

Time-sensitive Learning for Heterogeneous Federated Edge Intelligence

Yong Xiao, *Senior Member, IEEE*, Xiaohan Zhang, Yingyu Li, Guangming Shi, *Fellow, IEEE*, Marwan Krunz, *Fellow, IEEE*, Diep N. Nguyen, *Senior Member, IEEE*, and Dinh Thai Hoang, *Senior Member, IEEE*



Abstract—Real-time machine learning (ML) has recently attracted significant interest due to its potential to support instantaneous learning, adaptation, and decision making in a wide range of application domains, including self-driving vehicles, intelligent transportation, and industry automation. In this paper, we investigate real-time ML in a federated edge intelligence (FEI) system, an edge computing system that implements federated learning (FL) solutions based on data samples collected and uploaded from decentralized data networks, e.g., Internet-of-Things (IoT) and/or wireless sensor networks. FEI systems often exhibit heterogeneous communication and computational resource distribution, as well as non-i.i.d. data samples arrived at different edge servers, resulting in long model training time and inefficient resource utilization. Motivated by this fact, we propose a time-sensitive federated learning (TS-FL) framework to minimize the overall run-time for collaboratively training a shared ML model with desirable accuracy. Training acceleration solutions for both TS-FL with synchronous coordination (TS-FL-SC) and asynchronous coordination (TS-FL-ASC) are investigated. To address the straggler effect in TS-FL-SC, we develop an analytical solution to characterize the impact of selecting different subsets of edge servers on the overall model training time. A server dropping-based solution is proposed to allow some slow-performance edge servers to be removed from participating in the model training if their impact on the resulting model accuracy is limited. A joint optimization algorithm is proposed to minimize the overall time consumption of model training by selecting participating edge servers, the local epoch number (the number of model training iterations per coordination), and the data batch

size (the number of data samples for each model training iteration). Motivated by the fact that data samples at the slowest edge server may exhibit special characteristics that cannot be removed from model training, we develop an analytical expression to characterize the impact of both staleness effect of asynchronous coordination and straggler effect of FL on the time consumption of TS-FL-ASC. We propose a load forwarding-based solution that allows a slow edge server to offload part of its training samples to trusted edge servers with higher processing capability. We develop a hardware prototype to evaluate the model training time of a heterogeneous FEI system. Experimental results show that our proposed TS-FL-SC and TS-FL-ASC can provide up to 63% and 28% of reduction, in the overall model training time, respectively, compared with traditional FL solutions.

Index Terms—Time-sensitive machine learning, edge intelligence, federated learning, asynchronous coordination.

1 INTRODUCTION

The proliferation of smart applications that require real-time learning, adaptation, and decision making, such as self-driving vehicles, intelligent transportation systems [1], and Tactile Internet [2], has significantly increased the demand for machine learning (ML)-enabled solutions that support fast proactive learning and model construction based on large datasets. The so-called *real-time ML* has recently attracted significant interest due to its potential to quickly solve unfamiliar problems and self-adapt to unknown situations [3]. Existing ML solutions are often computational demanding and rely on large datasets to be collected and pre-loaded into a centralized location, e.g., cloud data center, and are therefore infeasible for real-time operation. Recent concerns over data privacy further exacerbate the challenge, as some “local” users do not wish to disclose their datasets to the high-performance data center due to privacy concerns or regulation restrictions.

A highly promising solution is federated learning (FL) [4], an emerging ML framework that enables multiple servers to jointly train a shared model without exposing the private data owned by individual users. The key idea is to allow edge servers to train local ML models using their local data samples and periodically coordinate with each other through their locally trained model parameters [5], [6]. Federated edge intelligence (FEI) [7], [8], is an emerging paradigm that focuses on the implementation of FL-based solutions in edge computing systems. FEI has recently been

*This work has been accepted at IEEE Transactions on Mobile Computing. Copyright may be transferred without notice, after which this version may no longer be accessible.

Y. Xiao is with the School of Electronic Information and Communications at the Huazhong University of Science and Technology, Wuhan 430074, China, also with the Peng Cheng Laboratory, Shenzhen, Guangdong 518055, China, and also with the Pazhou Laboratory (Huangpu), Guangzhou, Guangdong 510555, China (e-mail: yongxiao@hust.edu.cn).

X. Zhang is with the School of Electronic Information and Communications at the Huazhong University of Science and Technology, Wuhan, China 430074 (e-mail: xiaohan.zhang@hust.edu.cn).

Y. Li is with the School of Mechanical Engineering and Electronic Information, China University of Geosciences, Wuhan, China 430074 (e-mail: liyingyu29@cug.edu.cn).

G. Shi is with the Peng Cheng Laboratory, Shenzhen, Guangdong 518055, China, also with the School of Artificial Intelligence, the Xidian University, Xi’an, Shaanxi 710071, China, and also with the Pazhou Laboratory (Huangpu), Guangzhou, Guangdong 510555, China (e-mail: gmshi@xidian.edu.cn).

M. Krunz is with the Department of Electrical and Computer Engineering, the University of Arizona, Tucson, AZ 85721 (e-mail: krunz@arizona.edu).

D. Nguyen and D. Hoang are with the School of Electrical and Data Engineering, University of Technology Sydney Faculty of Engineering and Information Technology, 120558 Sydney, New South Wales, Australia (e-mail: {hoang.dinh, diep.nguyen}@uts.edu.au).

promoted by both industry and academia as a key candidate solution in the next generation mobile technologies, e.g., beyond 5G and 6G [9], [10].

Although FEI has the potential to protect data privacy, enable parallel computation, and avoid the delay of transporting raw data samples to the cloud data center, it brings new challenges when applied to real-time ML applications. First, the performance of FEI can be affected by both computation and communication-related performance metrics, including the processing capabilities of edge servers (clients), data collection and uploading delay of data collecting devices, frequency of coordination during model training, and communication bandwidth between edge servers and the coordinator. These challenges significantly increase the complexity to search for the optimal solution to minimize the overall run-time of training the ML model. For example, reducing the number of local training iterations between successive coordination rounds may accelerate the convergence speed. It however increases the frequency of model coordination which will result in high communication delay. Similarly, processing more local training samples during each iteration reduces the required number of model coordination rounds to reach the target model accuracy. However, it also slows down the data loading speed and increases the computational time in each iteration. How to jointly optimize computation and communication-related parameters to minimize the overall time consumed for model training is still an open problem. Furthermore, traditional FL, especially FL with synchronous coordination, is known to suffer from the *straggler effect* [11], i.e., the overall ML model training delay is dominated by the slowest edge server. There is still lacking a comprehensive solution that can alleviate the straggler effect and expedite the model training speed in FEI systems under both system heterogeneity, i.e., various hardware and software resources available at different edge servers, and data heterogeneity, i.e., non-i.i.d. distributions of data samples across different edge servers. Recent studies have investigated FL with asynchronous coordination in which each edge server does not have to wait for other edge servers, but can request an instantaneous model update from the coordinator. However, previous works suggest that the asynchronous coordination often results in degraded convergence performance, compared to the synchronous coordination solutions [12], due to the *staleness effect*, that is the increasing difference in the model updating frequencies between edge servers with different computational speeds and communication bandwidths will result in out-of-date models at some slow edge servers, especially when the total number of model coordination rounds becomes large.

In this paper, we propose the time-sensitive FL (TS-FL) framework, which aims at minimizing the overall runtime for training an ML model with the guaranteed accuracy, the percentage of the correct predictions among all the predictions made by the trained model. In contrast to existing works that focus on reducing the required number of stochastic gradient descent (SGD) coordination rounds for model training [13], the overall runtime needed to construct an ML model is a much more important performance metric in many real-world applications, especially the time-sensitive applications. We observe that, in many practical systems, the required number of SGD coordination rounds

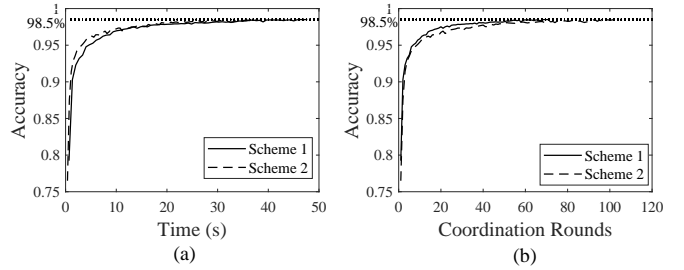


Fig. 1. (a) Model accuracy vs. training time, and (b) model accuracy vs. number of coordination rounds under two FL model training schemes: *Scheme 1* (200 local epoch number, 20 mini-batch size) and *Scheme 2* (200 local epoch number, 10 mini-batch size). The target accuracy of the trained model is set to 98.5%.

and the overall runtime for model training may not follow the same trend, i.e., a smaller number of SGD rounds does not always mean shorter runtime for training the model with the same accuracy. To shed more light on this observation, we conduct experiments on a hardware prototype that consists of 20 Raspberry Pis (version 4B) serving as edge servers. These mini-computers are connected to a dedicated Raspberry Pi, which serves as the coordinator, via a Wi-Fi router. We compare the required time consumption (in Fig. 1(a)) and the number of SGD coordination rounds (in Fig. 1(b)) for training a shared handwritten digit recognition model based on the MNIST dataset with the same requirement in terms of the model prediction accuracy (e.g., 98.5%) under two different schemes of model training parameters, labelled as *Schemes 1* and *2*, respectively. We can observe that, although *Scheme 2* requires a smaller number of SGD coordination rounds than *Scheme 1* for training the model with the required accuracy, it consumes more time for model training than *Scheme 1*. In other words, the number of SGD rounds cannot fully characterize the overall runtime of the learning process in a heterogeneous FEI system.

Motivated by the above observation, in this paper, we investigate model training time-acceleration solutions for two types of TS-FLs: TS-FL with synchronous coordination (TS-FL-SC) and TS-FL with asynchronous coordination (TS-FL-ASC). In TS-FL-SC, the coordinator updates the global model when it receives the local models from all the edge servers selected for participating in model training. To alleviate the straggler effect of TS-FL-SC, we develop an analytical solution that characterizes the impact of selecting different subsets of edge servers on the overall model training time. We then develop a server dropping-based solution in which some slow-performing edge servers are removed from participating in model training so long as their impact on the model accuracy is limited. A novel algorithm is proposed to jointly optimize the selection of participating edge servers, number of local epochs (i.e., number of model training iterations between two successive coordination rounds), and data batch size (data samples used for each local model training iteration) so as to minimize the overall time of model training. Motivated by the fact that, in some scenarios, data samples at the slowest edge server may exhibit special characteristics that are essential for resulting model accuracy, we consider the TS-FL-ASC, where model coordination among edge servers can be asynchronous. In

particular, for TS-FL-ASC, we develop an analytical expression to characterize the *staleness effect* of asynchronous coordination systems. To reduce staleness and alleviate the straggler effect in this case, we propose a load forwarding-based solution that allows slow edge servers to offload part of their training datasets to other trusted edge servers with high processing capabilities. Extensive experiments are conducted based on a hardware prototype consisting of different versions of Raspberry Pi mini-computers (versions 4B and 3A), which we use to simulate model training time under various scenarios.

To the best of our knowledge, this is the first work that investigates the time-sensitive FEI systems with precision guarantee by jointly optimizing server dropping, multivariate parameter control, as well as asynchronous load offloading under both system heterogeneity and data heterogeneity.

The main contributions of this paper are as follows:

- (1) **Time-sensitive FL for Heterogeneous FEI:** We provide a unified analytical model that quantifies the impact of different training parameters on the time needed for different steps throughout the entire model training process for both TS-FL-SC and TS-FL-ASC.
- (2) **New Theoretical Results and an Optimization Algorithms:** We derive new theoretical results as well as optimization solutions for both TS-FL-SC and TS-FL-ASC. In particular, for TS-FL-SC, we derive an analytical upper bound to characterize the overall model training time under different selections of participating edge servers as well as combinations of their model training parameters. We also develop an optimization algorithm that can select the optimal subset of participating edge servers and their parameters so as to minimize the model training time for an FEI system with both system and data heterogeneity. For TS-FL-ASC, we derive a new theoretical convergence result with non-independent and identically distributed (non-i.i.d.) data samples at edge servers. We propose a load forwarding-based solution in which slow edge servers can offload part of their training datasets to trusted edge servers with higher processing capability. We prove that the training time minimization problem based on our derived bound is biconvex and then develop a novel algorithm to select the optimal subset of participating edge servers as well as their model training parameters that can minimize the overall model training time.
- (3) **Extensive Experiments Based on a Hardware Platform:** We develop a hardware platform consisting of 23 low-cost mini-edge servers (Raspberry Pi mini-computers) with various computing capabilities. Extensive experiments are conducted to evaluate the model training time under various setups. Experimental results have shown that our proposed TS-FL-SC and TS-FL-ASC can, respectively, provide up to 59.31% and 57.24% of reduction in the overall model training time compared to the existing FL solutions.

The remainder of this paper is organized as follows. Related works are reviewed in Section 2. We introduce the

system model and problem formulation in Section 3. TS-FL-SC and TS-FL-ASC are introduced in Sections 4 and 5, respectively. Numerical results are presented in Section 6, and the paper is concluded in Section 7.

2 RELATED WORK

FEI in Wireless Networks: By integrating federated learning into edge intelligence, FEI enables collaborative learning and model construction based on decentralized datasets across a large networking system [10]. Most existing works have focused on optimizing the allocation of communication and computational resources to improve the efficiency of FEI systems. The authors in [14] developed algorithms that can dynamically control the frequency of global aggregation at edge servers to improve the resource utilization and also reduce the energy consumption of a 5G-based edge networking system. Recently, FEI has been also extended into semantic communications [15], especially the implicit semantic-aware networks which allows the networking systems to automatically infer hidden information such as hidden rules and mechanisms from the communicated messages [16]–[18]. Although these works have shown significant progress in improving the communication and computational efficiency of FEI networks, they mainly focused on the impact of a single parameter, e.g., model updating frequency, local epoch number, or the selection of edge servers for participating model training, etc. Analytical frameworks that can characterize the impact of various combinations of key parameters on the performance of FEI networking systems are still lacking.

FEI for Heterogeneous Networks: System heterogeneity refers to differences in computation, communication, and storage capacities among edge devices. It is one of the major causes of the straggler effect in FEI [19], [20]. Server (device/client) sampling [4], [21] is one approach to address system heterogeneity in FEI. For example, the authors in [22] adopted an active server sampling method that allows the coordinator to aggregate as many locally trained model parameters as possible within a predefined time window so as to mitigate the influence of heterogeneous devices. However, these works are under the assumption of static system characteristics, which lacking the ability to handle device-specific latency in many practical applications. Data heterogeneity, which reflects the diversity of local data samples across different edge servers in terms of data type and statistical distribution may further aggravate the complexity of problem modeling, theoretical analysis, and evaluation of solutions of an FEI system [23], [24]. Recent works leveraging on meta-learning and multitask learning [25]–[27] provide feasible solutions for modeling heterogeneous data. For instance, the authors in [25] proposed a Bayesian network to model the star topology of an FEI system to mitigate the adverse effect of data heterogeneity. However, most of these solutions are expensive to implement and often involve solving complex non-convex problems, making them impractical for large scale FEI networks.

