$, round index $k$ and iteration index $\lambda$ |
| $\widetilde{\boldsymbol{w}}$ | Local model parameters submitted to the parameter server |
| $\boldsymbol{w}_k$ | Global model parameters at round index $k$ |
| $\boldsymbol{w}^f$ | Final global model parameters obtained at the end of process |
| $\boldsymbol{w}^*$ | True optimal model parameters that minimizes $F(\boldsymbol{w})$ |
| $\eta$ | Learning rate in SGD |
| $\tau_i$ | The number of local SGD iterations at work node $i$ in a round |
| $N$ | Number of work nodes |
| $K$ | Total number of rounds |
| $T$ | Total budget of time |
| $c_k^i$ | Consumption of time in one local SGD iteration at work node $i$ |
| $b_k$ | Consumption of time in one global aggregation step |
| $\rho_i$ | Lipschitz parameter of $F_i(\boldsymbol{w})$, defined in Assumption 1 |
| $\beta_i$ | Smoothness parameter of $F_i(\boldsymbol{w})$, defined in Assumption 1 |
| $\theta_k$ | Gradient divergence in round $k$, defined in Definition 2 |
| $g_i(\lambda)$ | Function defined in Theorem 1, gap between the local model parameters obtained from $\lambda$ and $\lambda-1$ at work node $i$ |
| $\omega$ | Constant defined in Lemma 1, control parameter |

- We perform a qualitative analysis of the FL problem in the COVID-19 detection system based on the heterogeneous devices and networks, and obtain a novel convergence bound of the AAFL scheme. The convergence bound incorporates with the heterogeneous number of local SGD iterations at each work node and the loss function of the trained model under a fixed time budget.
- Using the above theoretical convergence bound, we propose the AAFL algorithm that adaptively balances the number of local SGD iterations at each work node to accelerate the speed of model convergence.
- We evaluate the general performance of the AAFL scheme via extensive experiments on general public datasets, and apply the AAFL scheme to public COVID-19 X-ray images on our heterogeneous prototype system, which confirms that the AAFL scheme provides efficient performance in Edge-Based wireless communication systems for countering communicable infectious diseases.

The main notations in this paper are summarized in Table I and this paper is organized as follows. Related work is introduced in Section II. The basics of FL is summarized in Section III. The convergence analysis and control algorithm of AAFL scheme are presented in Section IV. Experimentation results and the application in COVID-19 are shown in Section V. The conclusion is presented in Section VI.

## II. RELATED WORK

The most recent work focuses on applying FL for COVID-19 detection [16], [17], [18], which study medical diagnostic images, for example, the chest X-ray images of COVID-19. There are two main motivations for applying the FL approach in these works, the insufficiency of data and privacy concerns. However, they do not address the coordination of different work nodes (medical devices) in the

COVID-19 detection system based on heterogeneous devices and networks, which is important for improving the efficiency of FL for countering the communicable infectious disease of COVID-19. In the setup of a heterogeneous FL system, there are two common schemes for the coordination based on parallel iterative optimization algorithms: asynchronous and synchronous.

The asynchronous scheme can better solve the problem of scattered devices in the heterogeneous multi-device environment, and computation heterogeneity is a more pronounced bottleneck [19]. Thus, the asynchronous FL requires flexibility and scalability, and it is proved that the asynchronous FL has near-linear convergence to a global optimum, for both strongly convex and a restricted family of non-convex problems [20]. In addition, the practicality of asynchronous SGD is a non-trivial problem due to the complex heterogeneous environment. A Semi-Asynchronous federated learning protocol (FedSA) in [21] addresses the problems in FL such as low round efficiency and poor convergence rate. An FL architecture in [22], called CoLearn, is deployed on resource-constrained IoT devices to demonstrate an asynchronous participation mechanism. An asynchronous FL scheme in [23] performs the convergence boosting by updating verification and weighted aggregation for resource sharing in vehicular networks. An enhanced federated learning technique in [24] is an asynchronous learning strategy on the work nodes that the different layers of the Deep Neural Networks (DNNs) are categorized into shallow and deep layers. All these works show that the asynchronous scheme is widely adopted for the efficiency of FL.

Despite its popularity, the practical performance of asynchronous FL methods for solving large scale machine learning problems are not as good as theoretical results indicate, as known to suffer from the problem of delayed gradients [25]. A natural solution is to use the local solver in which the work nodes train their model independently and synchronize every once in a while, called synchronous scheme [6].

In synchronous FL, most recent works analyze the convergence of federated optimization algorithms with the assumption that the dataset of local SGD is Idenpendent and Identically Distributed (IID). For IID data, it is proved that synchronous FL converges for convex problems with local GD/SGD [26]. Also, the intensive convergence analysis of IID data shows that synchronous FL can be applied more generally and is less expensive [27]. Further, the convergence bound of synchronous FL is analyzed from a theoretical point of view [15], [28]. Except for the convergence analysis of IID data, the local SGD method is also a concern. A new analysis of local SGD removes unnecessary assumptions and elaborates on the difference between two data regimes: IID and Non-IID [29]. Likewise, local SGD is analyzed with delayed updates on smooth quasiconvex and non-convex functions and derives concise, non-asymptotic, convergence rates [30]. It is proved that local SGD strictly dominates minibatch SGD and presents a lower bound on the performance [31].

Further, for analyzing the convergence of synchronous local SGD in FL, the Non-IID dataset is a much more common situation. It is provided in a thorough and rigorous theoretical study that shows why local SGD can work as well as parallel mini-batch SGD with significantly less communication overhead and Non-IID dataset [32]. The proof of local SGD shows that there is a trade-off between communication efficiency and convergence rate for Non-IID data [33], and achieves a linear iteration speedup with a lower communication complexity even if work nodes access Non-IID datasets [34]. The convergence of local SGD on Non-IID data is analyzed in [33] and [35], which establish a convergence rate of strongly convex and smooth problems and demonstrates the applicability of local GD/SGD in federated learning. However, it is proved that heterogeneity (Non-IID) data results in a 'drift' in the local SGD resulting in poor performance [36]. Therefore, it is difficult to determine the optimal number of local SGD iterations for work nodes in each round to improve the training speed in the COVID-19 detection system based on heterogeneous devices and networks.

To our best knowledge, we are the first to propose a novel Asynchronous-Adaptive scheme and formally address the problem of dynamically determining the number of local SGD iterations at different work nodes. The AAFL scheme has a faster convergence rate than the synchronous scheme because of the asynchronous iteration strategy, and has a more stable performance than the asynchronous scheme because of adaptively controlling the number of local SGD iterations. For these reasons, the AAFL scheme is able to optimize the FL model with a given resource budget in the heterogeneous Edge-Based wireless communication systems for countering the communicable infectious disease of COVID-19.

## III. Preliminaries and Definitions

In this section, we first briefly introduce the definitions of loss function in the COVID-19 detection system based on heterogeneous devices and networks (Section III-A). Then, we describe the definitions and calculations of the FL basic algorithm which is the Federated Averaging (FedAvg) algorithm [6], and show the detailed workflow of FedAvg in pseudo-code form (Section III-B). Finally, we formalize our learning problem and ultimate goal with a given time resource constraint (Section III-C).

### A. Loss Function

In the COVID-19 detection system based on heterogeneous devices and networks, each work node has a local model from the parameter server. The local model has a loss function $f_j(\boldsymbol{w})$ defined on its vector of model parameters $\boldsymbol{w}$ (is assumed to be column vectors) for each data sample $j$ [6]. The local loss function captures the prediction variance of the model on the local training datasets of COVID-19 X-ray images, and the model learns the local image features by local SGD iterations to update model parameters, that is to minimize the local loss function. In the next step, we aggregate the loss functions on all work nodes and obtain the global loss function.

According to the definitions in [6] and [15], assume that we have $N$ work nodes with the local training datasets of COVID-19 X-ray images $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_i, \ldots, \mathcal{D}_N$, and the

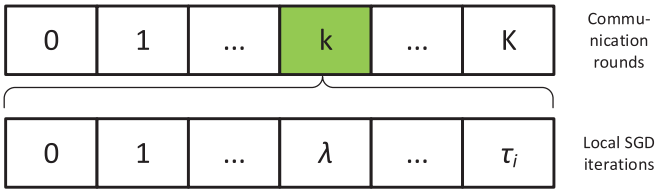



Fig. 2. Two time scales of FedAvg which have $K$ rounds. Each round comprises $\tau_i$ number of local SGD iterations.

loss function on the collection of data samples at work node $i$ is

$$F_i(\boldsymbol{w}) \triangleq \frac{1}{|\mathcal{D}_i|} \sum_{j \in \mathcal{D}_i} f_j(\boldsymbol{w}), \tag{1}$$

where "$\triangleq$" denotes "is defined to be equal to". We define $D_i \triangleq |\mathcal{D}_i|$, where $|.|$ denotes the size of the set, and $D \triangleq \sum_{i=1}^{N} D_i$. Assuming $\mathcal{D}_i \cap \mathcal{D}_{i'} = \emptyset$ for $i \neq i'$, we define the global loss function on all the distributed datasets as

$$F(\boldsymbol{w}) \triangleq \frac{\sum_{i=1}^{N} D_i F_i(\boldsymbol{w})}{D}. \tag{2}$$

Note that $F(\boldsymbol{w})$ cannot be directly computed without sharing information among multiple work nodes.

### *B. The Federated Averaging (FedAvg) Algorithm*

Federated Averaging (FedAvg) algorithm was first proposed in [6], and provided the basic principles for our AAFL scheme in the heterogeneous Edge-Based wireless communication systems for countering the communicable infectious disease of COVID-19. As shown in Fig. 2, FedAvg has $K$ ($K \geq 1$ ) rounds, and each work node has $\tau_i$ ($\tau_i \geq 1$ ) local SGD iterations in a round, where $k = 0, 1, 2, \ldots, K$ denotes the round index and $\lambda = 0, 1, 2, \ldots, \tau_i$ denotes the local SGD iteration index, and the number of $\tau_i$ is set to be fixed on all work nodes and rounds. The server has global model parameters $\boldsymbol{w}_k$ and each work node $i$ has its local model parameters $\boldsymbol{w}_k^{i,\lambda}$ in all rounds. When FedAvg begins ($k = 0$), the server initializes the global model parameters $\boldsymbol{w}_0$ and sends it to all work nodes. At $\lambda = 0$, the local model parameters for all work nodes are received from the server ($\boldsymbol{w}_k^{i,\lambda=0} = \boldsymbol{w}_k$). For $\lambda > 0$, the new values of $\boldsymbol{w}_k^{i,\lambda}$ are computed according to the SGD update rule on the local loss function, based on the local model parameters in the previous iteration $\lambda - 1$. After one or multiple local SGD iterations, a global aggregation is performed through the aggregator (at the parameter server) to update the global model parameters, which is the weighted average of the local model parameters at all work nodes. We define that each above round includes one global aggregation step and $\tau_i$ local SGD iterations. The values of $\tau_i$ are assigned to the same at each work node $i$ and throughout all rounds.

The local SGD in each local iteration is performed on the local model parameters in the previous local iteration. For each work node $i$, the update rule is as follows:

$$\boldsymbol{w}_k^{i,\lambda} = \boldsymbol{w}_k^{i,\lambda-1} - \eta \nabla F_i \left( \boldsymbol{w}_k^{i,\lambda-1} \right), \tag{3}$$

**Algorithm 1** FedAvg in the $k$-th ($k \in [1, K]$) Round

**Input:** $\tau_i$
**Output:** Global model parameters $\boldsymbol{w}_k$ in the $k$-th round
1: Receive the last global model parameters $\boldsymbol{w}_{k-1}$ from aggregator;
2: Initialize $\lambda = 0$ for all work nodes $i$;
3: $\boldsymbol{w}_{k-1}^{i,\lambda} \leftarrow \boldsymbol{w}_{k-1}$;
4: **for** $\lambda = 1, 2, \ldots, \tau_i$ **do**
5: Update $\boldsymbol{w}_{k-1}^{i,\lambda}$ using (3);
6: **end for**
7: Collect $\boldsymbol{w}_{k-1}^{i,\lambda=\tau_i}$ from all work nodes $i$ by aggregator;
8: Compute $\boldsymbol{w}_k$ using (4) by aggregator;

where $k \in [0, K]$, $\lambda \in [1, \tau_i]$ and $\eta > 0$ denotes the learning rate which is used in SGD. In this paper, $\eta$ is fixed with a pre-specified value and is equal at all work nodes. For the global aggregation step, we define

$$\boldsymbol{w}_k = \frac{\sum_{i=1}^{N} D_i \boldsymbol{w}_{k-1}^{i,\lambda=\tau_i}}{D}. \tag{4}$$

We define that the value of round $K$ is limited by the total budget of time $T$. In the $k$-th ($k \in [1, K]$) round, the logic of FedAvg with distributed SGD is presented in Algorithm 1, which ignores aspects related to the communication between the aggregator and edge work nodes [6]. We define that $\boldsymbol{w}_{k-1}$ is the global model parameters in the last round $k-1$ (Line 3 of Algorithm 1), and the global model parameters is defined in (4). When $k = 1$, the value of $\boldsymbol{w}_{k-1}$ is equal to the initialized $\boldsymbol{w}_0$.

We note that when $\tau_i = 1$ in Algorithm 1, i.e., when we perform the global aggregation step after every local SGD iteration, the distributed SGD (ignoring communication aspects) is equivalent to the centralized SGD, where the latter assumes that all data samples are available at a centralized location. This is due to the linearity of the gradient operator. See Appendix A for detailed discussions about this.

### *C. Problem Formulation*

In this paper, our ultimate goal is efficiently getting the highly accurate identifications of COVID-19 patients through chest X-ray images, which is based on FedAvg algorithm in the heterogeneous Edge-Based wireless communication systems. Thus, the first learning problem of FL is to minimize the global loss function $F(\boldsymbol{w})$, i.e., to find

$$\boldsymbol{w}^* \triangleq \arg\min \; F(\boldsymbol{w}), \tag{5}$$

and we define

$$\boldsymbol{w}^f \triangleq \underset{\boldsymbol{w} \in \{\boldsymbol{w}_k : k=0,1,2,\ldots,K\}}{\arg\min} F(\boldsymbol{w}). \tag{6}$$

There are the inherent complexity and communication overhead of the large-scale distributed training of neural networks in FL, it is usually impossible to find a closed-form solution within the effective time to (5). Thus, (5) is often solved using SGD techniques for FedAvg algorithm [37], [38], [39].

When a large amount of chest X-ray images are distributed at different work nodes in a heterogeneous health system, the FL process can consume a significant amount of time. One often has to limit the time used for learning each model, in order not to backlog the system and to keep the operational cost low. This is particularly important in Edge-Based wireless environments where the computation and communication resources are not as abundant as in data centers.

Therefore, a natural question is how to make the efficient use of a given time to minimize the loss function of model training. Our solution is the asynchronous iteration strategy, where the number of local SGD iterations $\tau_i$ is no longer set to be fixed can be adaptively controlled, called Asynchronous-Adaptive FL (AAFL), and the question narrows down to determining the optimal values of $\tau_i$ at each work node $i$ in all rounds, so that the global loss function is minimized subject to a given time resource constraint for this learning task.