FL with Asynchronous Coordination: Due to the heterogeneity of devices and local datasets, FL with synchronous coordination suffers from the “straggler effect” that may deteriorate the convergence performance. There are already

some recent works trying to alleviate the straggler effect in FEI systems by considering the asynchronous coordination solutions for FL. For example, the authors in [21] proposed FedProx, a variant of FedAvg, in which a proximal term is proposed to balance the trade-off between data and system heterogeneity in FL [19] as well as ensure convergence for both convex and nonconvex loss functions. The authors in [28] proposed an asynchronous federated optimization algorithm that could mitigate the staleness of edge servers and guarantee near-linear convergence to a global optimal solution. In [11], the authors proposed an asynchronous local SGD algorithm to overcome the communication bottlenecks in a large FEI system with i.i.d. distributed data samples across edge servers.

Time-sensitive FL: With the popularity of time-sensitive smart applications and services, acceleration solutions that can reduce the overall model training time of FL has attracted significant interest recently. Most of existing works focus on optimizing the model parameters to accelerate the training speed. For example, in [14], [29], the authors optimized the number of local epochs to reduce the runtime of model training. In [30], the authors proposed a novel multi-armed bandit-based solution to alleviate the straggler effect of FL by optimizing the scheduling of edge server. In [31], the authors proposed a multi-armed bandit-based online edge server scheduling solution that can accelerate the training speed without knowing the channel state information and statistic characteristics of edge servers. In [32], the authors jointly optimized the number of local epochs, scheduling of edge servers, and the maximum number of coordination rounds to minimize the energy consumption and learning time. Different from these existing works, in this paper, we introduce a unified framework for FL with both synchronous coordination and asynchronous coordination. We propose a framework that can jointly optimize the edge server selection, number of local epochs, batchsizes, as well as training load re-distribution to address the straggler effect for FL with synchronous coordination as well as the staleness effect for FL with asynchronous coordination. We have verified the effectiveness of our proposed framework both theoretically and practically. We summarize the main difference between the existing works and ours in Table 1.

3 SYSTEM MODEL

We first introduce the basic components and the processing steps of an FEI system. We then discuss the possible time consumption in each step of the training process and finally formulate the optimization problem for time-sensitive learning of an FEI system.

3.1 FEI Model

We consider an FEI system consisting of three major components:

- (1) *Data collecting network:* It consists of a large number of devices that can collect local data samples to be uploaded to the corresponding edge servers.
- (2) *Edge servers:* A set $\mathcal{K} = \{1, 2, \dots, K\}$ of K edge servers are deployed to store and process the data samples uploaded from the data collecting devices.

In this paper, we assume that each edge server is associated with an exclusive set of data collecting devices. Let \mathcal{D}_k be the distribution of the data samples arrived at the k th edge server for $k \in \mathcal{K}$. Generally speaking, data samples arrived at different edge servers follow different distributions, i.e., we have $\mathcal{D}_k \neq \mathcal{D}_j$ for $k \neq j$ and $k, j \in \mathcal{K}$.

- (3) *Coordinator:* It coordinates the model training between edge servers via their intermediate results. It can be deployed at the cloud data center or one of the edge servers. All or a subset of edge servers can be selected to upload the intermediate model training results, e.g., model parameters, to the coordinator once in a while. These edge servers will wait for the coordinator to feedback an updated result, e.g., aggregation of all the received result in the FedAvg algorithm [23], before continuing with their model training process.

We consider the standard FL-based model training process described as follows. At the beginning of process, a subset of edge servers can download an initial model from the coordinator. Each edge server will then train a local model based on its local data samples and coordinate its local model training process with others once in a while. We focus on the real-time data uploading and training process. During each round of model training, an edge server first receives a batch of data samples from its associated data collecting devices and updates its local model based on this batch before coordinating with other edge servers.

The main objective is to collaboratively train a shared model that minimizes a global objective function $F(\mathbf{w})$, corresponding to the weighted sum of the local objective functions of edge servers. Formally, we can write

$$\min_{\mathbf{w} \in \mathbb{R}^d} \{F(\mathbf{w}) = \sum_{k \in \mathcal{K}} p_k F_k(\mathbf{w})\} \quad (1)$$

where $F_k(\mathbf{w})$ is the local objective function of edge server k , \mathbf{w} is the set of parameters of the model with dimensional size d , and p_k is the weight of edge server k , where $0 \leq p_k \leq 1$ and $\sum_{k=1}^K p_k = 1$. In some settings, p_k can represent the relative portion of the local dataset of each edge server k [4], [33].

$F_k(\mathbf{w})$ corresponds to the average loss of prediction over all data samples:

$$F_k(\mathbf{w}) = \mathbb{E}_{x_{k,j} \sim \mathcal{D}_k} [l_k(\mathbf{w}; x_{k,j})] \quad (2)$$

where $l_k(\mathbf{w}; x_{k,j})$ is the loss function of edge server k with parameters \mathbf{w} and local data sample $x_{k,j}$.

In a standard FL with synchronous coordination, problem (1) is solved by allowing each edge server k to employ local SGDs to iteratively update its local parameters \mathbf{w}_k and periodically upload its model to the coordinator for model aggregation $\mathbf{w} = \sum_{k \in \mathcal{K}} p_k \mathbf{w}_k$.

In this paper, we consider a more general setting that can support both synchronous and asynchronous coordination. We use subscript t to denote the t th iteration of the local model training process at an edge server. Let \mathcal{I}^k be the set of iteration steps that edge server k uploads its local model to the coordinator. Let $\mathbf{v}_{k,t}$ be the updated model received

TABLE 1
SUMMARY OF RELATED WORK IN TIME-SENSITIVE FL

Reference	Model Coord.		Parameter Optimization		Server Drop.	Client Sched.
	Syn. Coord.	Asyn. Coord.	# of Local Epoch	Batchsize		
[29]	✓					✓
[30]		✓				✓
[14], [31]	✓		✓			
[32], [33]	✓					✓
[34]	✓		✓			✓
This Work	✓	✓	✓	✓	✓	✓

by edge server k from the coordinator during iteration t , $t \in \mathcal{I}^k$. We can write the model updating process as follows

$$\mathbf{w}_{k,t+1} = \begin{cases} \mathbf{w}_{k,t} - \eta_t \mathbf{g}_{k,t}, & \text{if } t+1 \notin \mathcal{I}^k \\ \mathbf{v}_{k,t+1}, & \text{if } t+1 \in \mathcal{I}^k \end{cases} \quad (3)$$

where η_t is the learning rate. In this paper, we consider a general scenario in which the learning rate η_t can be a time-varying variable. It has already been proved that adopting time-varying learning rate, e.g., diminishing learning rate, is critical to ensure the convergence for most SGD-based algorithmic solutions [34] [35] [36]. $\mathbf{g}_{k,t}$ is the stochastic gradient, calculated based on the locally received data samples:

$$\mathbf{g}_{k,t} = \frac{1}{n_k} \sum_{j=1}^{n_k} \nabla l_k(\mathbf{w}_{k,t}, x_{k,j,t}), \quad (4)$$

and $\mathbf{v}_{k,t+1}$ is the model aggregated by the coordinator.

In (4), we assume n_k training data samples $\{x_{k,1,t}, x_{k,2,t}, \dots, x_{k,n_k,t}\}$ are computed by edge server k during the t th training iteration. We follow the commonly adopted setting [37] and assume data samples at any edge server have unbiased independent stochastic gradients, i.e., $\mathbb{E}_{x_{k,j,t} \sim \mathcal{D}_k}[\mathbf{g}_{k,t} | \mathbf{x}_{t-1}] = \nabla F_k(\mathbf{w}_{k,t})$, where $\mathbf{x}_{t-1} = \langle x_{k,j,\tau} \rangle_{k \in \mathcal{K}, j \in \{1, \dots, n_k\}, \tau \in \{0, \dots, t-1\}}$.

FL with synchronous and asynchronous coordination is illustrated in Fig. 2. With synchronous coordination, the coordinator will periodically aggregate all local models uploaded by the edge servers that are participating in model training. Let \mathcal{M} be the subset of edge servers participating in model training, $\mathcal{M} \subseteq \mathcal{K}$. We can write $\mathcal{I}^k = \mathcal{I}^j = \mathcal{I}$ for $k \neq j$, $k, j \in \mathcal{M}$ and $\mathbf{v}_{k,t}$ in (3) can be written as

$$\mathbf{v}_{k,t} = \sum_{k \in \mathcal{M}} p_k (\mathbf{w}_{k,t} - \eta_t \mathbf{g}_{k,t}), \text{ for } t \in \mathcal{I}. \quad (5)$$

In the asynchronous coordination case, the coordinator will update the global model whenever it receives a local model from any edge server. In this case, the model training and updating speeds at different edge servers are different, i.e., the t th iteration of model training at edge server k may correspond to the $(t+j)$ th iteration at edge server l . Without loss of generality, we focus on the model updating process of an individual edge server, i.e., edge server k . Let $\mathcal{W}_{k,t}^h \subseteq \{0, 1, 2, \dots, T-1\}$ be the set of local SGD updates of $\{\mathbf{w}_{h,t}\}_{t \geq 0}$ that have been aggregated and integrated into the global model during the past updating steps and before the t th interaction of edge server k . Note that $\mathcal{W}_{k,t}^h$ can be different from $\mathcal{W}_{k',t}^h$ for $k \neq k'$, as different edge servers usually reach iteration t at different times. Since the coordinator keeps aggregating more local updates into the

global model, we have $\mathcal{W}_{k,t}^h \subseteq \mathcal{W}_{k,t'}^h$ for $t' \geq t$. We use $|\mathcal{W}_{k,t}^h|$ to represent the cardinality of $\mathcal{W}_{k,t}^h$, i.e., the total number of local SGD updates uploaded from edge server h that have already been aggregated into the global model at iteration t . We can then write the updated global model received from the model coordination for edge server k in its t th iteration for $t \in \mathcal{I}^k$ as

$$\mathbf{v}_{k,t} = \mathbf{w}_0 - \sum_{h \in \mathcal{M}} p_h \sum_{j \in \mathcal{W}_{k,t}^h} \eta_j \mathbf{g}_{k,j}, \text{ for } t \in \mathcal{I}^k. \quad (6)$$

where \mathbf{w}_0 is the initial model distributed by the coordinator.

3.2 Problem Formulation

We investigate the time-sensitive learning of an FEI system in which the main design objective is to learn an ML model with a target accuracy as quickly as possible. This is different from most existing works that mainly focus on improving the convergence rate [13]. It has already been observed that the overall time spent on training an ML model not only depends on the number of global and local SGD iterations performed by edge servers, but also on data collection, model computation, and updating performance.

As illustrated in Fig. 3, each round of collaborative training in an FEI system consists of four major steps: (1) model distribution, (2) data uploading, (3) model training, and (4) model updating and aggregation. Next, we discuss the time consumption of each step.

Model Distribution: The coordinator updates the global model using (3) and distributes the updated model to a subset \mathcal{M} of edge servers. Because in this step the coordinator broadcasts the same model to all selected edge servers, the time consumption does not scale with the number of edge servers and can therefore be considered as a constant, denoted by ζ .

Data Uploading: Each edge server k will update its local model based on data samples uploaded from its associated local data collecting network. We follow the standard FL approach and assume that in each round of model coordination, each edge server first loads the required number of data samples from its associated data collecting network and then randomly samples a mini-batch of the loaded training samples at every local epoch (local model training iteration). Let e_k be the number of local epochs for each round of model updating. Edge server k will load $(n_k e_k)$ training samples in total during each round of model updating. Let a_k (in seconds per sample) be the sample arrival

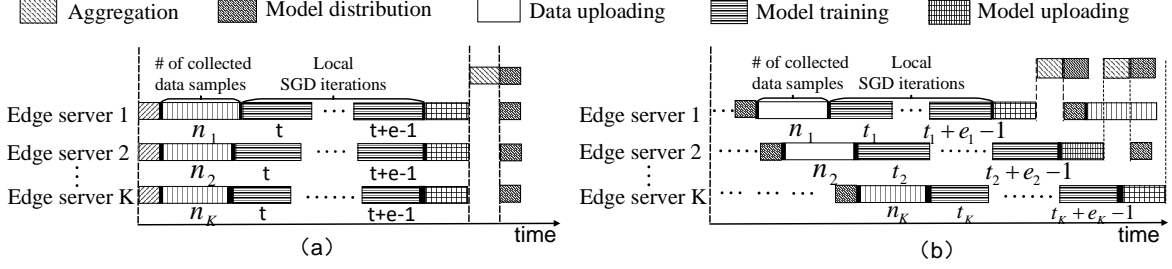


Fig. 2. Illustration of collaborative model training in an FEI system under: (a) synchronous coordination, and (b) asynchronous coordination.

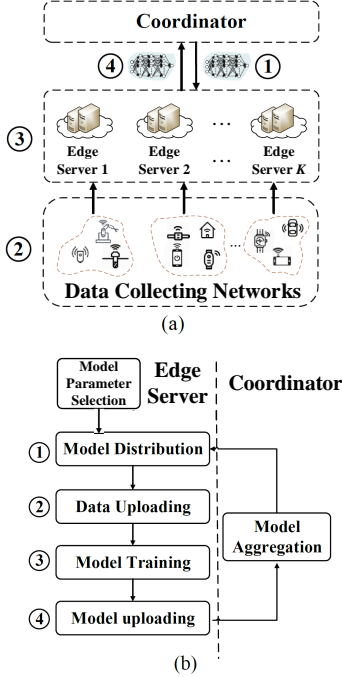


Fig. 3. An FEI architecture: (a) key system components, and (b) key training procedures.

rate at edge server k . The total time duration of the data uploading step can be written as

$$q_k(n_k, e_k) = a_k n_k e_k. \quad (7)$$

Local Model Training: Each edge server k performs e_k local SGD steps between two consecutive model updating steps. In addition to the values of e_k and n_k , the time needed to train the local model is closely related to the computational capacity of the edge server. As observed in [38], the time duration of model training is closely related to the type ς (image, voice, text, etc.) of data samples, the time for processing each data sample by edge server k , denoted as $b_{k,\varsigma}$, and the time consumption for computing the gradient and updating the local model weight, denoted as $\beta_{k,\varsigma}$. We can therefore write the time consumption of local model training at edge server k as $\nu_k(e_k, n_k, b_{k,\varsigma}, \beta_{k,\varsigma}, \varsigma)$.