In the $k$-th ($k \in [1, K]$) round, we define that each local SGD iteration at work node $i$ consumes $c_k^i$ time resource, and each global aggregation step consumes $b_k$ time resource. To make the problem tractable, we assume that $c_k^i$ is the same for all local SGD iterations $\lambda$ of work node $i$ on the $k$-th round, and $b_k$ is the same for all work nodes. Moreover, we assume that the initialization process for $k = 0$ does not consume $c_k^i$ and $b_k$. When $T$ denotes the total budget of time, we seek the solution to the following problem:

$$\min_{\{\tau_i\}, K \in \{1,2,3,\ldots\}} F\left(\boldsymbol{w}^f\right) \quad \text{s.t.} \sum_{k=1}^{K} \left(\max_i c_k^i \tau_i + b_k\right) \leq T, \tag{7}$$

where $\{\tau_i\}$ represents the set of the number of local SGD iterations on $N$ work nodes $\{\tau_1, \tau_2, \ldots, \tau_i, \ldots, \tau_N\}$.

To solve (7), we need to find out how $\{\tau_i\}$ and $K$ (and thus T) affect the loss function $F\left(\boldsymbol{w}^f\right)$ computed on the final global model parameters $\boldsymbol{w}^f$.

## IV. Our Proposed AAFL Scheme

In this section, we first analyze the convergence of the AAFL scheme and get a convergence upper bound. Next, we find an approximate solution to (7) according to the relationship between the convergence upper bound and the value of $\{\tau_i\}$. Based on the solution, we design the algorithm of parameter server and work nodes to adaptively controls the value of $\{\tau_i\}$ for the AAFL scheme.

### A. Convergence Analysis

We analyze the convergence of the AAFL scheme in this subsection and find an upper bound of $F(\boldsymbol{w}^f) - F(\boldsymbol{w}^*)$. To facilitate the analysis, we first introduce some notations. At the end of each round $k = 1, 2, \ldots, K$, the aggregator computes the global loss function that we define

$$F\left(\boldsymbol{w}_k\right) \triangleq \frac{\sum_{i=1}^{N} D_i F_i\left(\boldsymbol{w}_{k-1}^{i,\lambda=\tau_i}\right)}{D}, \tag{8}$$

where $F_i\left(\boldsymbol{w}_{k-1}^{i,\lambda=\tau_i}\right)$ is local loss function which is computed from $\boldsymbol{w}_{k-1}^{i,\lambda}$ and local training dataset at work node $i$.

To facilitate comparison with $F\left(\boldsymbol{w}_k\right)$, we use $\boldsymbol{v}_k^{i,\lambda}$ to denote an auxiliary vector of model parameters that has the same size of $\boldsymbol{w}_k^{i,\lambda}$, and follows a centralized SGD according to

$$\boldsymbol{v}_k^{i,\lambda} = \boldsymbol{v}_k^{i,\lambda-1} - \eta \nabla F\left(\boldsymbol{v}_k^{i,\lambda-1}\right), \tag{9}$$

where $\boldsymbol{v}_k^{i,\lambda}$ is only defined for $\lambda \in [1, \tau_i]$ for a given $k$ at work node $i$. This update rule is based on the loss function $F\left(\boldsymbol{v}_k^{i,\lambda}\right)$ which is trained on all data samples at a central place (thus we call it centralized SGD), whereas the iteration in (3) is on the local loss function $F_i\left(\boldsymbol{w}\right)$ and local data samples. Equivalently, we define

$$F\left(\boldsymbol{v}_k\right) \triangleq \frac{\sum_{i=1}^{N} D_i F\left(\boldsymbol{v}_{k-1}^{i,\lambda=\tau_i}\right)}{D}. \tag{10}$$

Note that $F\left(\boldsymbol{v}_k\right) \neq F\left(\boldsymbol{v}_{k-1}^{i,\lambda=\tau_i}\right)$ when $\tau_i$ is different (and thus the different loss functions $F\left(\boldsymbol{v}_{k-1}^{i,\lambda=\tau_i}\right)$ due to centralized SGD) for each work node. We define that $\boldsymbol{v}_k^{i,\lambda}$ is "synchronized" with $\boldsymbol{w}_k^{i,\lambda}$ at the beginning of each round $k$, i.e., $\boldsymbol{v}_k^{i,\lambda=0} = \boldsymbol{w}_k^{i,\lambda=0} = \boldsymbol{w}_k$ ($k \in [0, K]$), where $\boldsymbol{w}_k$ is global model parameters defined in Section III-B.

The above definitions enable us to find the convergence bound of FedAvg by taking a two-step approach. The first step is to find the gap between $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ for each round $k$, which is the difference between the distributed and centralized SGD after $\tau_i$ steps of local SGD and one global aggregation at all work nodes. The second step is to combine this gap with the convergence bound of $\boldsymbol{v}_k$ within each round $k$ to obtain the convergence bound of $\boldsymbol{w}_k$.

For further analysis, we use the following assumption in [15] to the loss function.

*Assumption 1: We assume the following is valid for any work node $i$:*

- $F_i\left(\boldsymbol{w}\right)$ *is convex*
- $F_i\left(\boldsymbol{w}\right)$ *is* $\rho$*-Lipschitz, i.e.,* $\left\|F_i\left(\boldsymbol{w}\right) - F_i\left(\boldsymbol{w}'\right)\right\| \leq \rho_i \left\|\boldsymbol{w} - \boldsymbol{w}'\right\|$ *for any* $\boldsymbol{w}$, $\boldsymbol{w}'$ *at work node* $i$
- $F_i\left(\boldsymbol{w}\right)$ *is* $\beta$*-smooth, i.e.,* $\left\|\nabla F_i\left(\boldsymbol{w}\right) - \nabla F_i\left(\boldsymbol{w}'\right)\right\| \leq \beta_i \left\|\boldsymbol{w} - \boldsymbol{w}'\right\|$ *for any* $\boldsymbol{w}$, $\boldsymbol{w}'$ *at work node* $i$

The symbol of $\|.\|$ denotes the $L^2$ norm, and Assumption 1 is satisfied for squared support vector machine (SVM), which is an example of the loss function defined in Section III-A and a generalized linear classifier that performs binary classification of data. The experimentation results that will be presented in Section V show that our control algorithm also works well for models (such as the neural network) whose loss functions do not satisfy Assumption 1.

We also define the following metric to capture the divergence between the gradient of a local loss function and the gradient of the global loss function. This divergence is related to how the training data is distributed at different work nodes.

*Definition 1 (Gradient Divergence): For any* $i$ *and* $\boldsymbol{w}$*, we define* $\delta_i$ *as an upper bound of* $\left\|\nabla F_i\left(\boldsymbol{w}\right) - \nabla F\left(\boldsymbol{w}\right)\right\|$*, i.e.,*

$$\left\|\nabla F_i\left(\boldsymbol{w}_k^{i,\lambda}\right) - \nabla F\left(\boldsymbol{w}_k^{i,\lambda}\right)\right\| \leq \delta_i. \tag{11}$$

The below theorem gives an upper bound on the difference between $\boldsymbol{w}_k^{i,\lambda}$ and $\boldsymbol{v}_k^{i,\lambda}$ when $\lambda$ is within the round $k$.

*Theorem 1: For any $k$, $i$ and $\lambda \in [0, \tau_i]$, we have*

$$\left\| \boldsymbol{w}_k^{i,\lambda} - \boldsymbol{v}_k^{i,\lambda} \right\| \leq g_i(\lambda), \tag{12}$$

*where*

$$g_i(\lambda) \triangleq \frac{\delta_i}{\beta_i} \left( (\eta\beta_i + 1)^{\lambda} - 1 \right). \tag{13}$$

*Furthermore, as $F_i(\boldsymbol{w})$ is $\rho$-Lipschitz, we have*

$$F(\boldsymbol{w}_k) - F(\boldsymbol{v}_k) \leq \frac{\sum_{i=1}^{N} D_i \rho_i g_i(\tau_i)}{D}. \tag{14}$$

*Proof:* We first obtain an upper bound of $\left\| \boldsymbol{w}_k^{i,\lambda=\tau_i} - \boldsymbol{v}_k^{i,\lambda=\tau_i} \right\|$ for each work node $i$, then accumulate the above results. For details, see Appendix B. □

Note that we always have $\eta > 0$ and $\beta_i > 0$ because otherwise the SGD procedure or the loss function becomes trivial. Therefore, we have $(\eta\beta_i + 1)^{\lambda} \geq \eta\beta_i\lambda + 1$ for $\lambda = 0, 1, 2, \ldots, \tau_i$ due to Bernoulli's inequality. Substituting this into (13) confirms that we always have $g_i(\lambda) \geq 0$ .

It is easy to see that $g_i(0) = 0$. Therefore, when $\lambda = 0$, i.e., at the beginning of the round $k$, the upper bound in (12) is zero. This is consistent with the definition of $\boldsymbol{v}_k^{i,\lambda=0} = \boldsymbol{w}_k^{i,\lambda=0}$ for any $k$ at all work nodes $i$. When $\lambda = 1$ (i.e., the second iteration in round $k$), the upper bound in (12) is $\eta\delta_i$. This agrees with the definition of (3), (9) and (11), showing that there is gradient divergence between distributed and centralized SGD when only one local SGD iteration is performed after the global aggregation.

For $\lambda > 1$, the value of $\lambda$ can be larger. When $\lambda$ is large, the exponential term with $(\eta\beta_i + 1)^{\lambda}$ in (13) becomes dominant, and the gap between $\boldsymbol{w}_k^{i,\lambda}$ and $\boldsymbol{v}_k^{i,\lambda}$ can increase exponentially with $\lambda$ for $\lambda \in [0, \tau_i]$. We also note that $g_i(\lambda)$ is proportional to the gradient divergence $\delta_i$, which is intuitive because the more the local gradient is different from the global gradient, the larger the gap will be. The gap is caused by the difference in the local gradients at different work nodes starting at the first local SGD iteration after each global aggregation. Thus, when all work nodes have the exactly same data samples (and thus the same local loss function), the gradients will always be the same and $\delta_i = 0$, in which case $\boldsymbol{w}_k^{i,\lambda}$ and $\boldsymbol{v}_k^{i,\lambda}$ are always equal in each round $k$.

Then, (14) gives an upper bound of the difference between distributed and centralized SGD at the end of round $k$ (after global aggregation). Based on these results, we first obtain the following lemma.

*Lemma 1: When all the following conditions are satisfied*:

- $0 < \eta\beta_i \leq 2$
- $F_i\left(\boldsymbol{v}_k^{i,\lambda}\right) - F(\boldsymbol{w}^*) \geq \varepsilon$ *for all* $k$ *and* $\lambda$
- $F(\boldsymbol{w}_K) - F(\boldsymbol{w}^*) \geq \varepsilon$

*where $\varepsilon$ is a unknown quantity that represent the lower bound of $F_i\left(\boldsymbol{v}_k^{i,\lambda}\right) - F(\boldsymbol{w}^*)$ and $F(\boldsymbol{w}_K) - F(\boldsymbol{w}^*)$, and we define $\omega = \min_{i,\lambda} \frac{1}{\left\| \boldsymbol{v}_k^{i,\lambda} - \boldsymbol{w}^* \right\|^2}$, and $\varphi_i \triangleq \left(1 - \frac{\eta\beta_i}{2}\right)\omega$. It is easy to see that $0 \leq \varphi_i < \omega$ for $0 < \eta\beta_i \leq 2$ according to the definition of $\eta > 0$ in Section III-B and $\beta_i \geq 0$ in Assumption 1, then the convergence upper bound of Algorithm 1 after K iterations is given by*

$$F(\boldsymbol{w}_K) - F(\boldsymbol{w}^*) \leq I + KB_i - K\eta\varepsilon^2 C_i, \tag{15}$$

*for the definition of that $I = F(\boldsymbol{w}_0) - F(\boldsymbol{w}^*)$, $B_i = \frac{\sum_{i=1}^{N} D_i \rho_i g_i(\tau_i)}{D}$ and $C_i = \frac{\sum_{i=1}^{N} D_i \varphi_i \tau_i}{D}$.*

*Proof:* We first analyze the convergence of $F\left(\boldsymbol{v}_k^{i,\lambda}\right)$ within each round $k$. Then, we combine this result with the gap between $F(\boldsymbol{v}_k)$ and $F(\boldsymbol{w}_k)$ from Theorem 1 to obtain the final result. For details, see Appendix C. □

We then have the following theorem

*Theorem 2: When $0 < \eta\beta_i \leq 2$ , we have*

$$F\left(\boldsymbol{w}^f\right) - F(\boldsymbol{w}^*) \leq \sqrt{\frac{I}{K\eta C_i} + \frac{B_i}{\eta C_i}} + B_i. \tag{16}$$

*Proof:* Condition 1 in Lemma 1 is always satisfied due to the condition $\eta\beta_i \leq 2$ in this theorem.

We can choose $\varepsilon$ to be arbitrarily small (but greater than zero) so that conditions 2–3 in Lemma 1 are satisfied. Combining with the condition 3 and (15), we have

$$\varepsilon \leq F(\boldsymbol{w}_K) - F(\boldsymbol{w}^*) \leq I + KB_i - K\eta\varepsilon^2 C_i. \tag{17}$$

Solving for $\varepsilon$, we obtain

$$\begin{cases} \varepsilon \geq \dfrac{-\sqrt{1 + 4K\eta C_i (I + KB_i)} - 1}{2K\eta C_i} \\ \varepsilon \leq \dfrac{\sqrt{1 + 4K\eta C_i (I + KB_i)} - 1}{2K\eta C_i}, \end{cases}$$

it is obvious that the first term is less than 0, and be ignored according to the definition $\varepsilon > 0$. Thus we define that $\varepsilon_0 > 0$ is the maximum value of $\varepsilon$ and is equal to

$$\begin{aligned} \frac{\sqrt{1 + 4K\eta C_i (I + KB_i)} - 1}{2K\eta C_i} &\leq \frac{\sqrt{4K\eta C_i (I + KB_i)}}{2K\eta C_i} \\ &= \sqrt{\frac{I}{K\eta C_i} + \frac{B_i}{\eta C_i}}, \end{aligned} \tag{18}$$

where the first term is because $\sqrt{1 + \chi} \leq 1 + \sqrt{\chi}$ for $\chi = 4K\eta C_i (I + KB_i) \geq 0$ (according to the definitions in Lemma 1). We know that there does not exist $\varepsilon > \varepsilon_0$ that satisfies both conditions 2 and 3 in Lemma 1, otherwise the $\varepsilon_0$ contradicts with its definition of the maximum value of $\varepsilon$. This means that

$$\min\left\{ \min_{k=1,2,\ldots,K} F(\boldsymbol{v}_k); F(\boldsymbol{w}_K) \right\} - F(\boldsymbol{w}^*) \leq \varepsilon_0. \tag{19}$$

From Theorem 1, $F(\boldsymbol{w}_k) - F(\boldsymbol{v}_k) \leq B_i$ for any round $k$. Combining with (19), we get

$$\min_{k=1,2,\ldots,K} F(\boldsymbol{w}_k) - F(\boldsymbol{w}^*) \leq \varepsilon_0 + B_i. \tag{20}$$

□

Using (20) and (18), we obtain the result in (16). As a result, when $K = \infty$, the upper bound of $F\left(\boldsymbol{w}^f\right) - F(\boldsymbol{w}^*)$ will be converged to $\sqrt{\frac{B_i}{\eta C_i}} + B_i$. Furthermore, we will use the above definitions and theorems to solve the problem in (7).