Model Uploading and Aggregation: After e_k iterations of local SGD updates, each edge server k will upload its local model to the coordinator for global model updating. In the synchronous coordination scenario, the coordinator will have to wait for all the selected edge servers to upload their updated models before performing aggregation. In this case, the model uploading time mainly depends on the slowest

edge server selected to participate in the given round of model coordination. Motivated by the fact that edge servers and the coordination server are usually connected through high-speed link (e.g., fiber) with a fixed connecting speed, we follow a commonly adopted setup and assume the time consumption for model uploading is a constant, denoted as u .

Based on the above analysis, the minimization of the model training time of an FEI system can be formulated as follows:

$$\begin{aligned} (\mathbf{P}) \quad & \min_{e, \mathbf{n}, \Gamma, \mathcal{M}} \mathbb{E}[c(e, \mathbf{n}, \Gamma, \mathcal{M})] \\ & \text{s.t.} \quad \mathbb{E}[F(\mathbf{w}_{\mathcal{M}, \Gamma})] - F^* \leq \epsilon \quad (8) \\ & e_k, \Gamma, n_k \in \mathbb{Z}^+ \quad (9) \\ & \mathcal{M} \subseteq \mathcal{K} \quad (10) \end{aligned}$$

where $c(e, \mathbf{n}, \Gamma, \mathcal{M}) = \sum_{\tau=1}^{\Gamma} \max_{k \in \mathcal{M}} [\zeta + u + q_k(n_k, e_k) + \nu_k(e_k, n_k, b_{k,\varsigma}, \beta_{k,\varsigma}, \varsigma)]$, ϵ is the precision requirement of the ML model, $\mathbf{w}_{\mathcal{M}, \Gamma}$ denotes the parameters of the global model trained based on edge servers \mathcal{M} after Γ rounds of global model coordination, and $F^* = \min_{\mathbf{w} \in \mathcal{R}^d} F(\mathbf{w})$.

We can observe from problem (\mathbf{P}) that one of the key issues that slows down the FL-based model training is the so-called *straggler effect*, in which faster edge servers will have to wait for *stragglers*, i.e., slow edge servers with limited computation and communication capabilities, to finish each round of the global model coordination.

Solving problem (\mathbf{P}) involves joint optimization of multiple entangled model training parameters, including e , \mathbf{n} , and \mathcal{M} . The heterogeneity of the system (e.g., in terms of the computational capacities of different edge servers) and datasets (e.g. non-iid distributions of data samples at different edge servers) further exacerbate the challenge. An analytical framework for characterizing the relationship and understanding the joint impact of these parameters is needed. Joint optimization of all these relevant parameters to improve the speed of model training process in the FEI system with both system and data heterogeneity is still an open problem.

4 TS-FL WITH SYNCHRONOUS COORDINATION

4.1 Training Time Modeling and Approximation

It is known that the model training speed of each individual edge server is dominated by the computational capacity of the edge server as well as the adopted model training parameters [39]. We follow a commonly used setting and

assume the time duration of each local model training iteration of a specific edge server is a fixed value under a given combination of model training parameters.

During each round of model coordination in TS-FL-SC, the coordinator needs to wait for all the selected edge servers to finish uploading their model before performing model aggregation and updating. In this case, each round of model coordination will be dominated by the slowest edge server. Without loss of generality, we order the set \mathcal{M} of M servers from the fastest to the slowest based on their computational capacity and relabel edge servers according to their speed rankings, i.e., we abuse the notation and again use subscript m to denote the m th fastest edge server and therefore the set of edge servers is relabeled from the fastest to the slowest as $1, 2, \dots, M$. The overall time consumption during each round of model updating among set \mathcal{M} of edge servers is therefore dominated by the slowest edge server M . For Γ iterations of local model training in TS-FL-SC, the training time of the set \mathcal{M} of edge servers then can be written as

$$\mathbb{E}[c(e, n, \Gamma, \mathcal{M})] = \Gamma \cdot c_M(e_M, n_M, b_{M,\varsigma}, \beta_{M,\varsigma}, \varsigma) \quad (11)$$

where $c_M(e_M, n_M, b_{M,\varsigma}, \beta_{M,\varsigma}, \varsigma)$ is the time duration of each iteration in the model training of the slowest edge server M under parameters e_M and n_M and a given type of data sample ς . $b_{k,\varsigma}$ and $\beta_{k,\varsigma}$ are closely related to the computational capacity of edge server and can therefore be considered as a constant under given combination of e_k and n_k .

From (11), one can envision two potential solutions to alleviate the straggler effect for TS-FL-SC:

- (1) *Server Dropping*, which involves directly removing the slowest edge server as well as its corresponding data samples from participating in the model training process, and
- (2) *Parameter Optimization*, which focuses on optimizing a combination of parameters, especially e_k and n_k , to reduce the time consumed for each iteration and also improve the convergence performance.

Both of the above solutions have their pros and cons. In particular, simply removing the slowest edge server from the model training can reduce the waiting time in each round of coordination. It will however make the resulting model impossible to converge to the global optimal solution with all the data samples from all the edge servers being included in the training process. The parameter optimization solution is easy for implementation. However, it is generally impossible for the coordinator or edge servers to know the optimal combination of all the parameters. In the rest of this section, we will develop solutions to the above issues. In particular, we will first consider the server dropping approach. Later on, we will investigate the parameter optimization approach.

4.2 Server Dropping

4.2.1 Impact of Server Dropping

Even though server dropping can significantly reduce the overall time consumed by collaborative model training and

updating process, it may bias the result and slow convergence speed due to the removal of some data samples from model training. More specifically, recent results [40] suggest that if the data samples from all edge servers follow the same distribution, the FL convergence speed increases almost linearly with the number of edge servers. However, if data samples at different servers follow different distributions (non-i.i.d.), the impact of removing some edge servers from model training on the convergence performance will be much more complex. In particular, it is possible that the data samples at the slowest edge server may exhibit unique features which, when removed from the model training process, may prevent the resulting model from converging to the global optimal solution. In this paper, we focus on such scenarios. In this case, we need to first quantify the impact of removing different subsets of edge servers from model training on the convergence performance and then evaluate the overall time consumption of the training process.

Suppose the optimal model parameter that minimize the global objective function with all K available edge servers being selected is given by $\mathbf{w}_K^* = \arg \min_{\mathbf{w}} \sum_{k \in \mathcal{K}} p_k F_k(\mathbf{w}_K)$. To simplify notations, we can also write $F_k^* = F_k(\mathbf{w}_K^*)$. We then have the following definition.

Definition 1. We define the *impact of server dropping* as follows.

The impact of removing a subset $\mathcal{Q} \subset \mathcal{K}$ of edge servers and their corresponding data samples from the model training is defined as difference between the global optimal objective function with all K edge servers and that with subset $\mathcal{M} = \mathcal{K} \setminus \mathcal{Q}$ of edge servers, given by

$$D_{\mathcal{M}}^*(F) = \sum_{k \in \mathcal{M}} p_k (F_k(\mathbf{w}_{\mathcal{M}}^*) - F_k^*), \quad (12)$$

where $\mathbf{w}_{\mathcal{M}}^*$ is the optimal models that minimize the weighted sum of the loss functions of edge server set \mathcal{M} , defined as $\mathbf{w}_{\mathcal{M}}^* = \arg \min_{\mathbf{w}} \sum_{k \in \mathcal{M}} p_k F_k$.

One of the key differences of the above definition and the previously proposed concepts of server' impact or contribution [] is that, in our definition, the impact of server dropping depends on the minimized loss functions with the optimal model parameters. In the rest of this section, we first quantify the gap of loss functions caused by server dropping when a given combination of multiple relevant model parameters. We then propose a sample-based pre-training solution for estimating the optimal combination of parameters under a subset of selected edge servers.

4.2.2 Theoretic Analysis and Preliminary Experiments

We can derive the following theoretical upper bound on the difference of loss functions with and without server dropping.

Theorem 1. Suppose the following assumptions hold: (1) F_1, F_2, \dots, F_K are all L -smooth and μ -convex, i.e., $\frac{\mu}{2} \|\mathbf{w} - \mathbf{w}'\|^2 \leq F_k(\mathbf{w}) - F_k(\mathbf{w}') + (\mathbf{w} - \mathbf{w}')^T \nabla F_k(\mathbf{w}')$ $\leq \frac{L}{2} \|\mathbf{w} - \mathbf{w}'\|^2$ for all $\mathbf{w}, \mathbf{w}' \in \mathbb{R}^d$; (2) data samples x_k are uniformly randomly sampled from \mathcal{D}_k ; and (3) the stochastic gradient satisfies $\mathbb{E} \|\nabla l_k(\mathbf{w}, x_k)\|^2 \leq G^2$ and $\mathbb{E} \|\nabla l_k(\mathbf{w}, x_k) - \nabla F_k(\mathbf{w})\|^2 \leq \sigma_k^2$. If $\kappa = \frac{L}{\mu}$,

$\gamma = \max\{8\kappa, e\}$, and the learning rate η_t satisfies $\eta_t = \frac{2}{\mu(\gamma+t)}$, we have

$$\mathbb{E}[F(\mathbf{w}_{\mathcal{M},T})] - F^* \leq \frac{4\kappa}{\mu(\gamma+T)} \left(\sum_{k \in \mathcal{M}} \frac{p_k^2 \sigma_k^2}{n_k} + 8e^2 G^2 + C_{\mathcal{M}} \right) + LD_{\mathcal{M}}^*(\mathbf{w}) \quad (13)$$

where $\mathbf{w}_{\mathcal{M},T}$ is the model trained by edge servers in the subset \mathcal{M} after the T th local iteration, $F^* = \sum_{k \in \mathcal{K}} F_k^*$ is the global minimum value of objective function when all K edge servers are selected to participate in model training, and $C_{\mathcal{M}} = 6LD_{\mathcal{M}}^*(F) + \frac{\mu^2(\gamma+1)}{4} \|\mathbf{w}_0 - \mathbf{w}_{\mathcal{M}}^*\|^2$.

Proof: See Appendix A. \square

Inequality (13) provides an upper bound on the gap between the loss function trained with only a subset \mathcal{M} of edge servers and that with all K edge servers. This gap characterizes the maximum bias of the model trained by only a subset of edge servers, compared to the global optimal model trained with all the available edge servers. As mentioned earlier, under the given dataset and loss function, the value of loss can be used to characterize the prediction accuracy of the output model. In other words, the above gap of loss values can be used to capture the bias in the model prediction accuracy caused by the removal of edge servers. From inequality (13), we can observe that the gap is closely related to a set of key parameters including the subset of selected edge servers \mathcal{M} , minibatch size n_k , the number of local epochs e , and the total number of local SGD iterations T . In the rest of this section, we will try to jointly optimize all these parameters to minimize the overall runtime of model training when the bias of the model accuracy can be controlled within a maximum tolerable level.

By substituting $T = e\Gamma$, we can simplify (13) into the following form:

$$\mathbb{E}[F(\mathbf{w}_{\mathcal{M},T})] - F^* \leq \frac{1}{e\Gamma} \left(\sum_{k \in \mathcal{M}} \frac{A}{n_k} + Be^2 + C_{\mathcal{M}} \right) + D_{\mathcal{M}} \quad (14)$$

where $A = \frac{4\kappa \max_{k \in \mathcal{M}} p_k^2 \sigma_k^2}{\mu}$, $B = \frac{32\kappa G^2}{\mu}$, $C_{\mathcal{M}} = \frac{24\kappa LD_{\mathcal{M}}^*(F)}{\mu} + \mu\kappa(\gamma+1)\|\mathbf{w}_0 - \mathbf{w}_{\mathcal{M}}^*\|^2$ and $D_{\mathcal{M}} = LD_{\mathcal{M}}^*(\mathbf{w})$. Generally speaking, these values cannot be known by edge servers or by the coordinator. They are, however, important for estimating the overall model training time. In Section 4.3.1, we will discuss how to estimate these constants using a sample-based pre-training approach.

Substituting (14) into (8), we obtain the following sufficient condition for the convergence constraint of TS-FL-SC:

$$\frac{1}{e\Gamma} \left(\sum_{k \in \mathcal{M}} \frac{A}{n_k} + Be^2 + C_{\mathcal{M}} \right) + D_{\mathcal{M}} \leq \epsilon. \quad (15)$$

The above inequality can be utilized to calculate the possible combination of parameters that achieve a given target of model accuracy. By substituting (15) into (11), we can obtain the following approximated version of problem (P):

$$\begin{aligned} \text{(P1)} \quad & \min_{e, \mathbf{n}, \mathcal{M}} \mathbb{E}[\tilde{c}(e, \mathbf{n}, \mathcal{M})] \\ & \text{s.t.} \quad D_{\mathcal{M}} < \epsilon \text{ and } e, n_k \in \mathbb{Z}^+, \text{ for } \mathcal{M} \subseteq \mathcal{K} \end{aligned}$$

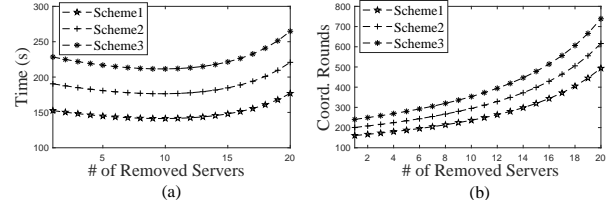


Fig. 4. (a) Time duration and (b) number of coordination rounds required to train a model with a target model accuracy when different numbers of slow edge servers have been removed from the training process. Three model training schemes have been compared: *Scheme 1* with $n_k = 200$ and $e_k = 80$, *Scheme 2* with $n_k = 200$ and $e_k = 100$, and *Scheme 3* with $n_k = 200$ and $e_k = 120$.

where

$$\mathbb{E}[\tilde{c}(e, \mathbf{n}, \mathcal{M})] = \frac{c_{\mathcal{M}}(e_{\mathcal{M}}, n_{\mathcal{M}}, b_{\mathcal{M},\varsigma}, \beta_{\mathcal{M},\varsigma}, \varsigma)}{e(\epsilon - D_{\mathcal{M}})} \left(\sum_{k \in \mathcal{M}} \frac{A}{n_k} + Be^2 + C_{\mathcal{M}} \right). \quad (16)$$

By exploiting the above reformulated problem (P1), we can calculate the optimal combination of parameters that minimize the overall runtime for training a model with the guaranteed output accuracy ϵ .