### B. Approximate Solution to (7)

We assume that $\eta$ is chosen small enough such that $\eta \leq \frac{1}{\beta_i}$ for each work node $i$, and use the upper bound in (16) as an approximation of $F\left(\boldsymbol{w}^f\right) - F\left(\boldsymbol{w}^*\right)$. Because for a given global loss function $F(\boldsymbol{w})$, its minimum value $F(\boldsymbol{w}^*)$ is a constant, the minimization of $F\left(\boldsymbol{w}^f\right)$ in (7) is equivalent to minimizing $F\left(\boldsymbol{w}^f\right) - F\left(\boldsymbol{w}^*\right)$. With this approximation and rearranging the inequality constraints in (7), we can rewrite (7) as

$$\min_{\{\tau_i\}, K\in\{1,2,3,\ldots\}} \sqrt{\frac{I}{K\eta C_i} + \frac{B_i}{\eta C_i}} + B_i$$
$$\text{s.t. } K \leq \frac{T}{\max_{i,k}\left(c_k^i \tau_i + b_k\right)}, \tag{21}$$

where $B_i = \frac{\sum_{i=1}^N D_i \rho_i g_i(\tau_i)}{D}$ and $C_i = \frac{\sum_{i=1}^N D_i \varphi_i \tau_i}{D}$.

It is easy to see that the objective function in (21) decreases with $K$. Therefore, for any $\tau_i$, the optimal value of $K$ is $\lfloor \frac{T}{\max_{i,k}\left(c_k^i \tau_i + b_k\right)} \rfloor$, i.e., the largest value of K that does not violate any inequality constraint in (21), where $\lfloor . \rfloor$ denotes the floor function for rounding down to integer. To simplify the analysis, we approximate by ignoring the rounding operation and substituting $K \approx \frac{T}{\max_{i,k}\left(c_k^i \tau_i + b_k\right)}$ into the objective function in (21), yielding

$$G(\{\tau_i\}) \triangleq \sqrt{\frac{IT}{\max_{i,k}\left(c_k^i \tau_i + b_k\right)\eta C_i} + \frac{B_i}{\eta C_i}} + B_i, \tag{22}$$

and we can define the (approximately) optimal $\{\tau_i\}$ as

$$\{\tau_i^*\} = \underset{\tau_i\in\{1,2,3,\ldots\}}{\arg\min} \; G(\{\tau_i\}). \tag{23}$$

There is no closed-form solution for $\{\tau_i^*\}$ and the determination of convexity for $G(\{\tau_i\})$, because $G(\{\tau_i\})$ includes both the polynomial and exponential terms of $\tau_i$, where the exponential term is embedded in $g_i(\tau_i)$. As the value of $\tau_i$ can only be a positive integer according to the definition of $\tau$, one strategy is to compute $G(\{\tau_i\})$ within a finite range of $\tau_i$ at all work nodes to find $\{\tau_i^*\}$ that minimizes $G(\{\tau_i\})$, and it can finally find the global optimal value [15]. However, due to its exponential complexity, when the number of work nodes increases, the computation becomes unworkable. Therefore, we use the other approximate solution strategy:

- We use the gradient descent method in each round $k$, where the set $\{\tau_i\}$ is the variable associated with the function $G(\{\tau_i\})$, and the parameters $c_k^i$, $b_k$, $\rho_i, \beta_i$, and $\delta_i$ are constants associated with the function $G(\{\tau_i\})$.
- The set $\{\tau_i\}$ is initialized to the same fixed value for all work nodes at the beginning of the gradient descent.
- During the gradient descent, the value of $\{\tau_i\}$ will change to reduce the value of the function $G(\{\tau_i\})$. Therefore, we obtain an approximate optimal result $\{\tau_i^*\}$ which minimize the function $G(\{\tau_i\})$ at the end of the gradient descent.

We assume that the value of $\tau$ is not an integer during gradient descent for smoother operation, and it finally obtains a rounded local optimum value around the initial value of $\{\tau_i^*\}$. Compared to training the model of FL, the time spent on the gradient descent method is negligible and belongs to the global aggregation time $b_k$. We implement our approximate solution strategy in the following control algorithm of AAFL and obtain the optimal result $\{\tau_i^*\}$ in each round. The effective performance of our strategy will be presented in Section V.

### C. Control Algorithm of AAFL

We propose an Asynchronous-Adaptive FL scheme, called AAFL, which approximately solves (7) in Edge-Based wireless communication systems for countering the communicable infectious disease of COVID-19. We first set the time budget $T$ and the initialized values of $\{\tau_i\}$ in (22) for the first local SGD iteration and the first global aggregation. Then, we consider practical scenarios where $c_k^i$, $b_k$, and some other parameters are unknown and may vary over time, and we propose a control algorithm that estimates the parameters and dynamically adjusts the value of $\{\tau_i\}$ in real-time. Specifically, we present the complete control algorithm for the AAFL scheme, which recomputes $\{\tau_i^*\}$ in every global aggregation step based on the most recent system state. We use the theoretical results above to guide the design of the algorithm.

As illustrated in Fig. 3, the parameter server and work nodes train a model from the datasets of COVID-19 X-ray images in medical devices, where different work nodes have different time points for completing tasks. The architecture of the AAFL scheme is divided into three main building blocks:

- **Local SGD:** All work nodes receive a global model $\boldsymbol{w}_k$ from parameter server, and perform local SGD iterations from the local training datasets of COVID-19 X-ray images. After finishing $\tau_i$ local SGD iterations, each work node sends the result of local model $\widetilde{\boldsymbol{w}}$ to the parameter server.
- **Server aggregate:** The parameter server receives the local model $\widetilde{\boldsymbol{w}}$ from each work node $i$, and computes $\boldsymbol{w}_k$ according to (4).
- **Compute new $\{\tau_i^*\}$:** The parameter server receives control parameters $E_i$ from each work node $i$, where $E_i$ is defined as the estimated value of $c_k^i$, $b_k$, $\rho_i, \beta_i$, and $\delta_i$. Then, the parameter server computes a new approximate value of $\{\tau_i^*\}$, and sends it to all work nodes for local updating.

As mentioned earlier, the local SGD iteration runs on the work nodes, and the global aggregation is performed with the assistance of the parameter server (logical component) which runs on a laptop, see details in Section V-A. The complete process of the parameter server and each work node is presented in Algorithm 2 and Algorithm 3, respectively, where Lines 13-16 of Algorithm 3 are local SGD iterations, and the rest are considered as a part of the initialization, global aggregation and computing new $\{\tau_i^*\}$. We assume that the aggregator initiates the learning process, then initializes the model parameters $\boldsymbol{w}_0$ and sends it to all work nodes. We note that instead of transmitting the entire model parameters in every global aggregation step, one can also transmit compressed or quantized model parameters to further save the communication bandwidth, where the compression or quantization can be performed using techniques that described in [40], for instance.

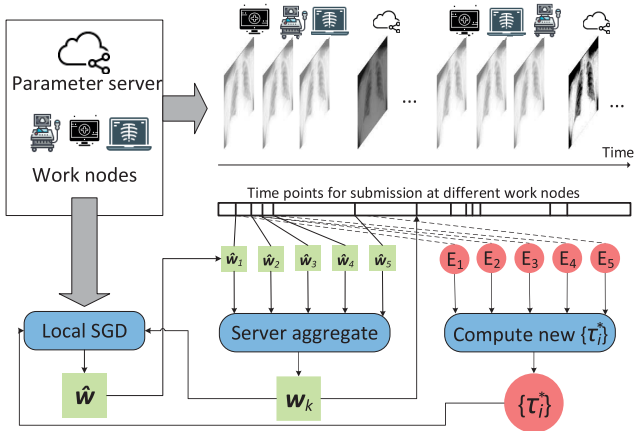



Fig. 3. The architecture of the AAFL scheme.

**Algorithm 2** Procedure at the Parameter Server

**Input:** Time budget $T$, control parameter $\omega$
**Output:** $\boldsymbol{w}^f$

1: Initialize $k=0, m_k=0, s_k=0$;
2: Initialize $\boldsymbol{w}_k$, set $\boldsymbol{w}^f \leftarrow \boldsymbol{w}_k$;
3: Send $\boldsymbol{w}_k$ to all work nodes;
4: Receive $F_i(\boldsymbol{w}_k)$;
5: Compute $F(\boldsymbol{w}_k)$ according to (2);
6: Initialize $I = F(\boldsymbol{w}_k)$;
7: **repeat**
8: Initialize $\{\tau_i\}$;
9: Send $\boldsymbol{w}_k, \tau_i$ and $k$ to all work nodes;
10: **if** $k > 0$ **then**
11: Receive $\hat{\beta}_i$ and $\hat{\rho}_i$;
12: Estimate $\hat{\delta}_i \leftarrow \|\nabla F_i(\widetilde{\boldsymbol{w}}) - \nabla F(\boldsymbol{w}_k)\|$;
13: Receive time consumption $\hat{c}_k^i$ from each work node $i$;
14: Estimate time consumption $\hat{b}_k$;
15: **if** $m_k < \max_i \hat{c}_k^i \tau_i + \hat{b}_k$ **then**
16: $m_k = \max_i \hat{c}_k^i \tau_i + \hat{b}_k$;
17: **end if**
18: Compute the new value of $\{\tau_i^*\}$ for each $i$;
19: $s_k \leftarrow s_k + \max_i \hat{c}_k^i \tau_i + \hat{b}_k$;
20: **if** $s_k \geq T$ **then**
21: Set $STOP$ flag, and send it to all work nodes;
22: **end if**
23: **end if**
24: $k \leftarrow k+1$;
25: Receive $\widetilde{\boldsymbol{w}}$ from each node $i$;
26: Compute $\boldsymbol{w}_k$ according to (4);
27: Receive $\nabla F_i(\widetilde{\boldsymbol{w}})$ from each work node $i$;
28: Compute $\nabla F(\boldsymbol{w}_k) \leftarrow \left(\sum_{i=1}^N D_i \nabla F_i(\widetilde{\boldsymbol{w}})\right)/D$;
29: Receive $F_i(\widetilde{\boldsymbol{w}})$ from each work node $i$;
30: Compute $F(\boldsymbol{w}_k)$ according to (8);
31: **if** $F(\boldsymbol{w}_k) < F(\boldsymbol{w}^f)$ **then**
32: $\boldsymbol{w}^f \leftarrow \boldsymbol{w}_k$;
33: **end if**
34: **until** $STOP$ flag is set

**Algorithm 3** Procedure at Each Work Node $i$

1: Receive $\boldsymbol{w}_k$ from parameter server;
2: Send $F_i(\boldsymbol{w}_k)$ to parameter server;
3: **repeat**
4: Initialize $\lambda \leftarrow 0$ ;
5: Receive $\boldsymbol{w}_k, \tau_i$ and $k$ from parameter server;
6: **if** $k > 0$ **then**
7: Compute $F_i(\boldsymbol{w}_k)$ and $\nabla F_i(\boldsymbol{w}_k)$;
8: Estimate $\hat{\rho}_i \leftarrow \|F_i(\widetilde{\boldsymbol{w}}) - F_i(\boldsymbol{w}_k)\| / \|\widetilde{\boldsymbol{w}} - \boldsymbol{w}_k\|$;
9: Estimate $\hat{\beta}_i \leftarrow \|\nabla F_i(\widetilde{\boldsymbol{w}}) - \nabla F_i(\boldsymbol{w}_k)\| / \|\widetilde{\boldsymbol{w}} - \boldsymbol{w}_k\|$;
10: Send $\hat{\beta}_i$ and $\hat{\rho}_i$ to parameter server;
11: **end if**
12: Set $\boldsymbol{w}_k^{i,\lambda} \leftarrow \boldsymbol{w}_k$;
13: **for** $\mu = 1, 2, \ldots, \tau_i$ **do**
14: $\lambda \leftarrow \lambda + 1$;
15: Perform local local SGD iteration at $\boldsymbol{w}_k^{i,\lambda}$;
16: **end for**
17: Set $\widetilde{\boldsymbol{w}} \leftarrow \boldsymbol{w}_k^{i,\lambda=\tau_i}$;
18: Compute $F_i(\widetilde{\boldsymbol{w}})$ and $\nabla F_i(\widetilde{\boldsymbol{w}})$;
19: Estimate time consumption $\hat{c}_k^i$;
20: Send $\hat{c}_k^i$, $\widetilde{\boldsymbol{w}}$, $F_i(\widetilde{\boldsymbol{w}})$ and $\nabla F_i(\widetilde{\boldsymbol{w}})$ to parameter server;
21: **until** $STOP$ flag is received;

*1) Estimation of Parameters in $G(\{\tau_i\})$:* The expression of $G(\{\tau_i\})$, which includes $g_i(\tau_i)$, has parameters that need to be estimated in practice. Among these parameters, $c_k^i$ and $b_k$ are related to time consumption, $\rho_i, \beta_i$, and $\delta_i$ are related to the loss function characteristics. These parameters are estimated in real-time during the learning process.

The values of $c_k^i$ and $b_k$ are estimated based on the measurements of time consumption at the work nodes and the aggregator of server (Line 14 of Algorithm 2 and Line 19 of Algorithm 3). The value of $m_k$ in Line 16 of Algorithm 2 is the max value of time consumption among all previous $k$ rounds, which is defined in (22), where $\max_i \hat{c}_k^i \tau_i + \hat{b}_k$ is described in (21) as the sum of time consumption in round $k$. The total time consumption of the first $k$ rounds $s_k$ is computed at Line 19 of Algorithm 2 and compared against the time budget $T$. Once the consumed time $s_k$ is over the budget limit, it stops the learning and returns the final result.

The values of $\rho_i$, $\beta_i$, and $\delta_i$ are estimated based on the loss functions and gradients (Line 12 of Algorithm 2 and Lines 8 and 9 of Algorithm 3). The gradient $\nabla F_i(\widetilde{\boldsymbol{w}})$ is computed at $F_i(\widetilde{\boldsymbol{w}})$, and the gradient $\nabla F(\boldsymbol{w}_k)$ is the aggregation of $\nabla F_i(\widetilde{\boldsymbol{w}})$ from all work nodes (Line 28 of Algorithm 2 and Lines 18 of Algorithm 3). To perform the estimation, each edge work node needs to have access to its local model parameters $\boldsymbol{w}_{k-1}^{i,\lambda=\tau_i}$ and the global model parameters $\boldsymbol{w}_k$ at the beginning of round $k$, where we use $\widetilde{\boldsymbol{w}}$ to save the last local model parameters $\boldsymbol{w}_{k-1}^{i,\lambda=\tau_i}$. The global model parameters $\boldsymbol{w}_k$ is only observable by each work node after the global aggregation (see Lines 24 and 26 of Algorithm 2), thus the estimated values of $\rho_i$, $\beta_i$, and $\delta_i$ are only available at the second global aggregation step to recompute $\{\tau_i^*\}$.

The parameter $\eta$ is the learning rate in SGD which is pre-specified and known. The parameter $I$ is the deviation between $F(\boldsymbol{w}_0)$ and $F(\boldsymbol{w}^*)$ according to the definition in Lemma 1, and the value of $I$ is estimated to be equal to the

value of $F(\boldsymbol{w}_0)$ at Line 6 of Algorithm 2, because the value of $F(\boldsymbol{w}^*)$ is usually small that $F(\boldsymbol{w}_0) - F(\boldsymbol{w}^*) \approx F(\boldsymbol{w}_0)$. The remaining parameter $\varphi_i$ (include $\omega$) is non-straightforward to estimate because the algorithm does not know $\boldsymbol{w}^*$, thus we regard it as a control parameter that is manually chosen and remains fixed for the same machine learning model. Experimentation results presented in the next section show that a fixed value of $\omega$ works well across different data distributions, various work nodes, and various time budgets. The value of $\omega$ can be approximated as the inverse of the difference between the initial model $\boldsymbol{w}_0$ and $\boldsymbol{w}^*$, according to the definition of $\boldsymbol{v}_k^{i,\lambda}$ and $\omega$. When a larger value of $\omega$ and $\eta\beta_i \leq 2$ in Lemma 1, it gives a larger value of $\varphi_i$ and then a larger value of $C_i$, yielding a smaller value of $G(\{\tau_i\})$, which represents a smaller gap between the final result $\boldsymbol{w}^f$ and $\boldsymbol{w}^*$. The above analysis shows that the larger value of $\omega$ means the smaller difference of the initial model $\boldsymbol{w}_0$ and the final result $\boldsymbol{w}^f$, and it is contrary to reality. Therefore, in practice, it is usually tuning the value of $\omega$ on a small setup. The sensitivity of $\omega$ is shown in Section V-B6.

*2) Recomputing $\{\tau_i^*\}$:* The value of $\{\tau_i^*\}$ is recomputed by the aggregator during each global aggregation step, based on the most updated parameter estimations. When using the gradient descent method, we set the initial $\{\tau_i\}$ as a trainable value and the other parameters in $G(\{\tau_i\})$ are constant. To prevent the negative values of $\{\tau_i^*\}$, the boundary of $\tau_i$ value is set to [1, 100], because if $\{\tau_i^*\}$ is too large, it is more likely to operate beyond the time budget due to incorrect estimation values in the Edge-Based wireless communication systems. When there are non-representable values in $\{\tau_i^*\}$, the process stops and the result is outputted. The result of new $\{\tau_i^*\}$ means the end of a round (see Line 18 and 24 of Algorithm 2), then the new value is sent to each work node together with $\boldsymbol{w}_k$ and $k$.

The process of computing the new value of $\{\tau_i^*\}$ begins at the round $k = 1$ (see Line 10 of Algorithm 2 and Line 6 of Algorithm 3). This is because for $k = 0$, $\boldsymbol{w}_k = \widetilde{\boldsymbol{w}}$ as the initialization in Line 2-5 of Algorithm 2 and Line 1-2 of Algorithm 3, where we obtain $\rho_i = \beta_i = \delta_i = 0$ and $\frac{\delta_i}{\beta_i}$ in $g_i(\tau_i)$ is invalid. Therefore, our AAFL algorithm actually performs a FedAvg algorithm step first before starting the control, and the experimentation results presented in Section V-B3 show that it only has a certain impact when the time budget is small and the initial value of $\{\tau_i^*\}$ is large.

*3) Handling of SGD:* When using SGD in our AAFL algorithm at all work nodes, all loss functions and their gradients are computed on mini-batches. Each local SGD iteration corresponds to a step of local model parameters update where the gradient is computed on a mini-batch of local training data, and then be performed for local model parameters in Line 15 of Algorithm 3. The mini-batch changes for every step of local iteration, i.e., for each new local iteration, a new mini-batch of a given size is randomly selected from the local training data.

The local SGD iterations of distributed SGD at the edge work nodes include Lines 13-16 of Algorithm 3, where Line 15 of Algorithm 3 corresponds to Line 5 of Algorithm 1. When the global aggregation is performed, Line 26 of Algorithm 2 computes the global model parameters $\boldsymbol{w}_k$ at the aggregator, which is sent to the work nodes in Line 9 of Algorithm 2, and each edge work node receives $\boldsymbol{w}_k$ in Line 5 of Algorithm 3 and sets $\boldsymbol{w}_k^{i,\lambda} \leftarrow \boldsymbol{w}_k$ to use $\boldsymbol{w}_k$ as the initial model parameters for the local SGD iterations.

The optimal model parameters $\boldsymbol{w}^f$ that minimizes $F(\boldsymbol{w})$ is obtained at the aggregator in Lines 32 of Algorithm 2. According to (20), we select the value of $\boldsymbol{w}_k$ in each round (Lines 31-33 of Algorithm 2) to ensure that the final value of $\boldsymbol{w}^f$ is the smallest value of $\boldsymbol{w}_k$. In the next section, we design experiments on the general dataset to verify the general effectiveness of the AAFL algorithm, then apply the AAFL algorithm with public COVID-19 X-ray images databases to validate performance.

## V. Performance Analysis

To evaluate the AAFL algorithm, the experiments focused on the qualitative and quantitative analysis of general properties under the setup of our heterogeneous system. We then applied it to counter the communicable infectious disease of COVID-19 and analyzed the performance.

### A. Setup

We first evaluate the general performance of our AAFL algorithm in Edge-Based wireless communication systems, thus we conducted experiments on a networked prototype system with five work nodes. The prototype system consists of five Raspberry Pi (version 4B) devices and one laptop computer, which are all interconnected via Wi-Fi in an office building. The laptop computer has an aggregator and implements the AAFL algorithm of parameter server, and the Raspberry Pi device implements the AAFL algorithm of work node. All of these five work nodes have model training with local datasets and different simulated submission time, which represents a heterogeneous system with the different computational capabilities of work nodes.

*1) Baselines:* We compare with the following baseline approaches:

- Centralized SGD where the entire training dataset is stored on a single edge device and the model is trained directly on that device using a standard (centralized) SGD procedure.
- Canonical federated learning approach which is equivalent to using FedAvg algorithm and the fixed (non-adaptive) value of $\tau_i$ at all work nodes in all rounds.

For a fair comparison, we set the result of the first baseline as the standard model under our fixed time budget, and use this model for evaluating the convergence of other trained models. After the time budget is set, we implement the estimation of time consumption for all baselines, and the training stops when we have reached the time budget. When the AAFL algorithm is compared to the second baseline, we give the fixed $\tau_i$ three special values: The maximum, average and minimum number of local SGD iterations after the AAFL algorithm runs under the same condition at all work nodes in all rounds. When using the fixed $\tau_i$, we remove any parts related to the parameter estimations and recomputation of $\{\tau_i^*\}$ in Algorithm 2 and 3.

*2) Model and Datasets:* We evaluate the training of two different models on two different datasets, which represent both small and large models and datasets, as one can expect all these variants existing in edge computing scenarios. The models include squared-SVM[1] (we refer to as SVM in short in the following) and deep convolutional neural networks (CNN).[2] Among them, the loss functions for SVM satisfy Assumption 1, whereas the loss functions for CNN are non-convex and thus do not satisfy Assumption 1. Unless otherwise specified, the datasets have data distribution that each data sample is randomly assigned to a work node (defined as **Case 1**), thus each work node has uniform (but not full) information.

SVM is trained on the original MNIST (which we refer to as MNIST in short in the following) dataset [41], which contains the gray-scale images of $7 \times 10^4$ handwritten digits ($6 \times 10^4$ for training and $10^4$ for testing).

CNN is trained using SGD on two different datasets: the MNIST dataset and the CIFAR-10 dataset [42], and the CIFAR-10 dataset includes $6 \times 10^4$ color images ($5 \times 10^4$ for training and $10^4$ for testing) associated with a label from 10 classes. A separate CNN model is trained on each dataset, to perform multi-class classification among the 10 different labels in the dataset.

*3) Simulation of Time Consumption:* For the prototype system, we train each model for a fixed amount of time budget. The values of $c$ and $b$ correspond to the simulated time used for each local SGD iteration and global aggregation, respectively. The simulated time values are randomly generated according to Gaussian distribution, where the mean and standard deviation values are obtained from the real-time consumption of the SVM model on the prototype system.

*4) Simulation of Heterogeneous System:* In real situations, the heterogeneous computing power and network connection of edge devices cause different submission times which is designed by adding delay time to the simulated time $c$. The delay time are multipliers of the simulated time, and the multipliers are divided into two intervals: $[1, 2]$ and $(2, 10]$. If any work node selects one of the above two delay time intervals, it will be added with a random delay time to all local SGD iterations in the selected interval. We note that the random delay time has a small variance in the interval of $[1, 2]$ (stable) and have a large variance in the interval of $(2, 10]$ (unstable). Therefore, the work nodes in the delay time interval of $[1, 2]$ is the stable fast work nodes which have stable short time in training the model (local SGD) and submitting the results, respectively, the work nodes in the delay time interval of $(2, 10]$ is the unstable slow work nodes that have unstable long time in training the model (local SGD) or submitting the results.

At the beginning of each round in experiments, we divide all these work nodes into the above two delay time intervals to represent fast work nodes and slow work nodes. According to the percentage of fast work nodes, the comparison trials were divided into six groups, from 0% to 100%, with an interval of 20%. Moreover, We divide the delay time interval of $(2, 10]$ into four independent parts of the same size in Section V-B2, each representing the delay time interval of stable slow work nodes, and then observe the effect of different delay time intervals on the AAFL algorithm. The following results in Section V-B show that the AAFL algorithm can handle these fast and slow work nodes correctly.

*5) Training and Control Parameters:* In all our experiments, we set the maximum $\tau_i$ value $\max \tau_i = 100$ according to the boundary of $\tau_i$ value in Section IV-C2. Unless otherwise specified, we set the control parameter $\omega = 5 \times 10^{-5}$ for SVM, $\omega = 5 \times 10^{-3}$ for CNN on MNIST dataset and $\omega = 5 \times 10^{-4}$ for CNN on CIFAR-10 dataset, these parameters were manually selected based on our experimental results in Section V-B6. We manually select the learning rate fixed at $\eta = 0.01$, which is acceptable for our learning process. Unless otherwise specified, the time budget is set as $T = 15$ seconds based on our experimental results in Section V-B3.

### *B. Results*

In the above experimental setup, we first analyze the convergence of the trained models and the delay time intervals of the slow work nodes in the prototype system. Then we single out the SVM model and the MNIST dataset for experiments on 80% fast work nodes. These experiments show the impact of initial $\{\tau_i\}$, the number of work nodes, straggles, $\omega$ and instantaneous behavior on the AAFL algorithm. Except for the instantaneous results in Section V-B7, the others are the average results of 10 independent experiment/simulation runs.

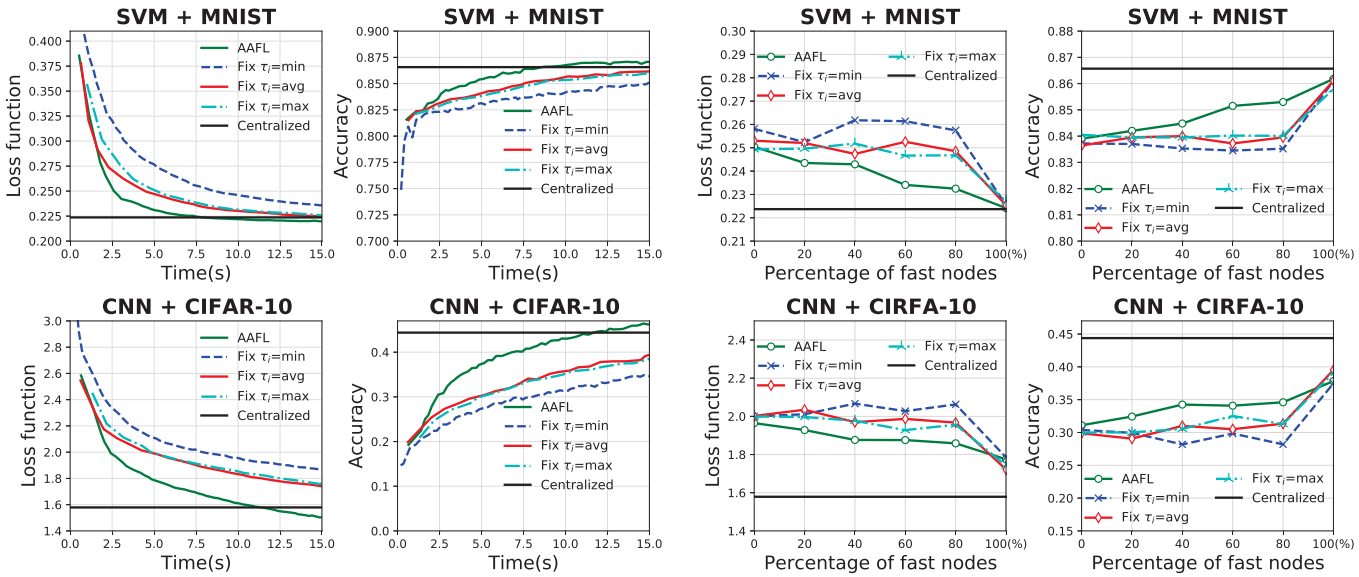
*1) Loss and Accuracy Values:* In our first set of experiments, the SVM and CNN models are trained on the prototype system with time consumptions generated in the way described in Section V-A3.

We record the values of loss function and classification accuracy on the SVM and CNN classifiers with the AAFL algorithm (with adaptive $\{\tau_i\}$), and compare them to baseline approaches, where the results are shown in Fig. 4. Additional results for CNN (MNIST) are included in Appendix D. We set time budget $T = 15$ seconds in Fig. 4a, and it shows the variation of loss function values and classification accuracy between the AAFL algorithm and the second baseline with the different fixed values of $\tau_i$ with 80% fast work nodes. The centralized case (first baseline) only has one optimal value as the training result, and we show a flat line across different time for the ease of comparison. The above percentage of fast work nodes represent the main reality scenario that fast work nodes are the majority. We note that the AAFL algorithm has convergence and performs close to the optimal point for all time and all models. In some cases, the AAFL algorithm can perform better than the centralized approach, because for a given amount of time budget, the AAFL algorithm is able to make the effective use of the computation resource at heterogeneous multiple work nodes, then get more total number of SGD iterations.

It can be observed from the Fig. 4a that the great difference between the AAFL algorithm and the fixed values of $\tau_i$ occurs

[1]The squared-SVM has a fully connected neural network, and outputs a binary label that corresponds to whether the digit is even or odd.

[2]The CNN has two $5 \times 5 \times 32$ convolution layers, two $2 \times 2$ MaxPoll layers, a $1568 \times 256$ fully connected layer, a $256 \times 10$ fully connected layer, and a softmax output layer with 10 units.




(a) Loss and accuracy with $80\%$ fast work nodes (b) Final loss and accuracy from $0\%$ to $100\%$ fast nodes

Fig. 4. Loss function and classification accuracy with the different percentage of fast work nodes. The curves show the results from the AAFL algorithm and the baseline with the different fixed values of $\tau_i$.