We can now evaluate the impact of the server dropping on the speed of the model training. In Fig. 4, we present the overall runtime and number of coordination rounds required to train a model with a guaranteed accuracy when different numbers of slowest edge servers are removed. We observe different trends when the required runtime and number of coordination rounds are evaluated for model training. This once again verifies our observations that these two performance metrics are fundamentally different in FEI systems. In particular, in Fig. 4(b), the required number of coordination rounds decreases when more and more edge servers as well as their corresponding data samples have been removed from participating in the model training. This result is aligned with previously reported results which suggest more data samples from edge servers can always help improving the convergence speed of the model training process [11]. In Fig. 4(a), however, we observe that the overall runtime for training a model with a guaranteed accuracy can be reduced when only a few slowest edge servers have been removed. As more and more edge servers have been dropped from the model training, the required model training time will eventually start to increase. This is because, by removing only a few slowest edge servers, the fast edge servers do not have to wait for these slow edge servers in each round of coordination. Therefore, despite the increase of the total number of required coordination rounds, the overall runtime to reach a target model accuracy can still be reduced. However, when more edge servers as well as their corresponding data samples have been removed from the model training, the adverse effect such as the slowed convergence as well as the bias in the resulting model caused by the removal of data samples will start to dominate the overall model training process.

4.3 Parameter Optimization

4.3.1 Sample-based Pre-training for Unknown Parameter Estimation

As mentioned earlier, the objective function $\mathbb{E}[\tilde{c}(e, n, \mathcal{M})]$ contains unknown constant parameters that may affect the time consumption of model training. These unknown parameters can be estimated by using a pre-training-based solution described as follows.

Estimation of Parameters A , B and $C_{\mathcal{M}}$: We adopt the sampling-based approach in [32] to estimate the unknown parameters that affect the training time. The basic idea is to empirically select a set of I different combinations of parameters $\langle e^{(i)}, n^{(i)} \rangle$ for $i \in \{1, \dots, I\}$ to pre-train an ML model and use the observed convergence results to estimate the unknown parameters including A , B and $C_{\mathcal{M}}$ and $D_{\mathcal{M}}$ in the (15). We use superscript (i) to denote the i -th model pre-training process based on a selection of empirical parameters $(e^{(i)}, n^{(i)})$ for $1 \leq i \leq I$. In the pre-training, the coordinator first pre-select two global loss values F_a and F_b and then records the required numbers of coordination rounds $R_a^{(i)}$ and $R_b^{(i)}$ to reach the target losses F_a and F_b from the initial global model w_0 when employing different combinations of empirically selected parameters. To reduce the time duration of the pre-training process, F_a and F_b can be set to relatively high loss values.

More formally, by applying the convergence bound in (14) to approximate the global loss value after $R_a^{(i)}$ and $R_b^{(i)}$ coordination rounds, we have the following results

$$F_a - F^* \approx \frac{1}{e_i R_a^{(i)}} \left(\frac{A}{n^{(i)}} + B e^{(i)2} + C_{\mathcal{M}} \right) + D_{\mathcal{M}}, \quad (17)$$

$$F_b - F^* \approx \frac{1}{e_i R_b^{(i)}} \left(\frac{A}{n^{(i)}} + B e^{(i)2} + C_{\mathcal{M}} \right) + D_{\mathcal{M}}. \quad (18)$$

Combining (17) and (18), we have

$$e^{(i)}(R_b^{(i)} - R_a^{(i)}) \approx \left(\frac{1}{F_b - F^* - D_{\mathcal{M}}} - \frac{1}{F_a - F^* - D_{\mathcal{M}}} \right) \left(\frac{A}{n^{(i)}} + B e^{(i)2} + C_{\mathcal{M}} \right). \quad (19)$$

By sampling different pairs of (i, j) for $i, j \in \mathcal{I}$ and $i \neq j$, we can have

$$\frac{e_i(R_b^{(i)} - R_a^{(i)})}{e_j(R_b^{(j)} - R_a^{(j)})} \approx \frac{\frac{A}{n^{(i)}} + B e^{(i)2} + C_{\mathcal{M}}}{\frac{A}{n^{(j)}} + B \left(1 + \frac{1}{s^{(j)}}\right) e^{(i)2} + C_{\mathcal{M}}}. \quad (20)$$

Reorganizing (20), we can write

$$A \left(\frac{1}{n^{(i)}} - \chi_{i,j} \frac{1}{n^{(j)}} \right) + B e^{(i)2} - \chi_{i,j} \left(1 + \frac{1}{s^{(j)}} \right) e^{(i)2} \approx C_{\mathcal{M}} (\chi_{i,j} - 1). \quad (21)$$

where $\chi_{i,j} \triangleq \frac{e^{(i)}(R_b^{(i)} - R_a^{(i)})}{e^{(j)}(R_b^{(j)} - R_a^{(j)})}$. Repeating the above steps for all the different combinations of parameters, we can eventually obtain estimated solutions for the unknown constants A , B and $C_{\mathcal{M}}$ using standard linear fitting method, e.g., least square method.

Estimation of Parameter $D_{\mathcal{M}}$: Based on the estimated parameters A , B and $C_{\mathcal{M}}$, we can calculate $D_{\mathcal{M}}$ as follows:

$$D_{\mathcal{M}} \approx F_b - F^* - \frac{1}{e_i R_b^{(i)}} \left(\frac{A}{n^{(i)}} + B e^{(i)2} + C_{\mathcal{M}} \right). \quad (22)$$

Algorithm 1 Sample-based Pre-training Algorithm

- 1: **Input:** Target loss values: F_a, F_b , and I empirically selected combinations of training parameters, the subset \mathcal{M} of edge servers;
 - 2: **Output:** $A, B, C_{\mathcal{M}}, D_{\mathcal{M}}, \zeta, \alpha_k, \beta_k$, and u ;
 - 3: **for** $i = \{1, \dots, I\}$ **do**
 - 4: Empirically choose $(e^{(i)}, n^{(i)})$ for each edge server $k \in \mathcal{M}$;
 - 5: Record values $R_a^{(i)}$ and $R_b^{(i)}$ when target loss F_a and F_b are reached;
 - 6: Record the average time duration $\bar{c}_k(e^{(i)}, n^{(i)})$ of edge server k to complete each round of model coordination;
 - 7: **end for**
 - 8: Estimate the unknown parameters $A, B, C_{\mathcal{M}}$, and $D_{\mathcal{M}}$ using (21);
 - 9: Estimate the unknown parameters ζ, α_k, β_k , and u using (24);
 - 10: **Return** $A, B, C_{\mathcal{M}}, D_{\mathcal{M}}, \zeta, \alpha_k, \beta_k$, and u ;
-

Estimation of Parameters $\zeta, \alpha_k, \beta_k, \beta_{k,\varsigma}$, and u : To define the local model training delay $(\nu_k(e, n_k, b_{k,\varsigma}, \beta_{k,\varsigma}, \varsigma) + u)$, we adopt a commonly adopted setting [38] and assume the delay under fixed hardware and software environments, can be represented into the following form:

$$\nu_k(e_k, n_k, b_{k,\varsigma}, \beta_{k,\varsigma}, \varsigma) = e_k(b_{k,\varsigma} n_k + \beta_{k,\varsigma}). \quad (23)$$

Suppose the coordinator can record the time consumption of each coordination round for each selected edge server. We can therefore obtain the average time consumption $\bar{c}_k(e^{(i)}, n^{(i)})$ under a given pair of parameters $(e^{(i)}, n^{(i)})$ for $i \in \{1, 2, \dots, I\}$. We can then have

$$\zeta + \alpha_k e^{(i)} n^{(i)} + \beta_k e^{(i)} + u \approx \bar{c}_k(e^{(i)}, n^{(i)}), \quad \forall i \in \{1, 2, \dots, I\}. \quad (24)$$

By jointly solving all I equations in (24), we can obtain estimated values of unknown parameter $\zeta, \alpha_k = a_{k,\varsigma} + b_{k,\varsigma}, \beta_k = \beta_{k,\varsigma}$, and u . The detailed procedures of the pre-training for parameter estimation is presented in Algorithm 1.

Algorithm 2 Alternate Convex Search Algorithm

- 1: **Input:** Parameters: $A, B, C_{\mathcal{M}}, D_{\mathcal{M}}, \zeta, \alpha_k, \beta_k, u$;
 - 2: **Initialization:** Initial model parameters $z_0 = (e_0, n_0)$; Stopping criterion θ ; Domains \mathcal{Z}_n and \mathcal{Z}_e ; $i \leftarrow 0$;
 - 3: **Output:** Solution (e^*, n^*, \mathcal{M}) ;
 - 4: **while** $\mathbb{E}[\tilde{c}(z_{i-1}, \mathcal{M})] - \mathbb{E}[\tilde{c}(z_i, \mathcal{M})] \geq \theta$ **do**
 - 5: Find e^* of $\tilde{c}(e, n_i, \mathcal{M})$ in the search domain \mathcal{Z}_e when give s_i and n_i as well as \mathcal{M} and perform $e_{i+1} \leftarrow e^*$;
 - 6: Find n^* of $\tilde{c}(e_{i+1}, n, \mathcal{M})$ in the search domain \mathcal{Z}_n when given \mathcal{M} and perform $n^{i+1} \leftarrow n^*$;
 - 7: $z_{i+1} \leftarrow (e_{i+1}, n_{i+1}), i \leftarrow i + 1$;
 - 8: **end while**
 - 9: **Return** the solution (e_i, n_i, \mathcal{M}) .
-

4.3.2 Parameter Optimization Algorithm

As mentioned earlier, the model training time duration is closely related to the key training parameters, especially n_k and e_k . Unfortunately, these two parameters are entangled

with each other in affecting the overall model training time. For example, reducing the local epoch number e_k at each edge server k may accelerate the convergence of global model. It will however increase the frequency of model coordination, resulting in high communication overhead and coordination delay. Similarly, loading more training data samples $e_k n_k$ and choosing a large mini-batch size n_k at each training iteration at an edge server k will reduce the total number of required iterations for the model to converge to a satisfactory result. It will however slow down the data loading time and also extend computational time at each iteration of the local model training process. Therefore, it is generally difficult to jointly optimize both n_k and e_k to minimize the overall time duration for training a satisfactory model. Fortunately, we can prove that the objective function of problem **(P1)** is a biconvex optimization problem when $n_k = n$ for every $k \in \mathcal{M}$ under a given subset \mathcal{M} of edge servers. In particular, we can prove the following result.

Lemma 1. Suppose $n_k = n$ for every $k \in \mathcal{K}$ and changing n does not affect the rank of model training times among edge servers, problem **(P1)** is biconvex over e and n .

Proof: See Appendix B. \square

Motivated by the above observation, we can adopt a commonly adopted iterative optimization method called alternate convex search (ACS) to calculate the optimal solution of e^* and n^* to minimize the objective function $\mathbb{E}[\tilde{c}(e, \mathbf{n}, \mathcal{M})]$. We present the detailed procedures of the ACS approach in Algorithm 2.

5 TS-FL WITH ASYNCHRONOUS COORDINATION

As mentioned earlier, TS-FL-SC with server dropping may result in a biased model at the end of the model training process and therefore may not be suitable for the scenarios when the data samples at the slowest edge server possess unique characteristics that are important for the model. In this section, we investigate model training acceleration solutions for TS-FL-ASC in which each edge server can upload its local model to the coordinator right after finishing the required number of local training iterations and can receive an instantaneous update of the global model from the coordinator.

5.1 Modeling the Staleness Effect

Since TS-FL-ASC allows each edge server to obtain an instantaneous update of the global model at any time during the training process, it will have the potential to avoid waiting for the slow edge servers. Unfortunately, it is known that the convergence performance of the asynchronous coordination is adversely affected by the so-called *staleness effect*, that is the gap between the model updating iterations at the slowest edge server and that at the fastest edge server increases with the total number of iterations, resulting in slow convergence or even divergence of the global model in each round of model aggregation.

To quantify the impact of the staleness effect on the convergence of model training, we introduce the following virtual global model updating sequence $\{\bar{\mathbf{w}}_t\}_{t=1, \dots, T}$ as an equivalent form of a centralized mini-batch SGD process

updated by a set of decentralized edge servers following the asynchronous coordination as TS-FL-ASC, i.e., we define the virtual sequence of TS-FL-ASC as follows:

$$\bar{\mathbf{w}}_0 \triangleq \mathbf{w}_0, \quad (25)$$

$$\bar{\mathbf{w}}_t \triangleq \mathbf{w}_0 - \sum_{k=1}^K p_k \sum_{j=0}^{t-1} \eta_j \mathbf{g}_{k,j}. \quad (26)$$

We can observe that $\bar{\mathbf{w}}_t$ satisfies $\bar{\mathbf{w}}_t = \frac{1}{K} \sum_{k=1}^K p_k \mathbf{w}_{k,t}$. Note that the virtual sequence is only introduced for our analysis and it does not need to be calculated explicitly.

We also define an upper bound τ of staleness as the maximum gap between the numbers of iterations locally performed by different edge servers, i.e., we introduce the following constraint for TS-FL-ASC:

$$\max_{h \in \mathcal{K}} \Lambda(\mathcal{W}_{k,t}^h) - \min_{h \in \mathcal{K}} \Lambda(\mathcal{W}_{k,t}^h) \leq \tau \quad (27)$$

where $\Lambda(\mathcal{W}_{k,t}^h)$ is the index of local iterations reported by edge server h to the coordinator when edge server k uploads its local model in the t th iteration.

We can then prove the following result.

Lemma 2. Suppose x_k is data samples following distribution \mathcal{D}_k . We assume the following conditions hold: (1) the stochastic gradient satisfies $\mathbb{E}\|\nabla l_k(\mathbf{w}, x_k)\|^2 \leq G^2$; (2) the staleness is upper bounded by (27); (3) learning rate η_t is non-increasing and satisfies $\eta_t \leq 2\eta_{H+\tau+t}$ for all $t > 0$ where $H \triangleq \max_{k \in \mathcal{K}} e_k$. The gap between the local model trained at edge server k and the (virtual) global model is upper bounded by

$$\mathbb{E}(\|\bar{\mathbf{w}}_t - \mathbf{w}_{k,t}\|^2) \leq 24\eta_t^2 G^2 (2\tau^2 + e_k^2). \quad (28)$$

Similarly, we can write the upper bound of the gap between weighted sum of all the local models at edge servers and the global model as follows:

$$\mathbb{E}\left(\sum_{k=1}^K p_k \|\bar{\mathbf{w}}_t - \mathbf{w}_{k,t}\|^2\right) \leq 24\eta_t^2 G^2 (2\tau^2 + \sum_{k=1}^K p_k e_k^2) \quad (29)$$

where which represents the weighted average of models of all edge servers after the t th SGD step.

Proof: See Appendix C. \square

We can observe from the above result that the upper bound of the staleness effect τ directly affects the convergence of the model training process.

Based on the above lemma, we can prove the following result about the convergence of TS-FL-ASC.