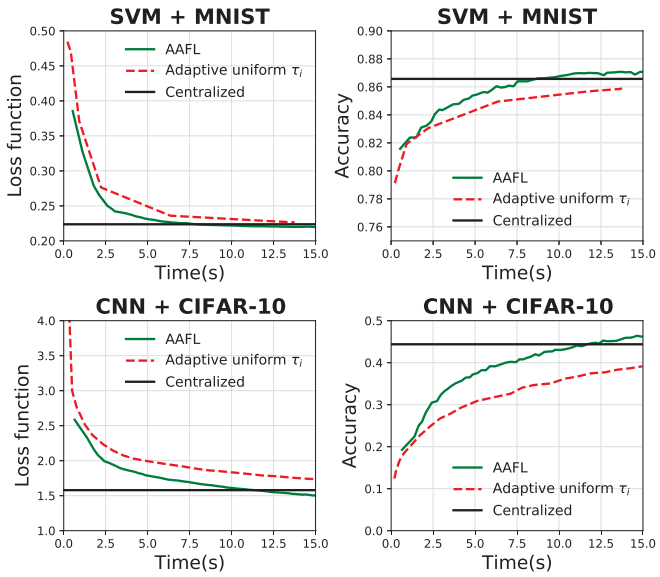



Fig. 5. Loss function and classification accuracy between the AAFL algorithm and the algorithm in [15].

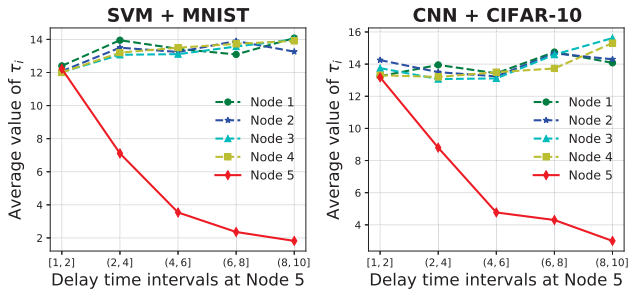



Fig. 6. Value of $\tau_i$ with different delay time intervals of slow work node.

around the fifth second. Thus, we set the time budget $T = 5$ in Fig. 4b to compare the performance of the AAFL algorithm and the fixed values of $\tau_i$ at the different percentages of fast work nodes. As shown in Fig. 4b, the AAFL algorithm has convergence and performs close to the optimal point for all models and almost all percentages (except 0% and 100%) of fast work nodes. The exception is because the AAFL algorithm degrades into FedAvg algorithm when the heterogeneity of the prototype system fades away at the 0% and 100% fast work nodes.

We also compare the AAFL algorithm to the algorithm which uses adaptive uniform $\tau$ (synchronous iteration strategy) [15], which means that the value of $\tau$ is uniform at all work nodes $i$ and adjusted in each communication round. The results are shown in Fig. 5. Also, we set the time budget $T = 15$ and 80% fast work nodes for ease of comparison. We note that both of them have convergence, but the AAFL algorithm has the fastest convergence rate and highest classification accuracy of model for all cases. The results demonstrate that our asynchronous iteration strategy performs better than above synchronous iteration strategy in the heterogeneous edge device system.

*2) Varying Delay Time Intervals of Slow Work Nodes:* We evaluate the impact of the slow work nodes with four independent intervals of delay time (Section V-A4) on the prototype system ($T = 15$ seconds). The five work nodes in the prototype system are divided into four fast work nodes and one slow woke node, and all work nodes train SVM model on MNIST dataset or train CNN model on CIFAR-10 dataset.

The results are shown in Fig. 6. We note that there are five delay time intervals, work nodes in the delay time interval of [1, 2] represents the stable fast work nodes and the delay time interval of $(2, 10]$ is divided into four small parts of

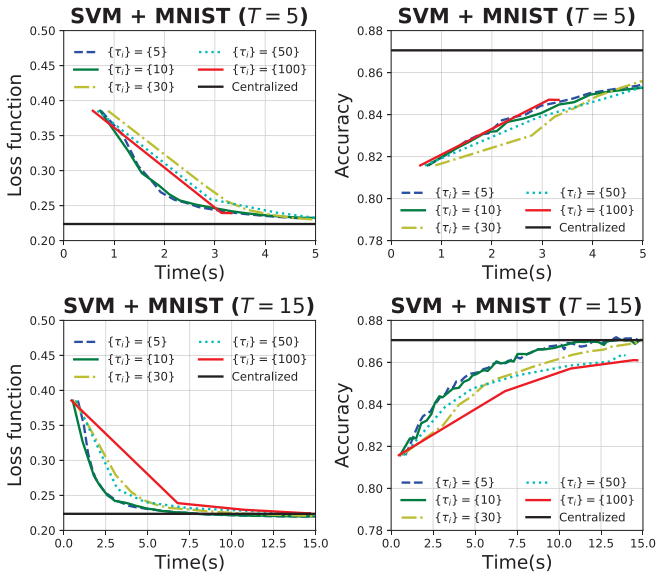



Fig. 7. Loss function and classification accuracy with different values of initial $\{\tau_i\}$.

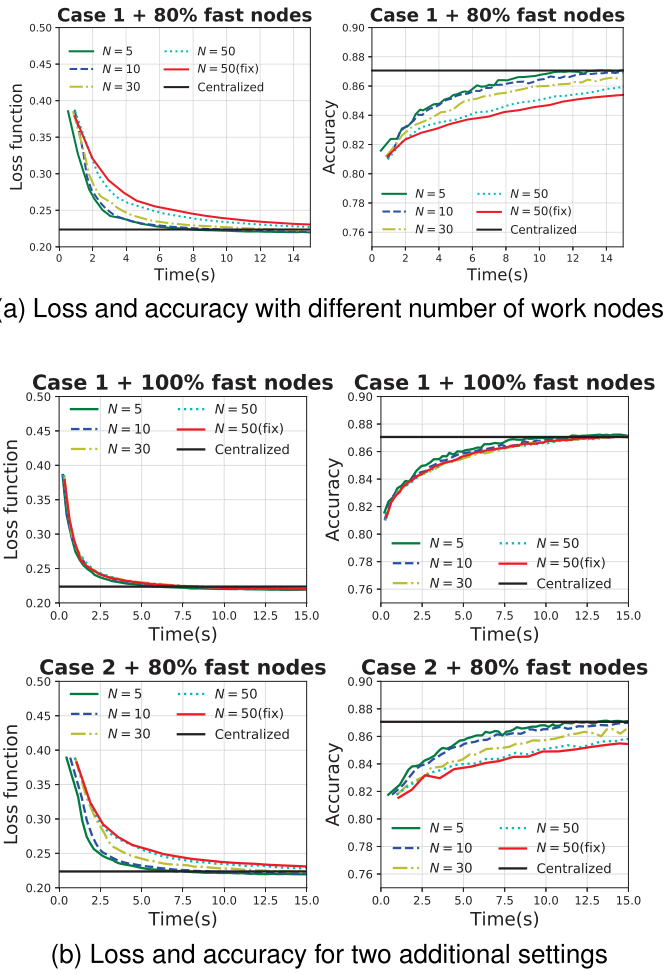



Fig. 8. Loss function and classification accuracy (SVM + MNIST) with the different number of work nodes.

the same size (more stable changes). All fast work nodes are in the delay time interval of [1, 2], the slow work node is selected as variables in the above five delay time intervals. By comparing $\tau_5$ of slow work node at the different delay time intervals, we find that the value of $\tau_5$ decreases with increasing delay and gradually approaches the minimum value of $\tau_5 = 1$. In addition, $\tau_1$ to $\tau_4$ of fast work nodes increase slightly as $\tau_5$ of slow work nodes decreases. The reason for this result is that when the delay of the slow work nodes has an increasing impact on the FL system, the AAFL algorithm will reduce the training frequency of the slow work nodes to make the whole FL system balanced, that is, the fast work nodes do not have to wait for the slow work nodes, which makes the model converge faster within the time budget $T$.

Although the results are presented for the full nodes participation setting, and the delay of slow work nodes would exceed the above range in practice, we can extend the AAFL algorithm to the case where a subset of nodes are sampled in each round to solve these problems [6].

Due to the high complexity of evaluating CNN models, we focus on the SVM model with the MNIST dataset in the following and provide further insights on the prototype system.

*3) Varying Value of Initial $\{\tau_i\}$:* As discussed in Section IV-B, our new resulting values of $\{\tau_i^*\}$ are always in the vicinity of the initial value of $\{\tau_i\}$. The effect of the initial value of $\{\tau_i\}$ for selection on the final results is shown in Figure 7, where there are two different time budgets with 80% fast work nodes. We find that the model has fast convergence speed and high classification accuracy when initial $\{\tau_i\}$ is $\{5\}$ or $\{10\}$, and the convergence speed and classification accuracy decrease with increasing $\{\tau_i\}$ at $T = 5$ and $T = 15$. This is because the larger initial value of $\tau_i$, the greater impact of slow work nodes brings, which reduces the convergence speed. Thus, when the time budget is small ($T = 5$) and the initial value of $\{\tau_i\}$ is large ($\{30\}$, $\{50\}$ and $\{100\}$), the curves in the loss function and classification accuracy are close to straight lines, because the step $k = 0$ in Algorithms 2 and 3 consumes the major time budget. Therefore, we set that the initial $\{\tau_i\} = \{10\}$ and the time budget $T = 15$ in subsequent experiments.

*4) Varying Number of Work Nodes:* The number of work nodes varying from 5 to 50, and the results of the loss function and classification accuracy are shown in Fig. 8a, which are obtained in a simulated environment that we allow the Algorithm 3 to run multiple times in parallel at each work node. The AAFL algorithm performs stably as the number of work nodes increases within the set of budget $T = 15$ and 80% fast work nodes. We note that the convergence rate and classification accuracy of model decrease as the number of work nodes increases, because the number of slow work nodes increases and the size of the dataset at each work node decreases.

In order to find out what kind of factors have the greatest influence on the experimental results, we add two settings of comparison experiments: **Case 1** with 100% fast work nodes and **Case 2** with 80% fast work nodes, where **Case 1** is defined in Section V-A2 and **Case 2** represents each work node has the entire dataset (thus full information). Results are shown in

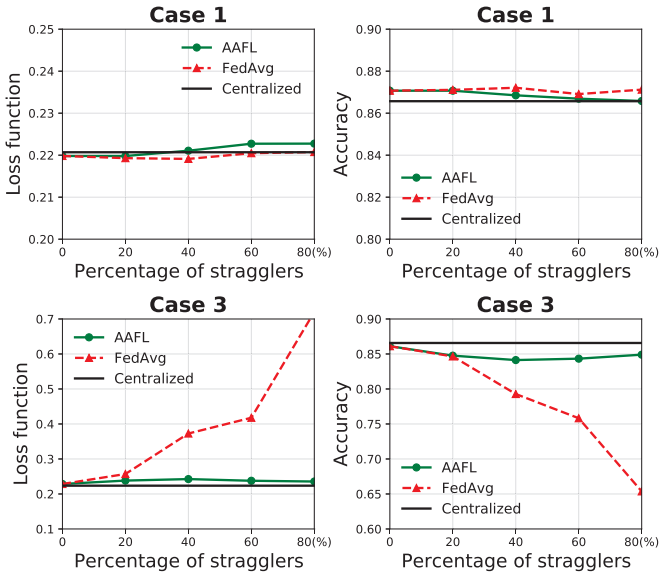



Fig. 9. Loss function and classification accuracy (SVM + MNIST) with the different number of stragglers.

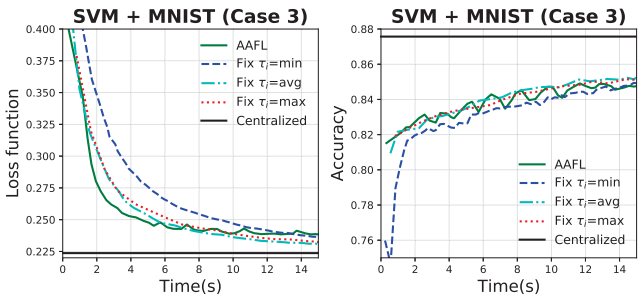



Fig. 10. Loss function and the classification accuracy in Case 3.

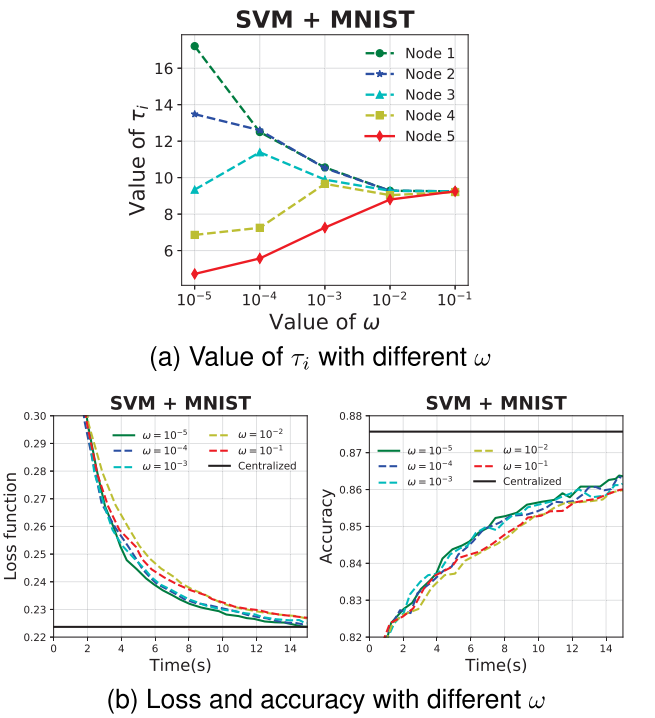



Fig. 11. Impact of $\omega$ on the average value of $\tau_i$, and loss function and classification accuracy with the different values of $\omega$.

Fig. 8b. In **Case 1** with 100% fast work nodes, the different number of work nodes have less impact on the convergence speed and classification accuracy. In addition, there is almost the same convergence speed and classification accuracy for the different number of work nodes between **Case 2** with 80% fast work nodes and **Case 1** with 80% fast work nodes. Thus, as the number of work nodes increases, the effect of slow work nodes significantly reduces the convergence speed and classification accuracy.

We also fix $\tau_i$ equal to the average value which the AAFL algorithm performs in all the above situations with 50 work nodes (second baseline), and the result shows that the AAFL algorithm performs better even when slow work nodes have a huge impact.

*5) Varying Number of Stragglers:* The FedAvg algorithm drops stragglers, thus resulting in missing information when there is system heterogeneity [6]. We assume that the slow work node is the straggler, and our AAFL algorithm does not drop stragglers. The influence of dropping stragglers is associates with the data distribution, thus, we set two **Case** to represent the data distribution: IID and Non-IID. The dataset in Section V-A2 is set for IID as **Case 1**, and we set all the data samples in each work node to have the same label (Non-IID) as **Case 3**. For better comparison, we set the 0–80% of stragglers which correspond to the 20–100% of fast work nodes. As shown in Fig. 9, we set the time budget $T = 15$, and the AAFL algorithm performs as well as the FedAvg in **Case 1**, but it has the distinctive gap between them in **Case 3**. The gap shows that the AAFL algorithm has better convergence by the adaptive $\{\tau_i^*\}$ instead of dropping stragglers which deteriorates the convergence of FedAvg with Non-IID data.

We also compare the loss function values and the classification accuracy of the AAFL algorithm to baseline approaches(not dropping stragglers) for the SVM (MNIST) in **Case 3**. We set the 80% stragglers, and the average results are shown in Fig. 10. We find that our AAFL algorithm still has a faster convergence rate in **Case 3**, and the final loss function values and classification accuracy are essentially the same as the 20% fast work nodes in **Case 1** (see Figure 4b).

*6) Sensitivity of $\omega$:* The sensitivity of the control parameter $\omega$ evaluated on the prototype system is shown in Fig. 11. Fig. 11a shows the average value of $\tau_i$ in the AAFL algorithm at each work node $i$ with the time budget $T = 15$. As for more visual observation, we set that the five work nodes have their uniform delay time interval which is one of the five intervals in Section V-B2. The simulated time of five work nodes, in order from smallest to largest, are Node 1, Node 2, Node 3, Node 4 and Node 5. We see that the slowest nodes have the small value of $\tau$, and the gap between $\tau_i$ decreases approximately linearly with $\log\omega$ and degrades into FedAvg algorithm at $\omega = 0.1$. When $\omega = 0.1$, all the fast work nodes and slow work nodes have the same value of local iterations that bring less global aggregation times. This is because the initial value of model

parameters is close to the optimal value where the model has no more training by the definition of $\omega$.

Fig. 11b shows the average value of loss function and classification accuracy in the AAFL algorithm with the different values of $\omega$, time budget $T = 15$ and 80% fast work nodes on the prototype system. We see that the small changes of $\omega$ does not change the value of loss function and classification accuracy much, indicating that one can take big steps when tuning $\omega$ in practice and the tuning is not difficult.

*7) Instantaneous Results:* We finally study the instantaneous behavior of the AAFL algorithm for a single run with the time budget $T = 15$ and 80% fast work nodes on the prototype system. Results for SVM (MNIST) is shown in Fig. 12, the red line represents the slowest work node (Node 5) in the delay time interval of $(2, 10]$, and the rest are the fast work nodes (Node 1-4) in the delay time interval of $[1, 2]$. There is only one curve for the $b$-value record, which is because all work nodes share a global aggregation time.

We see that the simulated value of $c_i$ at Node 5 changes dramatically over time, representing unstable performance, while on other work nodes the opposite is true. It is consistent with the description we mentioned in Section V-A4. Also, the simulated value of $b$ is unstable over time, where it represents our simulation of the heterogeneous system. Moreover, we find that The value of $\tau_5$ and $c_5$ (Node 5) has an opposite relationship. It represents that the slowest work node has a small value of $\tau$, and is the same as what we mentioned in Section V-B6.

The values of $\delta_i, \beta_i$ and $\rho_i$ are parameters to recompute $\{\tau_i^*\}$. By their definitions in Section IV, $\delta_i$ and $\beta_i$ are related to the gradients of the model on the training dataset, and the data samples on different work nodes are usually different, so $\delta_i$ and $\beta_i$ are consistently varied. In addition, the value of $\rho_i$ is related to the model loss function, and as the global model converges, the difference between the loss functions becomes small, and the value of $\rho_i$ at each work node reduces over time.

### C. Applications

We also apply the FedAvg algorithm and the AAFL algorithm on CNN-SVM model [43] with six public COVID-19 X-ray images databases ($D_a, D_b, D_c, D_d, D_e, D_f$),[3] and evaluate the performance under the situation of **Case 1** with 80% fast work nodes on the prototype system. The CNN-SVM model is CNN network with SVM output layer, and see [43] for the definitions of loss function and accuracy. These images are fed to CNN deep learning model followed by SVM classifier with the identification of 'Normal' or 'COVID-19'. It can learn with not much data, thus is more suitable to run on our prototype system. In databases $D_b$ and $D_f$, because there is no test dataset, we use the validation dataset to replace the test dataset. As for $D_a$, $D_c$, and $D_d$, we use the 10-fold cross-validation strategy.

[3]The six public databases can be accessed from these links: $D_a$:www.kaggle.com/amanullahasraf/covid19-pneumonia-normal-chest-xray-pa-dataset $D_b$: www.kaggle.com/fusicfenta/chest-xray-for-covid19-detection $D_c$: www.kaggle.com/tawsifurrahman/covid19-radiography-database $D_d$: www.kaggle.com/tarandeep97/covid19-normal-posteroanteriorpa-xrays $D_e$: www.kaggle.com/khoongweihao/covid19-xray-dataset-train-test-sets $D_f$: www.kaggle.com/darshan1504/covid19-detection-xray-dataset

TABLE II
PERFORMANCE ON VARIOUS COVID-19 DATASETS WITHIN FIXED TIME

| Dataset | Algorithm | Optimal Accuracy (%) | Loss | Accuracy (%) |
|---|---|---|---|---|
| $D_a$ | FedAvg | 96.23 | 0.21 | 84.38 |
| | **AAFL** | | **0.11** | **95.15** |
| $D_b$ | FedAvg | 95.56 | 0.27 | 92.93 |
| | **AAFL** | | **0.19** | **94.14** |
| $D_c$ | FedAvg | 96.36 | 0.22 | 93.13 |
| | **AAFL** | | **0.15** | **94.86** |
| $D_d$ | FedAvg | 93.03 | 0.26 | 87.79 |
| | **AAFL** | | **0.21** | **90.51** |
| $D_e$ | FedAvg | 88.44 | 0.51 | 82.00 |
| | **AAFL** | | **0.36** | **85.21** |
| $D_f$ | FedAvg | 82.28 | 0.60 | 75.34 |
| | **AAFL** | | **0.49** | **79.11** |

*1) Performance:* In this set of experiments, we fix the time to 500 seconds and record the loss and accuracy values of the models trained by the AAFL algorithm and the FedAvg algorithm on all datasets. We choose the time point of 500 seconds because the time required for all models to complete training is around 1000 seconds. The difference between the AAFL algorithm and the FedAvg algorithm is pronounced at around halfway through training according to what is shown by Section V-B1. Then, we conduct 10 independent experiments with the AAFL algorithm on each dataset and average the results.

As shown in Table II, Optimal accuracy is the optimal classification accuracy obtained using centralized SGD to measure whether the model converges. The difference is relatively large between the models obtained by the AAFL algorithm and the FedAvg algorithm, where the models are compared by both loss and accuracy values at the time node of 500 seconds. Meanwhile, by comparing with optimal accuracy, the accuracy of models obtained by the AAFL algorithm is closer to the optimal accuracy than Fedavg algorithm, although all models did not converge. Thus, the AAFL algorithm converges faster than Fedavg algorithm on all these COVID-19 X-ray images datasets within the time budget. To further prove this conclusion, we conducted the next experiment.

*2) Convergence Rate:* To further study the AAFL algorithm, we do not impose any constraint on time to analyze the convergence speed of the AAFL algorithm and FedAvg algorithm. The quantitative metric tested is the time consumption for each model to converge to near the optimum (optimal classification accuracy).

The average results of 10 independent experiment runs are shown in Table III. We note that the optimal accuracy of the CNN-SVM model is not high on datasets $D_e$ and $D_f$, which is due to the inherent characteristics of the model and dataset, so we still use it as the convergence criterion. We can see that both the AAFL algorithm and FedAvg algorithm converge without time constraints, but the AAFL algorithm takes less time to reach the optimum than FedAvg algorithm in all six public datasets. Moreover, in the case of the largest difference, the AAFL algorithm takes only about 67% time consumption

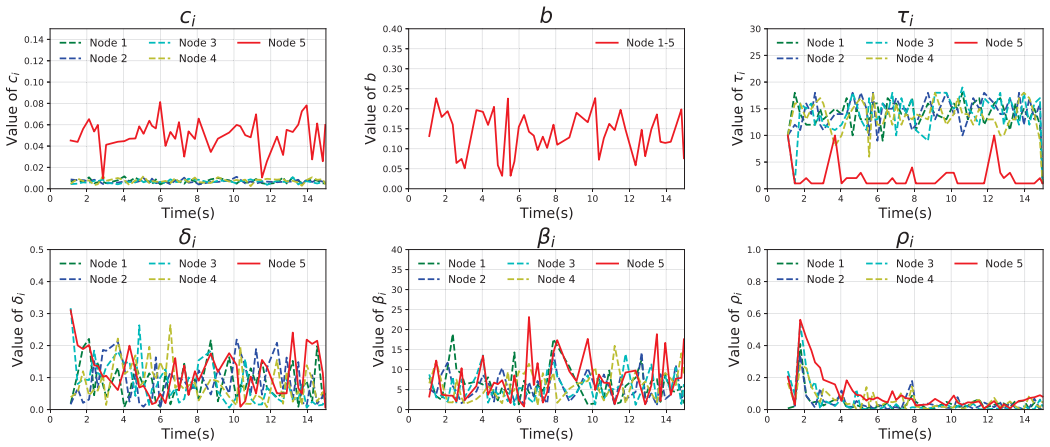



Fig. 12. Instantaneous results of a single run (SVM + MNIST) with the AAFL algorithm.

TABLE III
TIME CONSUMPTION ON DIFFERENT COVID-19 DATASET

| Dataset | Accuracy (%) | Algorithm | Time Consumption (seconds) |
|---|---|---|---|
| $D_a$ | 96.23 | FedAvg | 1194.93 |
| | | **AAFL** | **744.42** |
| $D_b$ | 95.56 | FedAvg | 1010.81 |
| | | **AAFL** | **681.17** |
| $D_c$ | 96.36 | FedAvg | 1179.02 |
| | | **AAFL** | **978.05** |
| $D_d$ | 93.03 | FedAvg | 938.92 |
| | | **AAFL** | **887.84** |
| $D_e$ | 88.44 | FedAvg | 867.01 |
| | | **AAFL** | **637.56** |
| $D_f$ | 82.28 | FedAvg | 1151.16 |
| | | **AAFL** | **957.43** |

of the FedAvg algorithm to complete the task. It shows that the AAFL algorithm in the heterogeneous prototype system has significant convergence rate gains over FedAvg for the selected datasets and model.

Considering the overall experiments and applications, we can conclude that the AAFL scheme provides an efficient performance for countering communicable infectious diseases (COVID-19) in Edge-Based wireless medical system.

## VI. Conclusion and the Future Work

In order to improve the efficiency of federated learning in the heterogeneous Edge-Based wireless communication systems for countering the communicable infectious disease of COVID-19, this paper proposed an AAFL scheme that controls the number of local SGD iterations on medical devices to obtain the fast convergence of training models. We analyze the mathematical relationships between the number of local SGD iterations and the optimal model parameters of FL, and design an adaptive control algorithm from this relationship to minimize the loss function under a time budget constraint. Our experimental results and application performance confirmed the effectiveness of our AAFL scheme. In the future, we will explore how AAFL performs in the various distribution of the COVID-19 data, and assign more effective weights for updating the global model based on the number and feature of samples per work node.

## Appendix A
## Distributed vs. Centralized SGD

*Proposition 1: When $\tau = 1$, Algorithm 1 yields the following recurrence relation for $w_k$:*

$$\boldsymbol{w}_k = \boldsymbol{w}_{k-1} - \eta \nabla F\left(\boldsymbol{w}_{k-1}\right). \tag{24}$$

*Proof:* When $\tau = 0$, we have $\boldsymbol{w}_k^{i,\lambda=0} = \boldsymbol{w}_{k-1}$ for all $k$ and $i$. Thus,

$$\begin{aligned}
\boldsymbol{w}_k &= \frac{\sum_{i=1}^{N} D_i \boldsymbol{w}_{\boldsymbol{k}}^{i,\lambda=\mathbf{1}}}{D} \\
&= \frac{\sum_{i=1}^{N} D_i \left(\boldsymbol{w}_k^{i,\lambda=0} - \eta \nabla F_i \left(\boldsymbol{w}_k^{i,\lambda=0}\right)\right)}{D} \\
&= \frac{\sum_{i=1}^{N} D_i \boldsymbol{w}_{k-1}}{D} - \frac{\sum_{i=1}^{N} D_i \eta \nabla F_i\left(\boldsymbol{w}_{k-1}\right)}{D} \\
&= \boldsymbol{w}_{k-1} - \eta \nabla F\left(\boldsymbol{w}_{k-1}\right),
\end{aligned}$$

where the second term in the last equality is because

$$\begin{aligned}
\frac{\sum_{i=1}^{N} D_i \eta \nabla F_i\left(\boldsymbol{w}_{k-1}\right)}{D} &= \eta \nabla \frac{\sum_{i=1}^{N} D_i F_i\left(\boldsymbol{w}_{k-1}\right)}{D} \\
&= \eta \nabla F\left(\boldsymbol{w}_{k-1}\right).
\end{aligned}$$

The last equation is due to the linearity of the gradient operator. □

We note that (24) is the recurrence relation for centralized SGD on the global loss $F(\boldsymbol{w})$. Therefore, the distributed SGD algorithm presented in Algorithm 1 is logically equivalent to centralized SGD for $\tau = 1$. The proof of Theorem 1 is presented in the next section.

## Appendix B
## Proof of Theorem 1

To prove Theorem 1, we first introduce the following lemma.

*Lemma 2: For any $k$, $i$ and $\lambda \in [0, \tau_i]$, we have*

$$\left\| \boldsymbol{w}_k^{i,\lambda} - \boldsymbol{v}_k^{i,\lambda} \right\| \leq g_i(\lambda),$$

*where we define the function $g_i(\lambda)$ as*

$$g_i(\lambda) \triangleq \frac{\delta_i}{\beta_i} \left( (\eta \beta_i + 1)^{\lambda} - 1 \right).$$

*Proof:* We show by induction that $\left\|\boldsymbol{w}_k^{i,\lambda}-\boldsymbol{v}_k^{i,\lambda}\right\| \leq g_i(\lambda)$ for any $k$ and all $\lambda \in [0,\tau_i]$.