Theorem 2. Suppose the following assumptions hold: (1) F_1, F_2, \dots, F_K are all L -smooth and μ -convex, i.e., $\frac{\mu}{2} \|\mathbf{w} - \mathbf{w}'\|^2 \leq F_k(\mathbf{w}) - F_k(\mathbf{w}') + (\mathbf{w} - \mathbf{w}')^T \nabla F_k(\mathbf{w}') \leq \frac{L}{2} \|\mathbf{w} - \mathbf{w}'\|^2$ for all $\mathbf{w}, \mathbf{w}' \in \mathbb{R}^d$; (2) Data sample x_k is uniformly randomly sampled from \mathcal{D}_k ; and (3) the stochastic gradient satisfies $\mathbb{E}\|\nabla l_k(\mathbf{w}, x_k)\|^2 \leq G^2$ and $\mathbb{E}\|\nabla l_k(\mathbf{w}, x_k) - \nabla F_k(\mathbf{w})\|^2 \leq \sigma_k^2$. By setting $\kappa = \frac{L}{\mu}$, $\gamma = \max\{8\kappa, \tau + H\}$ and $\eta_t = \frac{2}{\mu(\gamma+t)}$, we have

$$\mathbb{E}[F(\mathbf{w}_T) - F^*] \leq \frac{\kappa}{\gamma + T} \left(\frac{2B}{\mu} + \frac{\mu\gamma}{2} \mathbb{E}\|\mathbf{w}_0 - \mathbf{w}^*\|^2 \right) \quad (30)$$

where $B = \sum_{k=1}^K \frac{p_k \sigma_k^2}{n_k} + 6L(F^* - \sum_{k=1}^K p_k F_k^*) + 48G^2(2\tau^2 + \sum_{k=1}^K p_k e_k^2)$, \mathbf{w}^* is the global optimal model,

\mathbf{w}_T is weighted sum of all the local models trained by edge servers after T iterations for $T \in \mathcal{T}^k$ and $k \in \mathcal{K}$.

Proof: See Appendix D. \square

From the above result, we can also observe that, when $\tau \geq \sqrt{T}$, the upper bound of convergence result in (30) may not converge when the number of iterations T becomes large. This once again verifies our previous result that increasing the upper bound of the staleness τ may result in degradation of convergence speed of the model training process.

One potential solution to alleviate the staleness effect is to decrease the model training and updating speed of the fast edge servers to reduce the upper bound of the staleness within a certain limit. This however will result in more time consumption in the model training process and therefore may not be suitable for some time-sensitive applications.

5.2 Load Forwarding

In this subsection, we propose a load forwarding-based solution to reduce the staleness by allowing some slow edge servers to offload part of its data samples to some trusted edge servers with high processing capability. To simplify our description and also due to the fact that, in many practical systems, allowing a slow edge server to reveal its data samples to multiple fast edge servers may raise security concern, in the rest of this section, we mainly focus on the load forwarding solution within a trusted edge server pair consisting of a relatively fast edge server and a slower edge server. This is reasonable for many practical network systems in which, to further improve the robustness of its networking systems, a single network operator or service provider may sign contract with at least one other service provider for resource sharing when necessary. For example, major telecommunication operators in US and Europe have already signed contract with at least one other operator for various forms of resource sharing including infrastructure and spectrum sharing [41]. We write an edge server pair as $\langle k, k' \rangle$ where, without loss of generality, we label the slow edge server as k and fast edge server as k' . We abuse the notation and use k to denote edge server pair $\langle k, k' \rangle$. Let \mathcal{P} be the set of edge server pairs that support load forwarding, i.e., $k \in \mathcal{P}$.

Note that in each edge server pair, during each round of model coordination, the slow edge server only needs to send a batch of its local data samples to the fast edge server. The fast edge server k' can then directly obtain the updated model from the coordinator, perform model training based on the received batch of data samples, and finally upload the updated model to the coordinator. In other words, the fast edge server will take over the local training process for the slow edge servers without requiring any model transfer within the edge server pair. We define the data communication delay caused by forwarding a batch of data samples from edge servers k to k' as $o_{k,k'} = \lambda_k n_k e_k$ where λ_k is the data forwarding speed for sending each data sample from edge servers k' to k .

We write $\alpha_{k,k'}$ as the portion of model updating rounds of the slow edge server k that are helped by the fast edge

server k' . We can then write the overall time consumption of model training with load forwarding as

$$c(\mathbf{e}, \mathbf{n}, \mathbf{\Gamma}, \mathcal{P}, \boldsymbol{\alpha}) = \max_{k \in \mathcal{P}} \{ \Gamma_k (1 - \alpha_{k,k'}) c_k, (\Gamma_{k'} c_{k'} + \Gamma_k \alpha_{k,k'} c_{k,k'}) \} \quad (31)$$

where $\mathbf{e} = \langle e_k \rangle_{k \in \mathcal{K}}$, $\mathbf{n} = \langle n_k \rangle_{k \in \mathcal{K}}$, $\boldsymbol{\alpha} = \langle \alpha_{k,k'} \rangle_{k \in \mathcal{K}}$, $\mathbf{\Gamma} = \langle \Gamma_k \rangle_{k \in \mathcal{K}}$, $c_k = \zeta + q_k(n_k, e_k) + \nu_k(e_k, n_k) + u$ is the time required to complete each round of model updating, and $c_{k,k'} = (\zeta + q_k(n_k, e_k) + o_{k,k'}(n_k, e_k) + \nu_{k'}(e_k, n_k) + u)$ is overall time duration for edge server k' to help edge server k in training its local model.

We can observe that in an ideal condition, after applying the load forwarding, the model training time of fast and slow edge servers in each edge server pair becomes equal, i.e., $\Gamma_k (1 - \alpha_{k,k'}) c_k = \Gamma_{k'} c_{k'} + \Gamma_k \alpha_{k,k'} c_{k,k'}$. In this case, we can write the optimal value $\alpha_{k,k'}^*$ as

$$\alpha_{k,k'}^* = \frac{\Gamma_k c_k - \Gamma_{k'} c_{k'}}{\Gamma_k (c_k + c_{k,k'})}, \quad \forall k \in \mathcal{P}. \quad (32)$$

We can therefore rewrite the original optimization problem **(P)** as follows:

$$\begin{aligned} \text{(P2)} \quad & \min_{\mathbf{e}, \mathbf{n}, \mathbf{\Gamma}, \mathcal{P}, \boldsymbol{\alpha}} \mathbb{E}[c(\mathbf{e}, \mathbf{n}, \mathbf{\Gamma}, \mathcal{P}, \boldsymbol{\alpha})] \\ & \text{s.t.} \quad \mathbb{E}[F(\mathbf{w}_{\mathcal{P}, \mathbf{\Gamma}})] - F^* \leq \epsilon, \end{aligned} \quad (33)$$

$$e_k, \Gamma_k, n_k \in \mathbb{Z}^+, \quad (34)$$

$$0 \leq \alpha_{k,k'} \leq 1. \quad (35)$$

where $\mathbf{w}_{\mathcal{P}, \mathbf{\Gamma}}$ is the model trained by a set \mathcal{P} of edge server pairs in iteration $\mathbf{\Gamma}$.

In the rest of this section, we discuss how to solve problem **(P2)** using server dropping and parameter optimization.

5.3 Server Dropping

Similar to TS-FL-SC, it is also possible to further reduce the model training time in TS-FL-ASC by removing some slow edge servers from participating in the model training process.

We can prove the following upper bound of the convergence rate of TS-FL-ASC when a subset \mathcal{M} of edge servers are selected to participate in the model training.

Theorem 3. Suppose the following assumptions hold: (1) F_1, F_2, \dots, F_K are all L -smooth and μ -convex, i.e., $\frac{\mu}{2} \|\mathbf{w} - \mathbf{w}'\|^2 \leq F_k(\mathbf{w}) - F_k(\mathbf{w}') + (\mathbf{w} - \mathbf{w}')^T \nabla F_k(\mathbf{w}') \leq \frac{L}{2} \|\mathbf{w} - \mathbf{w}'\|^2$ for all $\mathbf{w}, \mathbf{w}' \in \mathbb{R}^d$; (2) data samples x_k are uniformly and randomly sampled from \mathcal{D}_k ; (3) the stochastic gradient satisfies $\mathbb{E} \|\nabla l_k(\mathbf{w}, x_k) - \nabla F_k(\mathbf{w})\|^2 \leq \sigma_k^2$ and $\mathbb{E} \|\nabla l_k(\mathbf{w}, x_k)\|^2 \leq G^2$; and (4) The staleness is bounded by (27). If $\kappa = \frac{L}{\mu}$, $\gamma = \max\{8\kappa, \tau + H\}$, and $\eta_t = \frac{2}{\mu(\gamma+t)}$, then we have

$$\begin{aligned} \mathbb{E}[F(\mathbf{w}_{\mathcal{M}, T})] - F^* & \leq LD_{\mathcal{M}}^*(\mathbf{w}) \\ & + \frac{4\kappa}{\mu(\gamma + T)} \left(\sum_{k \in \mathcal{M}} \frac{p_k^2 \sigma_k^2}{n_k} + 48 \sum_{k \in \mathcal{M}} p_k e_k^2 G^2 + C_{\mathcal{M}} \right) \end{aligned} \quad (36)$$

where $C_{\mathcal{M}} = 96G^2\tau^2 + 6LD_{\mathcal{M}}^*(F) + \frac{\mu^2\gamma}{4} \|\mathbf{w}_0 - \mathbf{w}_{\mathcal{M}}^*\|^2$ and $D_{\mathcal{M}}^*(\mathbf{w})$ is the difference of objective function between models collaboratively trained by edge server sets \mathcal{K} and \mathcal{M} .

Proof: See Appendix E. \square

Theorem 3 characterizes the impact of both server dropping and staleness bound τ on the convergence performance of the FEI system. We can also observe that the convergence upper bound in TS-FL-ASC is relatively more loose, compared to that of TS-FL-SC presented in Theorem 1, due to the staleness effect which is reflected by τ in the first term of $C_{\mathcal{M}}$.

We can then follow the same line as TS-FL-SC to model the overall time duration of training a satisfactory model using TS-FL-ASC as follows. We first rewrite (36) in 3 into the following simplified form:

$$\begin{aligned} \mathbb{E}[F(\mathbf{w}_{\mathcal{M},T})] - F^* & \\ & \leq \frac{1}{T} \left(\sum_{k \in \mathcal{M}} \frac{A'}{n_k} + B' \sum_{k \in \mathcal{M}} p_k e_k^2 + C'_{\mathcal{M}} \right) + D'_{\mathcal{M}} \end{aligned} \quad (37)$$

where $A' = \frac{4\kappa \max_{k \in \mathcal{K}} p_k^2 \sigma_k^2}{\mu}$, $B' = \frac{192\kappa G^2}{\mu}$, $C'_{\mathcal{M}} = \frac{384\kappa G^2 \tau^2}{\mu} + \frac{24\kappa L D_{\mathcal{M}}^*(F)}{\mu} + \mu\kappa\gamma \|\mathbf{w}_0 - \mathbf{w}_{\mathcal{M}}^*\|^2$ and $D'_{\mathcal{M}} = L D_{\mathcal{M}}^*(\mathbf{w})$.

By substituting (37) into constraint (33), we can obtain the following constraint of TS-FL-ASC:

$$\frac{1}{T} \left(\sum_{k \in \mathcal{M}} \frac{A'}{n_k} + B' \sum_{k \in \mathcal{M}} p_k e_k^2 + C'_{\mathcal{M}} \right) + D'_{\mathcal{M}} \leq \epsilon. \quad (38)$$

By replacing the constraint (33) in Problem (P3) with (38), we can obtain an approximated version of problem (P3) as follows:

$$\begin{aligned} \text{(P4)} \quad \min_{e, n} \quad & \mathbb{E}[\tilde{c}(e, \mathbf{n}, \mathcal{M}, \alpha)] \\ \text{s.t.} \quad & D'_{\mathcal{M}} < \epsilon, \text{ and } 0 \leq \alpha_{k,k'} \leq 1 \text{ and } e_k, n_k \in \mathbb{Z}^+, \end{aligned} \quad (39)$$

where $\mathbb{E}[\tilde{c}(e, \mathbf{n}, \mathcal{M}, \alpha)]$ is given by

$$\begin{aligned} \mathbb{E}[\tilde{c}(e, \mathbf{n}, \mathcal{M}, \alpha)] = & \frac{\frac{A'}{n_k} + B' \sum_{k \in \mathcal{M}} p_k e_k^2 + C'_{\mathcal{M}}}{\epsilon - D'_{\mathcal{M}}} \\ & \left(\frac{(1 - \alpha_{M,M'}) c_M(e_M, n_M)}{e_M} + \frac{\alpha_{M',M} c_{M',M}(e_{M'}, n_{M'})}{e_{M'}} \right) \end{aligned} \quad (40)$$

and M is the slowest edge server pair to finish each round of model updating for $1 \leq M \leq |\mathcal{P}|$.

5.4 Parameter Optimization

5.4.1 Unknown Parameter Estimation via Pre-training

To solve problem (P4), we need to first estimate the impact of various parameters on the convergence of model training. We follow the same line as TS-FL-SC to estimate these parameters by pre-running model training process with different combinations of parameters.

Estimation of Parameters A' , B' , $C'_{\mathcal{M}}$, $D'_{\mathcal{M}}$: We empirically select I different combinations of parameters $\langle e^{(i)}, n^{(i)} \rangle$ for $i \in \{1, \dots, I\}$ to pre-train the ML model and use the observed convergence results to estimate these parameters. Under any given training parameters $\langle e^{(i)}, n^{(i)} \rangle$, the convergence upper bound of TS-FL-ASC can be rewrite as:

$$\mathbb{E}[F(\mathbf{w}_T)] - F^* \leq \frac{1}{e\Gamma} \left(\frac{A'}{n^{(i)}} + B' e^{(i)2} + C'_{\mathcal{M}} \right) + D'_{\mathcal{M}} \quad (41)$$

where $\Gamma_k = \Gamma = \frac{T}{e^{(i)}}$ for all $k \in \mathcal{M}$. We can observe that the above upper bound is in the same form as (14). We

can therefore use the same parameter estimation procedures described in Section 4.3.1 to estimate A' , B' , $C'_{\mathcal{M}}$, $D'_{\mathcal{M}}$. We omit the details due to the space limit.

Estimation of Parameters ζ , $a_{k,\varsigma}$, $b_{k,\varsigma}$, u : These parameters are closely related to the computational and communication capacity of edge servers and can be obtained by adopting the same procedures discussed in Section 4.3.1 and Algorithm 1. We again omit the details due to the limit of space.