When $\lambda = 0$, we know that $\boldsymbol{v}_k^{i,\lambda=0} = \boldsymbol{w}_k^{i,\lambda=0}$ by the definition of $\boldsymbol{v}_k^{i,\lambda}$, and we have $\left\|\boldsymbol{w}_k^{i,\lambda=0}-\boldsymbol{v}_k^{i,\lambda=0}\right\| = g_i(0)$. Thus, for the induction with $\lambda \in [1,\tau_i+1]$, we assume that

$$\left\|\boldsymbol{w}_k^{i,\lambda-1}-\boldsymbol{v}_k^{i,\lambda-1}\right\| \leq g_i(\lambda-1). \tag{25}$$

We now show that $\left\|\boldsymbol{w}_k^{i,\lambda}-\boldsymbol{v}_k^{i,\lambda}\right\| \leq g_i(\lambda)$ holds for $\lambda$. We have

$$\begin{aligned}
&\left\|\boldsymbol{w}_k^{i,\lambda}-\boldsymbol{v}_k^{i,\lambda}\right\| \\
&= \left\|\boldsymbol{w}_k^{i,\lambda-1}-\eta\nabla F_i\left(\boldsymbol{w}_k^{i,\lambda-1}\right)-\boldsymbol{v}_k^{i,\lambda-1}+\eta\nabla F\left(\boldsymbol{v}_k^{i,\lambda-1}\right)\right\| \\
&= \| \left(\boldsymbol{w}_k^{i,\lambda-1}-\boldsymbol{v}_k^{i,\lambda-1}\right)-\eta\left(\nabla F_i\left(\boldsymbol{w}_k^{i,\lambda-1}\right)\right. \\
&\qquad \left.-\nabla F_i\left(\boldsymbol{v}_k^{i,\lambda-1}\right)\right) \\
&\quad -\eta\left(\nabla F_i\left(\boldsymbol{v}_k^{i,\lambda-1}\right)-\nabla F\left(\boldsymbol{v}_k^{i,\lambda-1}\right)\right)\| \\
&\text{(adding a zero term } \left[\eta\nabla F_i\left(\boldsymbol{v}_k^{i,\lambda-1}\right)-\eta\nabla F_i\left(\boldsymbol{v}_k^{i,\lambda-1}\right)\right], \text{ and rearranging)} \\
&\leq \left\|\boldsymbol{w}_k^{i,\lambda-1}-\boldsymbol{v}_k^{i,\lambda-1}\right\|+\eta\left\|\nabla F_i\left(\boldsymbol{w}_k^{i,\lambda-1}\right)-\nabla F_i\left(\boldsymbol{v}_k^{i,\lambda-1}\right)\right\| \\
&\qquad +\eta\left\|\nabla F_i\left(\boldsymbol{v}_k^{i,\lambda-1}\right)-\nabla F\left(\boldsymbol{v}_k^{i,\lambda-1}\right)\right\| \\
&\text{(from triangle inequality)} \\
&\leq (\eta\beta_i+1)\left\|\boldsymbol{w}_k^{i,\lambda-1}-\boldsymbol{v}_k^{i,\lambda-1}\right\|+\eta\delta_i \\
&\text{(from the Definition 1 and } \beta \text{ -smoothness of Assumption 1)} \\
&\leq \frac{\delta_i}{\beta_i}(\eta\beta_i+1)\left((\eta\beta_i+1)^{\lambda-1}-1\right)+\eta\delta_i \\
&\text{(from the assumption in (25))} \\
&= \frac{\delta_i}{\beta_i}\left((\eta\beta_i+1)^{\lambda}-1\right).
\end{aligned}$$

Using the above induction, we have shown that $\left\|\boldsymbol{w}_k^{i,\lambda}-\boldsymbol{v}_k^{i,\lambda}\right\| \leq g_i(\lambda)$ for any $k$ and all $\lambda \in [0,\tau_i]$. □

We are now ready to prove Theorem 1.

*Proof of Theorem 1:* From the definition of $g_i(\lambda)$ in Lemma 2 and the $\rho$-Lipschitz in Assumption 1, for any $k \in [1,K]$, we have

$$\begin{aligned}
&F(\boldsymbol{w}_k)-F(\boldsymbol{v}_k) \\
&= \frac{\sum_{i=1}^N D_i F_i\left(\boldsymbol{w}_{k-1}^{i,\lambda=\tau_i}\right)}{D}-\frac{\sum_{i=1}^N D_i F_i\left(\boldsymbol{v}_{k-1}^{i,\lambda=\tau_i}\right)}{D} \\
&= \frac{\sum_{i=1}^N D_i\left(F_i\left(\boldsymbol{w}_{k-1}^{i,\lambda=\tau_i}\right)-F_i\left(\boldsymbol{v}_{k-1}^{i,\lambda=\tau_i}\right)\right)}{D} \\
&\leq \frac{\sum_{i=1}^N D_i\rho_i\left\|\boldsymbol{w}_{k-1}^{i,\lambda=\tau_i}-\boldsymbol{v}_{k-1}^{i,\lambda=\tau_i}\right\|}{D} \\
&\leq \frac{\sum_{i=1}^N D_i\rho_i g_i(\tau_i)}{D}.
\end{aligned}$$

□

The proof of Lemma 1 is presented in the next section.

## APPENDIX C
## PROOF OF LEMMA 1

To prove Lemma 1, we first introduce some additional definitions and lemmas.

*Definition 2: For a round $k$ and work node $i$, we define $\theta_k^{i,\lambda} = F\left(\boldsymbol{v}_k^{i,\lambda}\right) - F(\boldsymbol{w}^*)$, $\lambda$ is defined between $0 \leq \lambda \leq \tau_i$.*

According to the convergence lower bound of SGD given in [44], Therem 3.14, we always have

$$\theta_k^{i,\lambda} > 0, \tag{26}$$

for any finite $\lambda$, $k$ and $i$.

*Lemma 3: For any $k$, $i$, and $\lambda \in [0,\tau_i-1]$, we have*

$$F_i\left(\boldsymbol{v}_k^{i,\lambda+1}\right)-F_i\left(\boldsymbol{v}_k^{i,\lambda}\right) \leq -\eta\left(1-\frac{\eta\beta_i}{2}\right)\left\|\nabla F_i\left(\boldsymbol{v}_k^{i,\lambda}\right)\right\|^2. \tag{27}$$

*Proof:* Because $F_i(.)$ $\beta$-smooth, we have

$$\begin{cases}
0 \leq f(\boldsymbol{x})-f(\boldsymbol{y})-\nabla f(\boldsymbol{y})^T(\boldsymbol{x}-\boldsymbol{y}) \leq \dfrac{\beta}{2}\|\boldsymbol{x}-\boldsymbol{y}\|^2 \\
f(\boldsymbol{x})-f(\boldsymbol{y}) \leq \nabla f(\boldsymbol{x})^T(\boldsymbol{x}-\boldsymbol{y})-\dfrac{1}{2\beta}\|\nabla f(\boldsymbol{x})-\nabla f(\boldsymbol{y})\|^2,
\end{cases}$$

for arbitrary $\boldsymbol{x}$ and $\boldsymbol{y}$ [44], where $\nabla f(.)^T$ is the transpose of $\nabla f(.)$. Thus,

$$\begin{aligned}
&F_i\left(\boldsymbol{v}_k^{i,\lambda+1}\right)-F_i\left(\boldsymbol{v}_k^{i,\lambda}\right) \\
&\leq \nabla F_i\left(\boldsymbol{v}_k^{i,\lambda}\right)^T\left(\boldsymbol{v}_k^{i,\lambda+1}-\boldsymbol{v}_k^{i,\lambda}\right)+\frac{\beta_i}{2}\left\|\boldsymbol{v}_k^{i,\lambda+1}-\boldsymbol{v}_k^{i,\lambda}\right\|^2 \\
&= \nabla F_i\left(\boldsymbol{v}_k^{i,\lambda}\right)^T\left(-\eta\nabla F_i\left(\boldsymbol{v}_k^{i,\lambda}\right)\right)+\frac{\eta^2\beta_i}{2}\left\|\nabla F_i\left(\boldsymbol{v}_k^{i,\lambda}\right)\right\|^2 \\
&= -\eta\left(1-\frac{\eta\beta_i}{2}\right)\left\|\nabla F_i\left(\boldsymbol{v}_k^{i,\lambda}\right)\right\|^2.
\end{aligned}$$

□

*Lemma 4: For any $k$, $i$, and $\lambda \in [0,\tau_i-1]$, we have*

$$\theta_k^{i,\lambda+1}-\theta_k^{i,\lambda} \leq -\eta\varphi_i\varepsilon^2, \tag{28}$$

*where $\omega = \min_{i,\lambda}\frac{1}{\left\|\boldsymbol{v}_k^{i,\lambda}-\boldsymbol{w}^*\right\|^2}$, and $\varphi_i \triangleq \left(1-\frac{\eta\beta_i}{2}\right)\omega$.*

*Proof:* By definition, $\theta_k^{i,\lambda} = F_i\left(\boldsymbol{v}_k^{i,\lambda}\right) - F(\boldsymbol{w}^*)$ and $\theta_k^{i,\lambda+1} = F_i\left(\boldsymbol{v}_k^{i,\lambda+1}\right) - F(\boldsymbol{w}^*)$. Substituting this into (27) in Lemma 3 yields.

$$\theta_k^{i,\lambda+1}-\theta_k^{i,\lambda} \leq -\eta\left(1-\frac{\eta\beta_i}{2}\right)\left\|\nabla F_i\left(\boldsymbol{v}_k^{i,\lambda}\right)\right\|^2. \tag{29}$$

The convexity condition gives

$$\begin{aligned}
\theta_k^{i,\lambda} &= F_i\left(\boldsymbol{v}_k^{i,\lambda}\right)-F(\boldsymbol{w}^*) \\
&\leq \nabla F_i\left(\boldsymbol{v}_k^{i,\lambda}\right)^T\left(\boldsymbol{v}_k^{i,\lambda}-\boldsymbol{w}^*\right) \\
&\quad -\frac{1}{2\beta_i}\left\|\nabla F_i\left(\boldsymbol{v}_k^{i,\lambda}\right)-\nabla F(\boldsymbol{w}^*)\right\|^2 \\
&\quad \text{(from the } \beta \text{ -smoothness in Lemma 3)} \\
&\leq \nabla F_i\left(\boldsymbol{v}_k^{i,\lambda}\right)^T\left(\boldsymbol{v}_k^{i,\lambda}-\boldsymbol{w}^*\right) \\
&\leq \left\|\nabla F_i\left(\boldsymbol{v}_k^{i,\lambda}\right)\right\|\left\|\boldsymbol{v}_k^{i,\lambda}-\boldsymbol{w}^*\right\|,
\end{aligned}$$

where the last inequality is from the Cauchy-Schwarz inequality. Hence,

$$\frac{\left(\theta_k^{i,\lambda}\right)^2}{\left\|(\boldsymbol{v}_k^{i,\lambda}-(\boldsymbol{w}^*\right\|^2} \leq \left\|\nabla F_i\left((\boldsymbol{v}_k^{i,\lambda}\right)\right\|^2. \tag{30}$$

Obviously, we have $\eta > 0$, and when $1-\frac{\eta\beta_i}{2} \geq 0$ , (30) is converted to

$$-\eta\left(1-\frac{\eta\beta_i}{2}\right)\left\|\nabla F_i\left(\boldsymbol{v}_k^{i,\lambda}\right)\right\|^2 \leq -\eta\left(1-\frac{\eta\beta_i}{2}\right)\frac{\left(\theta_k^{i,\lambda}\right)^2}{\left\|(\boldsymbol{v}_k^{i,\lambda}-(\boldsymbol{w}^*\right\|^2}. \tag{31}$$

Substituting (31) into (29), we get

$$\theta_k^{i,\lambda+1}-\theta_k^{i,\lambda} \leq -\eta\left(1-\frac{\eta\beta_i}{2}\right)\frac{\left(\theta_k^{i,\lambda}\right)^2}{\left\|(\boldsymbol{v}_k^{i,\lambda}-(\boldsymbol{w}^*\right\|^2}. \tag{32}$$

Recall that we defined $\omega = \min_{i,\lambda}\frac{1}{\left\|\boldsymbol{v}_k^{i,\lambda}-\boldsymbol{w}^*\right\|^2}$, and $\varphi_i \triangleq \left(1-\frac{\eta\beta_i}{2}\right)\omega$, and it is assumed that $F_i(\boldsymbol{v}_k^{i,\lambda})-F(\boldsymbol{w}^*) \geq \varepsilon$. Therefore, we have $\theta_k^{i,\lambda} = F_i\left(\boldsymbol{v}_k^{i,\lambda}\right)-F(\boldsymbol{w}^*) \geq \varepsilon$, and (32) is converted to

$$\theta_k^{i,\lambda+1}-\theta_k^{i,\lambda} \leq -\eta\varphi_i\varepsilon^2.$$

□

*Lemma 5: For any $k \in [1,K]$ and $i$, we have*

$$F(\boldsymbol{v}_k)-F(\boldsymbol{w}_{k-1}) \leq -\eta\varepsilon^2\frac{\sum_{i=1}^N D_i\varphi_i\tau_i}{D}. \tag{33}$$

*Proof:* Using Lemma 4 and considering $\lambda \in [0,\tau_i-1]$, we have

$$\begin{aligned}
&F_i\left(\boldsymbol{v}_k^{i,\lambda=\tau_i}\right)-F_i\left(\boldsymbol{v}_k^{i,\lambda=0}\right)\\
&=\theta_k^{i,\lambda=\tau_i}-\theta_k^{i,\lambda=0}\\
&=\sum_{\lambda_i=0}^{\tau_i-1}\theta_k^{i,\lambda+1}-\theta_k^{i,\lambda}\\
&\leq -\eta\varphi_i\tau_i\varepsilon^2.
\end{aligned}$$

Thus, for $k \in [1,K]$, we have

$$\begin{aligned}
&F(\boldsymbol{v}_k)-F(\boldsymbol{w}_{k-1})\\
&=\frac{\sum_{i=1}^N D_iF_i\left(\boldsymbol{v}_{k-1}^{i,\lambda=\tau_i}\right)}{D}-\frac{\sum_{i=1}^N D_iF_i(\boldsymbol{w}_{k-1})}{D}\\
&=\frac{\sum_{i=1}^N D_iF_i\left(\boldsymbol{v}_{k-1}^{i,\lambda=\tau_i}\right)}{D}-\frac{\sum_{i=1}^N D_iF_i\left(\boldsymbol{v}_{k-1}^{i,\lambda=0}\right)}{D}\\
&=\frac{\sum_{i=1}^N D_i\left(F_i\left(\boldsymbol{v}_{k-1}^{i,\lambda=\tau_i}\right)-F_i\left(\boldsymbol{v}_{k-1}^{i,\lambda=0}\right)\right)}{D}\\
&\leq -\eta\varepsilon^2\frac{\sum_{i=1}^N D_i\varphi_i\tau_i}{D}.
\end{aligned}$$

□

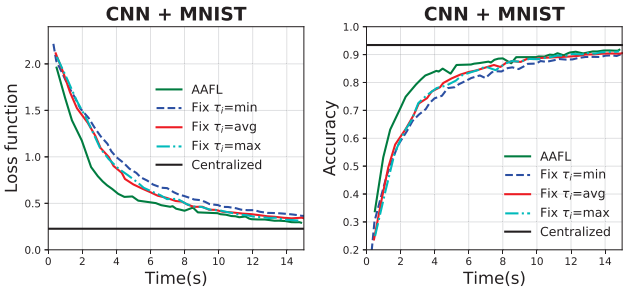



Fig. 13. Loss and accuracy with 80% fast work nodes for CNN + MNIST.

We are now ready to prove Lemma 2.