5.4.2 Parameter Optimization Algorithm

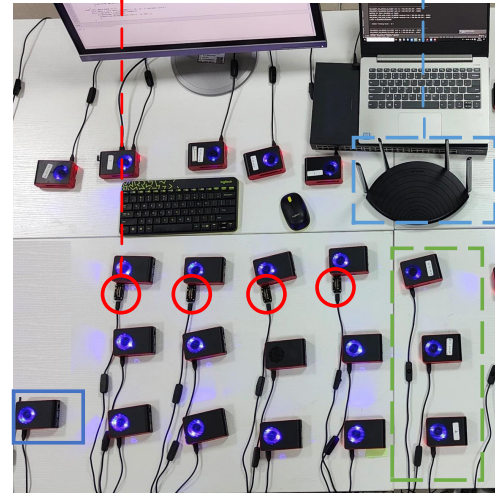
Similar to the TS-FL-SC, we can again prove that the objective function of problem (40) is a biconvex optimization problem when $e_k = e$ and n_k for every server $k \in \mathcal{M}$ and the subset \mathcal{M} edge servers participating in the model training is fixed. The following lemma can be proved by following the same line as Lemma 1. We omit the proof due to the limit of space.

Lemma 3. Suppose $n_k = n$ and $e_k = e$ for every $k \in \mathcal{K}$ and changing n does not affects the rank of model training times of edge servers, problem (40) is biconvex over e and n .

We can again adopt an ACS-based approach to calculate the optimal solution of e^* and n^* to minimize the overall model training time $\mathbb{E}[\tilde{c}(e, \mathbf{n}, \mathcal{M}, \alpha)]$ for TS-FL-ASC.

6 NUMERICAL RESULT

Training Process Tracking Wi-Fi Router



Coordinator 17 Raspberry Pis 3 Raspberry Pis
(Version 4B) (Version 3A)

Fig. 5. Hardware platform consisting of 20 Raspberry Pis serving as edge servers and one Raspberry Pi serving as the coordinator. Edge servers and coordinator are connected with each other via Wi-Fi links.

6.1 Experimental Setup

6.1.1 Hardware Platform

To evaluate the performance of our proposed solutions, we develop a hardware platform consisting of 20 Raspberry Pi mini-computers as edge servers, including 17 version

TABLE 1
Experimental Setup

Architecture Setup	Model Type	
	Multinomial Logistic Regression	
Dataset	MNIST	Traffic
Input Size	784×1	2500×1
Output Size	10×1	6×1
Activation Function	Sigmoid	
Optimizer	SGD, learning rate 0.01 with decay rate 0.995	

4B with a Quad-core Cortex-A72 (ARM V8) 64-bit SoC operated on 1.5GHz, and 3 version 3A with a Quad-core Cortex-A53 (ARM V8) 64-bit SoC operated on 1.4GHz, as illustrated in Figure 5. All the edge servers are connected to a coordinator (another Raspberry Pi version 4B) via a TP-LINK Wi-Fi Router. To estimate the time duration of the model training process, we installed a multi-function USB multi-meter POWER-Z KM001C to the power port of each edge server to measure and keep track of voltage, current, and power during the entire model training process. We set the sampling rate of each multi-meter to 1 kHz.

6.1.2 Experimental Setup

We conduct our experiment based on two commonly used datasets: (1) handwritten digit dataset, called MNIST [42], which contains 60,000 images training data samples and 10,000 testing data samples, and (2) real world network traffic dataset, called ISCX VPN-nonVPN, which consists of 60,000 traffic data samples for model training and other 12,000 for testing [43]. To guarantee the convex property of the classification problem, we follow a commonly adopted setting and use multinomial logistic regression model. For all the experiments, the global model is initialized with the same weight w_0 and the learning rate is set to $\eta_0 = 0.1$ with a fixed decay 0.995 per round. To simulate the data heterogeneity scenario, each edge server only has data samples associated with a limited number of randomly selected labels, e.g., in MNIST dataset, each edge server only has images of a limited number of digits, and in traffic dataset, each edge server can only store a limited number of types of traffic data samples. In this section, we compare five different scenarios:

Scenario (a): we consider handwritten digit dataset and assume each edge server only has data samples associated with a single digit and the data samples of the three low-performance edge servers, i.e., the stragglers (Raspberry Pi 3A), are associated with non-unique digits, i.e., there exist other edge servers (Raspberry Pi 4B) with data samples of the same digits. Since MNIST dataset only has 10 digits 0-9 and we have 20 edge servers, at least two edge servers have data samples associated with the same digit. In this case, even if all three slow edge servers have been removed from participating the model training, there are still able to find data samples associated with the removed digits in the rest of the high-performance edge servers.

Scenario (b): we follow the same setting as Scenario (a) but the three low-performance edge servers (Raspberry Pi 3A) consist of data samples associated with unique digits. In this case, if these low-performance edge servers have been removed from the training process, there is impossible to

find data samples with the removed digits in the rest of the edge servers.

Scenario (c): we consider handwritten digit dataset and assume each edge server has data samples randomly sampled from any five digits and the data samples of the three low-performance edge servers are associated with non-unique digits.

Scenario (d): we consider traffic dataset and assume each edge server has data samples randomly sampled from any two types of data traffic and the data samples of the three low-performance edge servers are associated with non-unique types of data traffics.

Scenario (e): we consider traffic dataset and assume each edge server has data samples randomly sampled from any four types of data traffic and the data samples of the three low-performance edge servers are associated with non-unique types of data traffics.

6.2 Measuring Time Consumption of Model Training

We record data traces of the power dynamics during the FL model training process performed at each edge server. We can observe a clear dynamic pattern being repeated at each round of model training. We can also observe a clear difference in the power levels at different steps of the model training process as illustrated in Fig. 6(a). In particular, we can observe the four training steps in each round of model training process which are described as follows.

Step (1)–Time duration of model distribution: The coordinator broadcasts the updated global model to a selected subset of the edge servers at the beginning of each round of model training process. In Fig. 6(a), we can observe that the time consumption of the model distribution through a wireless network (e.g. 2.4GHz WiFi in our case) is approximately $\zeta = 0.2$ sec.

Step (2)–Time duration of data uploading: To simulate the delay caused by uploading of data samples from the data collecting devices to the edge servers, we set the data arrival rate at each high-performance edge server to 50MB/s (the common data rate of Wi-Fi links) and that at each low-performance edge server to 5MB/s (the common data rate for Bluetooth connections). Since, in our experiment, all the dataset samples have been pre-loaded to each edge server, to simulate the data uploading delay, we set a pre-calculated idle time at the beginning of every round of model training process as shown in Fig. 6(a).

Step (3)–Time duration of local model training: Once edge server k obtains the model distributed by the coordinator, it will perform e local SGD iterations to update its model. In our data traces in Fig. 6(a), we can observe that each local iteration performed by an edge server corresponds to a peak in power level. For e local iterations, we can observe e peaks during the local model training step. The value of n_k mainly affects the duration of each local iteration (time duration between two consecutive peaks) as shown in Fig. 6(b) and Fig. 6(c). In Table 2, we present the time duration of local training step under different combinations of e and n_k at different edge servers (i.e. Raspberry Pi (version 4B) and (version 3A)) based on different datasets. We can observe that for a fixed n_k , the time duration of local model training step increases almost linearly with e . Similarly, when e is

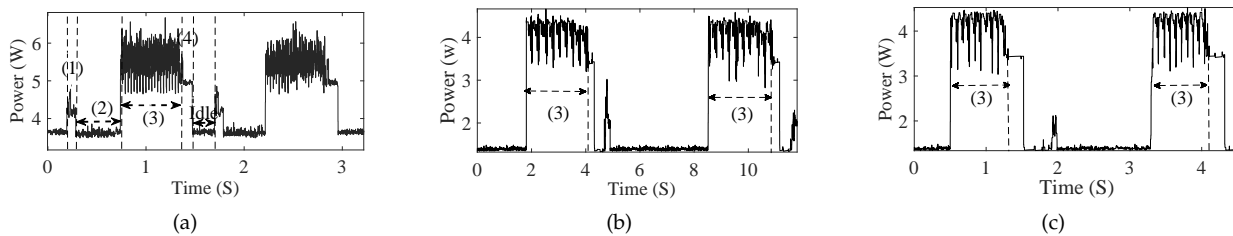


Fig. 6. (a) Recorded power tracking traces of an edge server (Raspberry Pi 4B) with four highlighted steps in each round of model coordination: (1) model distribution, (2) data uploading, (3) local model training, and (4) model uploading. (b) Comparison of time duration in step (3) under training parameters ($e = 10, n = 3000$), and (c) ($e = 10, n = 1000$).

TABLE 2
Recorded time duration (in sec) of local model training step (step (3)) under different setups

Raspberry Pi (4B)	(n, e)	(400,10)	(400,20)	(400,30)	(1000,10)	(1000,20)	(1000,30)
	MNIST		0.0618	0.1236	0.1856	0.1472	0.2908
Traffic		0.3700	0.7402	1.1092	0.4749	0.9473	1.4191

Raspberry Pi (3A)	(n, e)	(400,10)	(400,20)	(400,30)	(1000,10)	(1000,20)	(1000,30)
	MNIST		0.3815	0.7615	1.1408	0.8013	1.6003
Traffic		6.0476	12.0934	18.1386	6.5669	13.1309	19.6958

TABLE 3
Fitted parameters in the time duration model of local model training step

Raspberry Pi (4B)	Dataset type ς	$b_{k,\varsigma}$	β_k
	MNIST	$1.4 * 10^{-5}$	$5.2 * 10^{-4}$
	Traffic	$1.7 * 10^{-5}$	0.03

Raspberry Pi (3A)	Dataset type	$b_{k,\varsigma}$	β_k
	MNIST	$7 * 10^{-5}$	0.01
	Traffic	$8.6 * 10^{-5}$	0.57

fixed, the time consumption of each local iteration increases almost linearly with n_k . To summarize, the time duration of local model training step can be modeled as

$$\nu_k(e_k, n_k | \varsigma) = e_k(b_{k,\varsigma}n_k + \beta_k). \quad (42)$$

where $b_{k,\varsigma}$ characterizes the time duration for edge server k to process each training data sample and β_k is the time duration for performing local SGD and updating the local models. We present the fitted parameters of $b_{k,\varsigma}$ and β_k under different setup in Table 3.

Step (4)–Time duration of model uploading: Each edge server uploads its local model to the coordinator after e local SGD iterations. In our experiment, all edge servers and the coordinator are connected to a Wi-Fi router, we can observe that the time duration of model uploading does not affected by model training parameters such as n_k or e . We can therefore assume the time duration of this step is a constant, i.e., $u = 0.2s$ as observed in Fig. 6(a).

6.3 Experimental Results

As mentioned earlier, the convergence rate and time consumption of model training are two different concepts. The later is more complex and can be affected jointly by computation and communication delays influenced by multiple parameters. In Fig. 6.3, we compare the required number of coordination rounds and time duration of training a

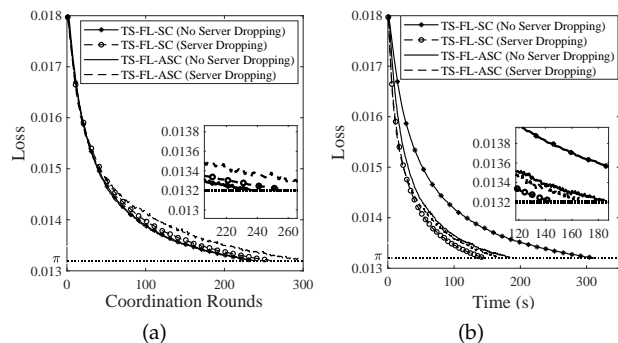


Fig. 7. (a) Number of coordination rounds and (b) model training time of TS-FL-SC and TS-FL-ASC under scenario (a).

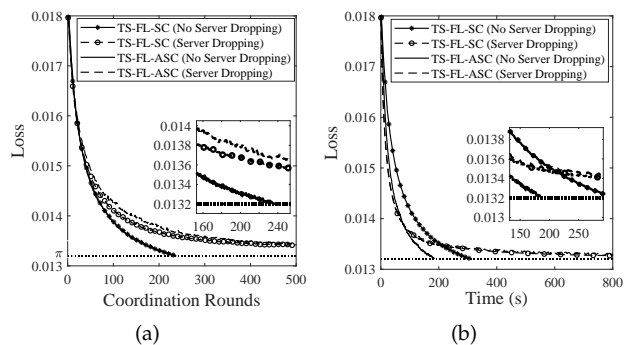


Fig. 8. (a) Number of coordination rounds and (b) model training time of TS-FL-SC and TS-FL-ASC under scenario (b).

model at a target loss ($\pi = 0.0132$) (corresponding to the model accuracy of 83.30% and 83.50% for MNIST and traffic datasets, respectively) using TS-FL-SC and TS-FL-ASC with and without using server dropping under scenario (a) where the low-performance edge servers do not have any uniquely labeled data samples. We can observe that the values of

TABLE 4
COMPARISON OF MODEL TRAINING TIME OF SCHEDULING-BASED SOLUTION AND SERVER DROPPING

Datasets	(e, n)	(10, 200)	(20, 200)	(30, 200)	(40, 200)	(50, 200)	(60, 200)
MNIST	Sched.-based	293.12	269.64	402.23	534.51	667.76	799.39
	Server Drop.	162.67	137.58	152.12	175.38	201.13	228.13
	Time Saving	130.45	132.06	250.11	359.13	466.63	571.26
Traffic	Sched.-based	1372.32	2835.49	4189.72	5550.56	6915.60	8280.64
	Server Drop.	288.37	311.64	345.50	382.40	420.58	459.65
	Time Saving	1083.95	2523.85	3844.22	5168.16	6495.02	7820.99

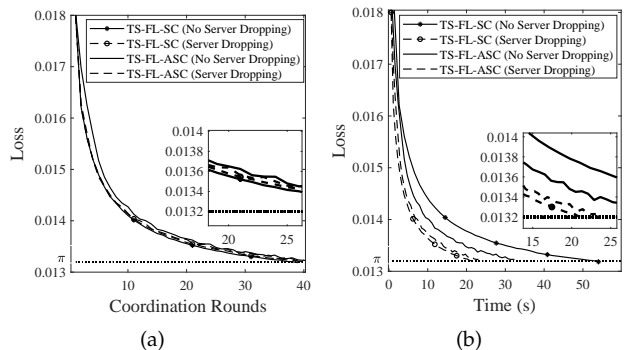


Fig. 9. (a) Number of coordination rounds and (b) model training time of TS-FL-SC and TS-FL-ASC under scenario (c).

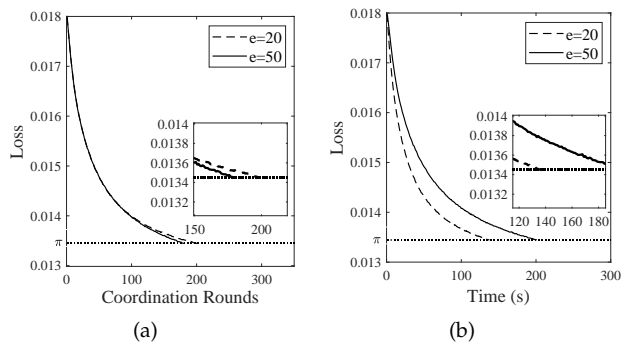


Fig. 10. (a) Number of coordination rounds and (b) model training time of TS-FL-SC with server dropping under different local epoch numbers e in scenario (a).

the loss functions under different numbers of coordination rounds and model training times exhibit different trends. In particular, in terms of convergence rate, the TS-FL-SC without server dropping has the fastest convergence, followed by TS-FL-ASC without server dropping. In other words, the server dropping cannot accelerate the convergence rate of an FEI system when measuring based on the number of coordination rounds. However, when we consider the time duration of the model training process, server dropping can significantly reduce the required time duration for training a satisfactory model (with target loss value). As observed in Fig. 6.3(b), the TS-FL-SC with server dropping can reduce the overall time required for model training almost by half, compared to TS-FL-SC without dropping any low-performance servers. We can also observe that TS-FL-ASC can also reduce the time duration of model training, especially when all edge servers are selected to participate in the model training. The performance improvement provided

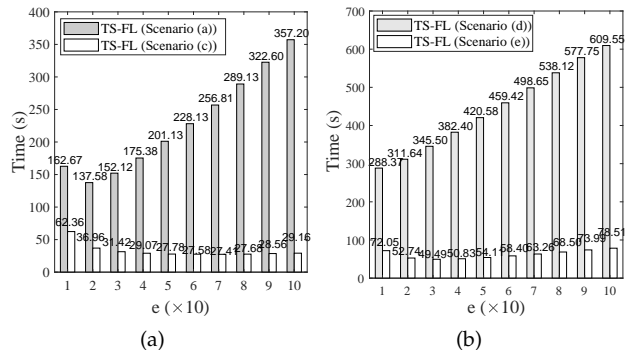


Fig. 11. Required time consumption for training a model with target loss ($\pi = 0.0132$) under different local epoch numbers e and a fixed mini-batch size $n = 500$ based on (a) handwritten digit dataset and (b) traffic dataset.

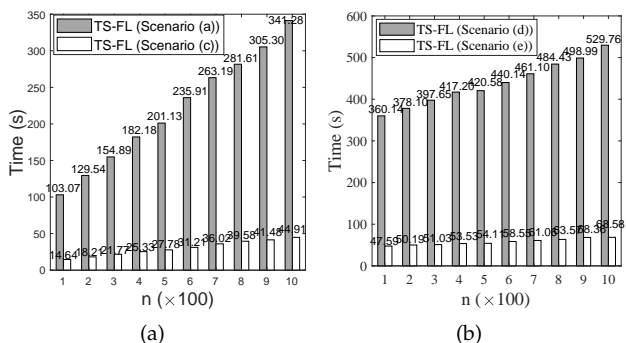


Fig. 12. Required time consumption for training a model with target loss ($\pi = 0.0132$) under different mini-batch sizes n and a fixed epoch number $e = 50$ based on (a) handwritten digit dataset and (b) traffic dataset.

by TS-FL-ASC, compared to TS-FL-SC, is reduced when server dropping is enabled. This is because the slowest edge server always dominates the overall time consumption of the model training and therefore when being removed from the model training process, the load forwarding between the slow and fast edge servers will have a limited effect in reducing the model training time.

In Fig. 8, we present the number of coordination rounds and time consumption required for training a satisfactory model with a target loss ($\pi = 0.0132$) under scenario (b). Recall that in scenario (b), some low-performance edge servers have data samples with unique labels. In this case, we can observe that removing the low-performance edge servers from participating in model training will slow the convergence rate, i.e., neither TS-FL-SC nor TS-FL-ASC can converge to the target loss within 500 rounds of coordination and if without server dropping, both solutions can obtain the target model after around 220 coordination rounds.

This observation becomes more noticeable if we consider the required time for training a satisfactory model. More specifically, in Fig. 8(b), we can observe that the TS-FL-ASC without server dropping is the fastest solution to train the model, followed by TS-FL-SC without server dropping. The later solution results in almost 40% more time consumption for training the same model.

In Fig. 9, we evaluate the required coordination rounds and time for reaching a target loss ($\pi = 0.0132$) under scenario (c). In this case, we can observe that in terms of required number of coordination rounds for training a model, the convergence performance of different solutions are quite similar, i.e., the difference in the required numbers of coordination rounds for reaching the same target loss when applying different solutions is within 2-3 coordination rounds. However, if we consider the overall time consumption for training the model, different solutions exhibits quite different convergence performance. More specifically, server dropping reduces the model training time in TS-FL-SC and TS-FL-ASC by around 63% and 28%, respectively. This once again verifies our previous observation that the model training time and convergence rate are different concepts and the former metric provides more important insight, especially in the smart applications and servers requiring time-sensitive learning and model updating.

As mentioned earlier, the model training time is closely related to two important model training parameters local epoch number e and mini-batch size n . In particular, in Sections 4 and 5, we have proved that if the overall time required for training any desirable model can be approximated by our derived convergence bound, the model training time can be considered as biconvex over n and e . In other words, if the approximated model training time is convex over n (or e) under any given e (or n). Let us now verify this results by comparing the overall time under different e and n . In particular, in Fig. 11 and 12, we compare the real model training time recorded in our hardware platform for reaching a target loss value ($\pi = 0.0132$) when either e or n is fixed while the other variables are changing into different values. In Fig. 11, we can observe that the real model training time verifies our theoretical results and the overall time required for training a satisfactory model exhibits convexity in our recorded traces. More specifically, there exists an optimal value of e which can minimize time consumption of model training under a given n . More specifically, the optimal epoch number e^* that can reach the target accuracy fastest in scenario (a) and scenario (c) in handwritten digit dataset are 20 and 60 with the recorded total time duration for model training at 137.58s and 27.41s, respectively. For the traffic dataset, the minimum model training time is achieved by selecting the smallest epoch number $e^* = 10$ in scenario (d) resulting 288.37s for training the model. In scenario (e) however, the minimum model training time is achieved at $e^* = 30$ resulting in 49.49s. In Fig. 12, we can observe that the required time for training the model increases almost linearly with the mini-batch sizes n . This is because a large mini-batch sizes not only means longer delay for data sample uploading from data collecting devices to the edge servers but also more time consumed for model training at each edge server.

To compare our proposed solution with the recent state-

of-the-art, in Table 6.3, we have presented the overall time consumption of model training process using our proposed solution and the scheduling-based approach proposed in [32] under two different datasets MNIST and traffic data with target model accuracy at 83.30% and 80.43% in scenarios (a) and (d), respectively. Since the scheduling-based approach in [32] is only applicable in the synchronous coordination scenario, to make the result comparable, we also present the model training time of server dropping in TS-FL-SC scenario. We can observe that the proposed server dropping solution can significantly reduce the overall runtime when training a model with the same guaranteed accuracy, especially when the local model training and updating speed difference between slow and the fast edge servers is large. In particular, our proposed server dropping solution can reduce up to 71.43% and 94.45% model training times in MNIST and traffic dataset, respectively, compared to the scheduling-based approach.

7 CONCLUSION

This paper investigated the real-time learning for FEI systems with system and dataset heterogeneity. A novel framework, called TS-FL, was proposed to minimize the overall run-time for collaboratively training a shared ML model with desirable accuracy. Training acceleration solutions for both TS-FL-SC and TS-FL-ASC were proposed. For the TS-FL-SC, a server dropping-based solution was proposed to allow some slow-performance edge servers to be removed from participating in the model training if their impact on the resulting model accuracy is limited. A joint optimization algorithm is proposed to minimize the overall time consumption of model training by selecting participating edge servers, the local epoch number (the number of model training iterations per coordination), and the data batch size (the number of data samples for each model training iteration). For the TS-FL-ASC, a load forwarding-based solution was proposed to allow the slow edge server to offload part of its training samples to trusted edge servers with higher processing capability. New theoretical convergence bound for TS-FL-SC and TS-FL-ASC are derived for model training based on non-i.i.d. datasets. We develop a hardware prototype to evaluate the model training time of a heterogeneous FEI system. Experimental results show that our proposed TS-FL-SC and TS-FL-ASC can provide up to 63% and 28% of reduction, in the overall model training time, respectively, compared with traditional FL solutions.

ACKNOWLEDGMENT

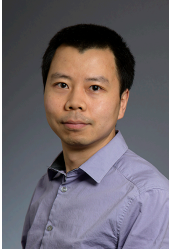
Y. Xiao was supported in part by the National Natural Science Foundation of China under grant 62071193 and the Key R & D Program of Hubei Province of China under grants 2021EHB015 and 2020BAA002. G. Shi was supported in part by the National Natural Science Foundation of China under grants 61871304, 61976169. Y. Xiao and G. Shi were supported in part by the major key project of Peng Cheng Laboratory (No. PCL2021A12). M. Krunk was supported by the National Science Foundation (grants 1910348, 1731164, 1813401, 2229386, and IIP-1822071) and by the Broadband Wireless Access & Applications Center (BWAC). Any opinions, findings, conclusions, or recommendations expressed

in this paper are those of the author(s) and do not necessarily reflect the views of NSF.

REFERENCES

- [1] Y. Xiao and M. Krunz, "AdaptiveFog: A modelling and optimization framework for fog computing in intelligent transportation systems," *IEEE Transactions on Mobile Computing*, vol. 21, no. 12, pp. 4187–4200, Dec. 2022.
- [2] Y. Xiao and M. Krunz, "Distributed optimization for energy-efficient fog computing in the Tactile Internet," *IEEE J. Sel. Area Commun.*, vol. 36, no. 11, pp. 2390–2400, Nov. 2018.
- [3] R. Nishihara, P. Moritz, S. Wang, A. Tumanov, W. Paul, J. Schleier-Smith, R. Liaw, M. Niknami, M. I. Jordan, and I. Stoica, "Real-time machine learning: The missing pieces," in *Proceedings of the 16th Workshop on Hot Topics in Operating Systems*, Whistler BC, Canada, Mar. 2017, pp. 106–110.
- [4] B. McMahan et al., "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, Ft. Lauderdale, FL, Apr. 2017, pp. 1273–1282.
- [5] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, pp. 1–19, Feb. 2019.
- [6] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, Apr. 2020.
- [7] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in iot," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5986–5994, Jul. 2019.
- [8] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: the confluence of edge computing and artificial intelligence," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7457–7469, Sep. 2020.
- [9] Y. Xiao, Y. Li, G. Shi, and H. V. Poor, "Optimizing resource-efficiency for federated edge intelligence in iot networks," in *International Conference on Wireless Communications and Signal Processing (WCSP)*, Wuhan, China, Nov. 2020, pp. 86–92.
- [10] Y. Xiao, G. Shi, Y. Li, W. Saad, and H. V. Poor, "Toward self-learning edge intelligence in 6G," *IEEE Communications Magazine*, vol. 58, no. 12, pp. 34–40, Dec. 2020.
- [11] S. U. Stich, "Local sgd converges fast and communicates little," *arXiv preprint arXiv:1805.09767*, May. 2018.
- [12] P. Han, S. Wang, and K. K. Leung, "Adaptive gradient sparsification for efficient federated learning: An online learning approach," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, Tamilnadu, India, Mar. 2020, pp. 300–310.
- [13] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *IEEE INFOCOM 2020*, Toronto, Canada, Jul. 2020, pp. 1698–1707.
- [14] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, Mar. 2019.
- [15] G. Shi, Y. Xiao, Y. Li, and X. Xie, "From semantic communication to semantic-aware networking: Model, architecture, and open problems," *IEEE Communications Magazine*, vol. 59, no. 8, pp. 44–50, Aug. 2021.
- [16] Y. Xiao, Z. Sun, G. Shi, and D. Niyato, "Imitation learning-based implicit semantic-aware communication networks: Multi-layer representation and collaborative reasoning," *To appear at IEEE J Sel. Areas in Commun.*, 2022.
- [17] Y. Xiao, Y. Li, G. Shi, and H. V. Poor, "Reasoning on the air: An implicit semantic communication architecture," in *Proc. of the IEEE ICC Workshop on Data Driven Intelligence for Networks and Systems*, Seoul, South Korea, May 2022.
- [18] Y. Xiao, X. Zhang, Y. Li, G. Shi, and T. Basar, "Rate-distortion theory for strategic semantic communication," in *Proc. of the IEEE Information Theory Workshop*, Mumbai, India, Nov. 2022.
- [19] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, May 2020.
- [20] M. Chen, B. Mao, and T. Ma, "Efficient and robust asynchronous federated learning with stragglers," in *International Conference on Learning Representations*, New Orleans, Sep. 2019.
- [21] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proceedings of Machine Learning and Systems*, vol. 2, Austin, TX, Mar. 2020, pp. 429–450.
- [22] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *IEEE International Conference on Communications (ICC)*, Shanghai, China, May 2019, pp. 1–7.
- [23] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Advances in neural information processing systems*, vol. 30, Long Beach, CA, 2017.
- [24] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data," *arXiv:1811.11479*, Nov. 2018.
- [25] L. Corinzia, A. Beuret, and J. M. Buhmann, "Variational federated multi-task learning," *arXiv:1906.06268*, Jun. 2019.
- [26] H. Eichner, T. Koren, B. McMahan, N. Srebro, and K. Talwar, "Semi-cyclic stochastic gradient descent," in *International Conference on Machine Learning*, Long Beach CA, Jun. 2019, pp. 1764–1773.
- [27] M. Khodak, M.-F. F. Balcan, and A. S. Talwalkar, "Adaptive gradient-based meta-learning methods," *Advances in Neural Information Processing Systems*, vol. 32, Vancouver, Canada, Dec. 2019.
- [28] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," *arXiv:1903.03934*, Mar. 2019.
- [29] J. Wang and G. Joshi, "Adaptive communication strategies to achieve the best error-runtime trade-off in local-update sgd," in *Proceedings of Machine Learning and Systems*, vol. 1, Stanford, CA, Mar. 2019, pp. 212–229.
- [30] Y. Li, F. Li, L. Chen, L. Zhu, P. Zhou, and Y. Wang, "Power of redundancy: Surplus client scheduling for federated learning against user uncertainties," *IEEE Transactions on Mobile Computing*, pp. 1–14, May 2022.
- [31] W. Xia, T. Q. S. Quek, K. Guo, W. Wen, H. H. Yang, and H. Zhu, "Multi-armed bandit-based client scheduling for federated learning," *IEEE Transactions on Wireless Communications*, vol. 19, no. 11, pp. 7108–7123, Jul. 2020.
- [32] B. Luo, X. Li, S. Wang, J. Huang, and L. Tassiulas, "Cost-effective federated learning design," in *IEEE INFOCOM 2021*, Vancouver BC, Canada, May 2021, pp. 1–10.
- [33] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 2022–2035, Mar. 2020.
- [34] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, Dec. 2012.
- [35] K. You, M. Long, J. Wang, and M. I. Jordan, "How does learning rate decay help modern neural networks?" *arXiv preprint arXiv:1908.01878*, Aug. 2019.
- [36] A. Lewkowycz, "How to decay your learning rate," *arXiv preprint arXiv:2103.12682*, Mar. 2021.
- [37] H. Yu, S. Yang, and S. Zhu, "Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning," in *AAAI 2019*, vol. 33, no. 01, Honolulu, Hawaii, Jul. 2019, pp. 5693–5700.
- [38] M. P. Perrone, H. Khan, C. Kim, A. Kyrillidis, J. Quinn, and V. Salapura, "Optimal mini-batch size selection for fast gradient descent," *arXiv:1911.06459*, Nov. 2019.
- [39] Q. Wang, Y. Xiao, H. Zhu, Z. Sun, Y. Li, and X. Ge, "Towards energy-efficient federated edge intelligence for iot networks," in *IEEE ICDCS Workshop*, Washington DC, Jul. 2021, pp. 55–62.
- [40] A. Khaled, K. Mishchenko, and P. Richtárik, "Tighter theory for local sgd on identical and heterogeneous data," in *International Conference on Artificial Intelligence and Statistics*, Virtual, Aug. 2020, pp. 4519–4529.
- [41] M. BOURREAU, "Cooperation between telecommunications operators for infrastructure deployment," CERRE Issue Paper, Oct. 2021. [Online]. Available: https://cerre.eu/wp-content/uploads/2021/10/211025_CERRE_IP_Co-Investment-Final.pdf
- [42] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Feb. 1998.
- [43] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related," in *Proceedings of the 2nd international conference on information systems security and privacy (ICISSP)*, Portugal, Feb. 2016, pp. 407–414.