*Proof of Lemma 2:* Summing up (33) for all $k = 1,2,\ldots,K$, we have

$$\sum_{k=1}^K F(\boldsymbol{v}_k)-F(\boldsymbol{w}_{k-1}) \leq -K\eta\varepsilon^2\frac{\sum_{i=1}^N D_i\varphi_i\tau_i}{D}.$$

Rewriting the left-hand side.

$$\begin{aligned}
&F(\boldsymbol{v}_K)-F(\boldsymbol{w}_0)-\sum_{k=2}^K F(\boldsymbol{w}_{k-1})-F(\boldsymbol{v}_{k-1})\\
&\leq -K\eta\varepsilon^2\frac{\sum_{i=1}^N D_i\varphi_i\tau_i}{D}.
\end{aligned}$$

According to Theorem 1, we get

$$\begin{aligned}
&F(\boldsymbol{v}_K)-F(\boldsymbol{w}_0)\\
&\leq (K-1)\frac{\sum_{i=1}^N D_i\rho_ig_i(\tau_i)}{D}-K\eta\varepsilon^2\frac{\sum_{i=1}^N D_i\varphi_i\tau_i}{D},
\end{aligned}$$

and

$$F(\boldsymbol{w}_K)-F(\boldsymbol{v}_K) \leq \frac{\sum_{i=1}^N D_i\rho_ig_i(\tau_i)}{D}.$$

Combining the above, we get

$$\begin{aligned}
&F(\boldsymbol{w}_K)-F(\boldsymbol{w}_0)\\
&\leq K\frac{\sum_{i=1}^N D_i\rho_ig_i(\tau_i)}{D}-K\eta\varepsilon^2\frac{\sum_{i=1}^N D_i\varphi_i\tau_i}{D}.
\end{aligned}$$

Equivalently,

$$\begin{aligned}
&(F(\boldsymbol{w}_K)-F(\boldsymbol{w}^*))-(F(\boldsymbol{w}_0)-F(\boldsymbol{w}^*))\\
&\leq K\frac{\sum_{i=1}^N D_i\rho_ig_i(\tau_i)}{D}-K\eta\varepsilon^2\frac{\sum_{i=1}^N D_i\varphi_i\tau_i}{D}.
\end{aligned}$$

Recall that we defined $I = F(\boldsymbol{w}_0)-F(\boldsymbol{w}^*)$, $B_i = \frac{\sum_{i=1}^N D_i\rho_ig_i(\tau_i)}{D}$ and $C_i = \frac{\sum_{i=1}^N D_i\varphi_i\tau_i}{D}$, thus

$$F(\boldsymbol{w}_K)-F(\boldsymbol{w}^*) \leq I+KB_i-K\eta\varepsilon^2C_i.$$

□

## APPENDIX D
## ADDITIONAL RESULTS ON LOSS AND ACCURACY VALUES

See Fig. 13, Fig. 14 and Fig. 15.

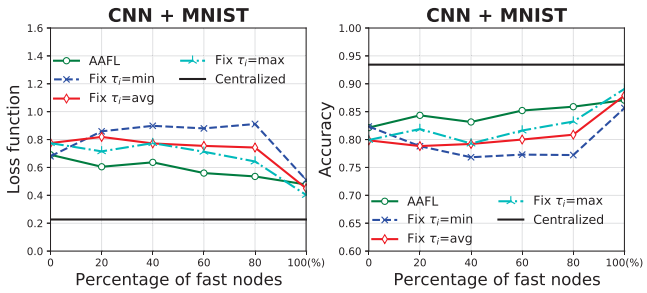



Fig. 14. Final loss and accuracy from 0% to 100% fast nodes for CNN + MNIST.

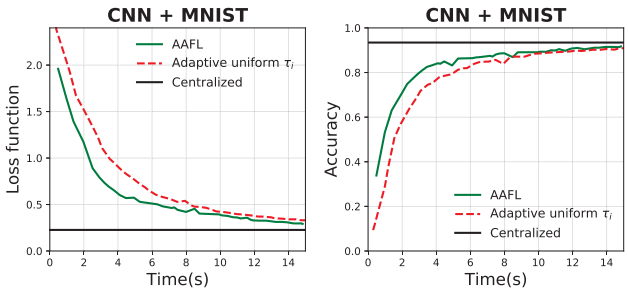



Fig. 15. Loss function and classification accuracy between the AAFL algorithm and the algorithm in [15] for CNN + MNIST.

## REFERENCES


[1] N. J. Beeching, T. E. Fletcher, and M. B. J. Beadsworth, "Covid-19: Testing times," *BMJ*, vol. 369, p. m1403, Apr. 2020.

[2] J. C. Gomes *et al.*, "IKONOS: An intelligent tool to support diagnosis of COVID-19 by texture analysis of X-ray images," *Res. Biomed. Eng.*, vol. 38, pp. 1–14, Sep. 2020.

[3] M. van der Schaar *et al.*, "How artificial intelligence and machine learning can help healthcare systems respond to COVID-19," *Mach. Learn.*, vol. 110, no. 1, pp. 1–14, Jan. 2021.

[4] S. Naz, K. T. Phan, and Y. P. Chen, "A comprehensive review of federated learning for COVID-19 detection," *Int. J. Intell. Syst.*, vol. 37, no. 3, pp. 2371–2392, Mar. 2022.

[5] W. N. Price and I. G. Cohen, "Privacy in the age of medical big data," *Nature Med.*, vol. 25, no. 1, pp. 37–43, Jan. 2019.

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

[7] P. Kairouz *et al.*, "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, nos. 1–2, pp. 1–210, Jun. 2021.

[8] I. Dayan *et al.*, "Federated learning for predicting clinical outcomes in patients with COVID-19," *Nature Med.*, vol. 27, no. 10, pp. 1735–1743, 2021.

[9] A. Castiglione, M. Umer, S. Sadiq, M. S. Obaidat, and P. Vijayakumar, "The role of Internet of Things to control the outbreak of COVID-19 pandemic," *IEEE Internet Things J.*, vol. 8, no. 21, pp. 16072–16082, Nov. 2021.

[10] J. Wang and G. Joshi, "Cooperative SGD: A unified framework for the design and analysis of local-update SGD algorithms," *J. Mach. Learn. Res.*, vol. 22, no. 213, pp. 1–50, 2021.

[11] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, vol. 2, 2020, pp. 429–450.

[12] M. S. H. Abad, E. Ozfatura, D. Gunduz, and O. Ercetin, "Hierarchical federated learning ACROSS heterogeneous cellular networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 8866–8870.

[13] H. Jiang, M. Liu, B. Yang, Q. Liu, J. Li, and X. Guo, "Customized federated learning for accelerated edge computing with heterogeneous task targets," *Comput. Netw.*, vol. 183, Dec. 2020, Art. no. 107569.

[14] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 7611–7623.

[15] S. Wang *et al.*, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 1205–1221, Jun. 2019.

[16] B. Yan *et al.*, "Experiments of federated learning for COVID-19 chest X-ray images," in *Advances in Artificial Intelligence and Security*. Cham, Switzerland: Springer, 2021, pp. 41–53.

[17] R. Kumar *et al.*, "Blockchain-federated-learning and deep learning models for COVID-19 detection using CT imaging," *IEEE Sensors J.*, vol. 21, no. 14, pp. 16301–16314, Jul. 2021.

[18] W. Zhang *et al.*, "Dynamic-fusion-based federated learning for COVID-19 detection," *IEEE Internet Things J.*, vol. 8, no. 21, pp. 15884–15891, Nov. 2021.

[19] C. Wang, X. Wei, and P. Zhou, "Optimize scheduling of federated learning on battery-powered mobile devices," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May 2020, pp. 212–221.

[20] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," 2019, *arXiv:1903.03934*.

[21] Q. Ma, Y. Xu, H. Xu, Z. Jiang, L. Huang, and H. Huang, "FedSA: A semi-asynchronous federated learning mechanism in heterogeneous edge computing," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3654–3672, Dec. 2021.

[22] A. Feraudo *et al.*, "CoLearn: Enabling federated learning in MUD-compliant IoT edge networks," in *Proc. 3rd ACM Int. Workshop Edge Syst., Analytics Netw.*, Apr. 2020, pp. 25–30.

[23] D. Conway-Jones, T. Tuor, S. Wang, and K. Leung, "Demonstration of federated learning in a resource-constrained networked environment," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, Dec. 2019, pp. 484–486.

[24] Y. Chen, X. Sun, and Y. Jin, "Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 10, pp. 4229–4238, Oct. 2020.

[25] S. Zheng *et al.*, "Asynchronous stochastic gradient descent with delay compensation," in *Proc. 34th Int. Conf. Mach. Learn. (PMLR)*, vol. 70, Aug. 2017, pp. 4120–4129.

[26] A. Khaled, K. Mishchenko, and P. Richtárik, "First analysis of local GD on heterogeneous data," 2019, *arXiv:1909.04715*.

[27] F. Haddadpour, M. M. Kamani, M. Mahdavi, and V. Cadambe, "Local SGD with periodic averaging: Tighter analysis and adaptive synchronization," in *Advances in Neural Information Processing Systems*, vol. 32. Red Hook, NY, USA: Curran Associates, 2019.

[28] H. T. Nguyen, V. Sehwag, S. Hosseinalipour, C. G. Brinton, M. Chiang, and H. V. Poor, "Fast-convergent federated learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 201–218, Jan. 2021.

[29] A. Khaled, K. Mishchenko, and P. Richtarik, "Tighter theory for local SGD on identical and heterogeneous data," in *Proc. 23rd Int. Conf. Artif. Intell. Statist.*, vol. 108, Aug. 2020, pp. 4519–4529.

[30] S. U. Stich and S. P. Karimireddy, "The error-feedback framework: SGD with delayed gradients," *J. Mach. Learn. Res.*, vol. 21, no. 237, pp. 1–36, 2020.

[31] B. E. Woodworth *et al.*, "Is local SGD better than minibatch SGD?" in *Proc. 37th Int. Conf. Mach. Learn.*, vol. 119, 2020, pp. 10334–10343.

[32] H. Yu, S. Yang, and S. Zhu, "Demystifying why model averaging works for deep learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 5693–5700.

[33] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," in *Proc. 8th Int. Conf. Learn. Represent. (ICLR)*, Addis Ababa, Ethiopia, Apr. 2020, pp. 1–26, [Online]. Available: https://OpenReview.net

[34] X. Liang, S. Shen, J. Liu, Z. Pan, E. Chen, and Y. Cheng, "Variance reduced local SGD with lower communication complexity," 2019, *arXiv:1912.12844*.

[35] F. Haddadpour and M. Mahdavi, "On the convergence of local descent methods in federated learning," 2019, *arXiv:1910.14425*.

[36] S. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proc. 37th Int. Conf. Mach. Learn. (ICML)*, 2020, pp. 5088–5099.

[37] N. Ivkin *et al.*, "Communication-efficient distributed SGD with sketching," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–11.

[38] H. Yu, R. Jin, and S. Yang, "On the linear speedup analysis of communication efficient momentum SGD for distributed non-convex optimization," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7184–7193.

[39] F. Haddadpour, M. M. Kamani, M. Mahdavi, and V. Cadambe, "Trading redundancy for communication: Speeding up distributed SGD for non-convex optimization," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2545–2554.

[40] M. K. Nori, S. Yun, and I.-M. Kim, "Fast federated learning by balancing communication trade-offs," *IEEE Trans. Commun.*, vol. 69, no. 8, pp. 5168–5182, Aug. 2021.
[41] 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.
[42] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Dept. Comput. Sci., Univ. Toronto, Tech. Rep., Jan. 2009, vol. 1.
[43] D. Sharifrazi *et al.*, "Fusion of convolution neural network, support vector machine and Sobel filter for accurate detection of COVID-19 patients using X-ray images," *Biomed. Signal Process. Control*, vol. 68, Jul. 2021, Art. no. 102622.
[44] S. Bubeck, "Convex optimization: Algorithms and complexity," *Found. Trends Mach. Learn.*, vol. 8, nos. 3–4, pp. 231–357, 2015.

**Jieren Cheng** (Member, IEEE) received the Ph.D. degree in computer science and technology from the National University of Defense Technology (NUDT) in 2010.

He is currently a Professor and the Associate Dean of the School of Computer Science & Technology, Hainan University, China. He is awarded as "Famous South China Sea Scholar." He works as the Director of the Hainan Provincial Blockchain Technology Engineering Research Center. His research interests include blockchain, big data, cloud computing, cyberecurity, artificial intelligence, and intelligent transportation. He hosted three National Natural Science Foundation of China, National Defense Key Research Projects, China–U.S. Computer Science Research Center Open Project, Ministry of Education Industry-University-Research Collaborative Education Project, Ministry of Education "Tiancheng Huizhi" Innovation and Education Fund, Hainan Province Key, Research and Development Innovation Team Projects, Hainan Provincial Key Research and Development Projects, Hainan Provincial Natural Science Foundation Projects, Hainan Provincial Science and Technology Enterprise Technology Innovation Fund Projects, and Hunan Provincial Twelfth Five-Year Plan Projects, with a total project funding of more than ten million. He has participated in ten national key projects as the main person in charge, including the National Natural Science Foundation of China, the National Defense Preliminary Research Key Project, the National Support Plan, and the Innovation Planning Project of the Ministry of Public Security. He has won 24 provincial-level projects and 12 school-level projects, such as the Provincial Natural Science Foundation, the Provincial Science and Technology Plan Fund, and the Provincial Department of Education Key Project.

Dr. Cheng is a Senior Member of CCF and a member of ACM. He has been invited to serve as a Reviewer in several journals and international conferences, e.g., *Computer Research and Development*, *Computer Science*, and FAW, and a PC member for several international conferences. He won the ICAIS 2021 Outstanding Organization Chairperson, ICCCS 2018 Outstanding Contribution Award, and the first prize of ICCCS 2018 and ICCCS 2017 Excellent Papers.

**Ping Luo** (Student Member, IEEE) received the B.Eng. degree in cyberspace security (cryptology) from Hainan University, Haikou, China, where he is currently pursuing the M.A.Eng. degree in computer science and technology. His research interests include convex optimization, the Industrial Internet of Things, and artificial intelligence.

**N. Xiong** (Senior Member, IEEE) received the Ph.D. degree from the School of Information Science, Japan Advanced Institute of Science and Technology (JAIST), in March 2008.

He is currently a Distinguished Professor at the School of Computer Science and Technology, Hainan University, Haikou, Hainan, China. He is also with the Department of Computer Science, Georgia State University, Atlanta, GA, USA. His research interests include deep learning, reliable networks, software engineering, and big data analytics. He works at CCNU for many years, and obtained many research funding and many industrial projects. He has published over 600 journal articles with more than 200 IEEE journal articles. He also creates a company about design and analysis for complex reliable software systems and obtains over ten patents.

**Jie Wu** (Fellow, IEEE) is currently the Director of the Center for Networked Computing and a Laura H. Carnell Professor at Temple University. He also serves as the Director of International Affairs at the College of Science and Technology. He served as Chair of the Department of Computer and Information Sciences from Summer 2009 to Summer 2016 and an Associate Vice Provost for International Affairs from Fall 2015 to Summer 2017. Prior to joining Temple University, he was the Program Director of the National Science Foundation and a Distinguished Professor at Florida Atlantic University. His current research interests include mobile computing and wireless networks, routing protocols, network trust and security, distributed algorithms, applied machine learning, and cloud computing. He regularly publishes in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON SERVICE COMPUTING, *Journal of Parallel and Distributed Computing*, and *Journal of Computer Science and Technology*. He is/was the General Chair/the Co-Chair for IEEE IPDPS'08, IEEE DCOSS'09, IEEE ICDCS'13, ACM MobiHoc'14, ICPP'16, IEEE CNS'16, WiOpt'21, and ICDCN'22, and the Program Chair/the Co-Chair for IEEE MASS'04, IEEE INFO-COM'11, CCF CNCC'13, and ICCCN'20. He was an IEEE Computer Society Distinguished Visitor, an ACM Distinguished Speaker, and the Chair of the IEEE Technical Committee on Distributed Processing (TCDP). He is a fellow of the AAAS. He was a recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.