- [44] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, Jul. 2019.



Yong Xiao (Senior Member, IEEE) received his B.S. degree in electrical engineering from China University of Geosciences, Wuhan, China in 2002, M.Sc. degree in telecommunication from Hong Kong University of Science and Technology in 2006, and his Ph. D degree in electrical and electronic engineering from Nanyang Technological University, Singapore in 2012. He is now a professor in the School of Electronic Information and Communications at the Huazhong University of Science and Technology (HUST),

Wuhan, China. He is also with Peng Cheng Laboratory, Shenzhen, China and Pazhou Laboratory (Huangpu), Guangzhou, China. He is the associate group leader of the network intelligence group of IMT-2030 (6G promoting group) and the vice director of 5G Verticals Innovation Laboratory at HUST. Before he joins HUST, he was a research assistant professor in the Department of Electrical and Computer Engineering at the University of Arizona where he was also the center manager of the Broadband Wireless Access and Applications Center (BWAC), an NSF Industry/University Cooperative Research Center (I/UCRC) led by the University of Arizona. His research interests include machine learning, game theory, distributed optimization, and their applications in cloud/fog/mobile edge computing, green communication systems, wireless communication networks, and Internet-of-Things (IoT).



Xiaohan Zhang received his B.S. degree in electronic information engineering from China University of Geosciences, Wuhan, China in 2017, M.Sc. degree in information and communication engineering from Huazhong University of Science and Technology, Wuhan, China in 2022. His research interests include federated learning and cloud/edge computing networks.



Yingyu Li (Member, IEEE) received the B.Eng. degree in electronic information engineering and the Ph.D. degree in circuits and systems from the Xidian University, Xi'an, China, in 2012 and 2018, respectively. From 2014 to 2016, she was a Research Scholar with the Department of Electronic Computer Engineering at the University of Houston, TX, USA. She was a post-doctoral researcher in the School of Electronic Information and Communications at Huazhong University of Science and Technology from 2018

to 2021. She is now an associate professor at the School of Mechanical Engineering and Electronic Information, China University of Geosciences (Wuhan). Her research interests include semantic communications, edge intelligence, green communication networks, and IoT.



Guangming Shi (Fellow, IEEE) received the M.S. degree in computer control and the Ph.D. degree in electronic information technology from Xidian University, Xi'an, China, in 1988, and 2002, respectively. He was the vice president of Xidian University from 2018 to 2022. Currently, he is the Vice Dean of Peng Cheng Laboratory and a Professor with the School of Artificial Intelligence, Xidian University. He is an IEEE Fellow, the chair of IEEE CASS Xi'an Chapter, senior member of ACM and CCF, Fellow of Chinese

Institute of Electronics, and Fellow of IET. He was awarded Cheung Kong scholar Chair Professor by the ministry of education in 2012. He won the second prize of the National Natural Science Award in 2017. His research interests include Artificial Intelligence, Semantic Communications, and Human-Computer Interaction.



Marwan Krunz (Fellow, IEEE) is a Regents Professor at the University of Arizona. He holds the Kenneth VonBehren Endowed Professorship in ECE and is also a professor of computer science. He directs the Broadband Wireless Access and Applications Center (BWAC), a multi-university NSF/industry center that focuses on next-generation wireless technologies. He also holds a courtesy appointment as a professor at University Technology Sydney. Previously, he served as the site director for Connection One,

an NSF/industry-funded center of five universities and 20+ industry affiliates. Dr. Krunz's research is in the fields of wireless communications, networking, and security, with recent focus on applying AI and machine learning techniques for protocol adaptation, resource management, and signal intelligence. He has published more than 320 journal articles and peer-reviewed conference papers, and is a named inventor on 12 patents. His latest h-index is 60. He is an IEEE Fellow, an Arizona Engineering Faculty Fellow, and an IEEE Communications Society Distinguished Lecturer (2013-2015). He received the NSF CAREER award. He served as the Editor-in-Chief for the IEEE Transactions on Mobile Computing. He also served as editor for numerous IEEE journals. He was the TPC chair for INFOCOM'04, SECON'05, WoWMoM'06, and Hot Interconnects 9. He was the general vice-chair for WiOpt 2016 and general co-chair for WiSec'12. Dr. Krunz served as chief scientist/technologist for two startup companies that focus on 5G and beyond wireless systems.



Diep N. Nguyen (Senior Member, IEEE) received the M.E. degree in electrical and computer engineering from the University of California at San Diego (UCSD), La Jolla, CA, USA, in 2008, and the Ph.D. degree in electrical and computer engineering from The University of Arizona (UA), Tucson, AZ, USA, in 2013. He is currently a Faculty Member with the Faculty of Engineering and Information Technology, University of Technology Sydney (UTS), Sydney, NSW, Australia. Before joining UTS, he was a

DECRA Research Fellow with Macquarie University, Macquarie Park, NSW, Australia, and a Member of the Technical Staff with Broadcom Corporation, San Jose, CA, USA, and ARCON Corporation, Boston, MA, USA, and consulting the Federal Administration of Aviation, Washington, DC, USA, on turning detection of UAVs and aircraft, and the U.S. Air Force Research Laboratory, Wright-Patterson Air Force Base, OH, USA, on anti-jamming. His research interests include computer networking, wireless communications, and machine learning application, with emphasis on systems' performance and security/privacy. Dr. Nguyen received several awards from LG Electronics, UCSD, UA, the U.S. National Science Foundation, and the Australian Research Council. He is currently an Editor, an Associate Editor of the IEEE Transactions on Mobile Computing, IEEE Communications Surveys & Tutorials (COMST), IEEE Open Journal of the Communications Society, and Scientific Reports (Nature's).



Dinh Thai Hoang is currently a faculty member at the School of Electrical and Data Engineering, University of Technology Sydney, Australia. He received his Ph.D. in Computer Science and Engineering from the Nanyang Technological University, Singapore, in 2016. His research interests include emerging wireless communications and networking topics, especially machine learning applications in networking, edge computing, and cybersecurity. He has received several awards, including the Australian Research

Council and IEEE TCSC Award for Excellence in Scalable Computing (Early Career Researcher). He is an Editor of IEEE Transactions on Wireless Communications, IEEE Transactions on Cognitive Communications and Networking, IEEE Transactions on Vehicular Technology, and Associate Editor of IEEE Communications Surveys & Tutorials.

APPENDIX A PROOF OF THEOREM 1

Let us now derive the convergence rate of TS-FL-SC with server dropping. To facilitate our analysis, we introduce a virtual sequence $\bar{\mathbf{w}}_{\mathcal{M},t} = \sum_{k \in \mathcal{M}} p_k \mathbf{w}_{k,t}$ to represent the (virtual) sequence of equivalent global model if all the local models at a subset \mathcal{M} of edge servers can be aggregated at every local SGD iteration. Note that, after T SGD iterations, the resulting global model aggregated by the coordinator should be equivalent to the most updated model in the virtual sequence, i.e., $\mathbf{w}_{\mathcal{M},T} = \bar{\mathbf{w}}_{\mathcal{M},T}$. We can therefore focus on deriving the convergence rate of the virtual sequence during the rest of this proof.

Note that our proposed TS-FL-SC with server dropping is different from the traditional FL with partial server participation. In the latter solution, every edge server will have equal chance of being selected to participate in the model training process and therefore no edge servers will be removed from the entire model training process.

One of the key issue of TS-FL-SC with server dropping is to estimate the gap between the model $\bar{\mathbf{w}}_{\mathcal{M},t}$ trained with a subset \mathcal{M} of edge servers and the global optimal model $\mathbf{w}_{\mathcal{K}}^*$ trained with the set \mathcal{K} of all K edge servers, i.e., $\|\bar{\mathbf{w}}_{\mathcal{M},t} - \mathbf{w}_{\mathcal{K}}^*\|^2$.

In [44], it has proved the following upper bound for model training with a set \mathcal{M} edge server with full participation.

Lemma 4. [44, Theorem 1] Suppose the following assumptions hold: (1) F_1, F_2, \dots, F_K are all L -smooth and μ -convex, i.e., $\frac{\mu}{2} \|\mathbf{w} - \mathbf{w}'\|^2 \leq F_k(\mathbf{w}) - F_k(\mathbf{w}') + (\mathbf{w} - \mathbf{w}')^T \nabla F_k(\mathbf{w}') \leq \frac{L}{2} \|\mathbf{w} - \mathbf{w}'\|^2$ for all $\mathbf{w}, \mathbf{w}' \in \mathbb{R}^d$; (2) data samples x_k are uniformly randomly sampled from \mathcal{D}_k ; and (3) the stochastic gradient satisfies $\mathbb{E} \|\nabla l_k(\mathbf{w}, x_k)\|^2 \leq G^2$ and $\mathbb{E} \|\nabla l_k(\mathbf{w}, x_k) - \nabla F_k(\mathbf{w})\|^2 \leq \sigma_k^2$. If $\kappa = \frac{L}{\mu}$, $\gamma = \max\{8\kappa, e\}$, and the learning rate η_t satisfies $\eta_t = \frac{2}{\mu(\gamma+t)}$, we have

$$\mathbb{E}(\|\bar{\mathbf{w}}_{\mathcal{M},t} - \mathbf{w}_{\mathcal{M}}^*\|^2) \leq \Delta \quad (43)$$

where Δ is given by

$$\begin{aligned} \Delta &= \frac{4}{\mu^2(\gamma+t)} \left(\sum_{k \in \mathcal{M}} \frac{p_k^2 \sigma_k^2}{n_k} + 8e^2 G^2 + 6LD_{\mathcal{M}}^*(F) \right. \\ &\quad \left. + \frac{\mu^2(\gamma+1)}{4} \|\mathbf{w}_0 - \mathbf{w}_{\mathcal{M}}^*\|^2 \right). \end{aligned} \quad (44)$$

Decomposing the term $\|\bar{\mathbf{w}}_{\mathcal{M},t} - \mathbf{w}_{\mathcal{M}}^*\|^2$, and we can write

$$\begin{aligned} &\mathbb{E} \|\bar{\mathbf{w}}_{\mathcal{M},t} - \mathbf{w}_{\mathcal{M}}^*\|^2 - \mathbb{E}(\|\bar{\mathbf{w}}_{\mathcal{M},t} - \mathbf{w}_{\mathcal{K}}^*\|^2) \\ &= 2\langle \bar{\mathbf{w}}_{\mathcal{M},t} - \mathbf{w}_{\mathcal{M}}^* + \mathbf{w}_{\mathcal{M}}^* - \mathbf{w}_{\mathcal{K}}^*, \mathbf{w}_{\mathcal{K}}^* - \mathbf{w}_{\mathcal{M}}^* \rangle \\ &\quad + \|\mathbf{w}_{\mathcal{K}}^* - \mathbf{w}_{\mathcal{M}}^*\|^2 \\ &= 2\langle \bar{\mathbf{w}}_{\mathcal{M},t} - \mathbf{w}_{\mathcal{M}}^*, \mathbf{w}_{\mathcal{K}}^* - \mathbf{w}_{\mathcal{M}}^* \rangle - \|\mathbf{w}_{\mathcal{K}}^* - \mathbf{w}_{\mathcal{M}}^*\|^2 \quad (45) \\ &\geq -\mathbb{E} \|\bar{\mathbf{w}}_{\mathcal{M},t} - \mathbf{w}_{\mathcal{M}}^*\|^2 - 2\|\mathbf{w}_{\mathcal{K}}^* - \mathbf{w}_{\mathcal{M}}^*\|^2 \quad (46) \end{aligned}$$

where (45) comes from the fact that $2ab \geq -a^2 - b^2$.

Combining Lemma 4 and equation (46), we have

$$\mathbb{E}(\|\bar{\mathbf{w}}_{\mathcal{M},t} - \mathbf{w}_{\mathcal{K}}^*\|^2) \leq 2\Delta + 2\|\mathbf{w}_{\mathcal{K}}^* - \mathbf{w}_{\mathcal{M}}^*\|^2. \quad (47)$$

Combining (47) and the L -smooth property of F , we can obtain

$$\begin{aligned} \mathbb{E}(F(\bar{\mathbf{w}}_{\mathcal{M},t})) - F^* &\leq \frac{L}{2} \mathbb{E}(\|\bar{\mathbf{w}}_{\mathcal{M},t} - \mathbf{w}_{\mathcal{K}}^*\|^2) \quad (48) \\ &\leq L\Delta + L\|\mathbf{w}_{\mathcal{K}}^* - \mathbf{w}_{\mathcal{M}}^*\|^2. \quad (49) \end{aligned}$$

This concludes the proof.

APPENDIX B

PROOF OF LEMMA 1

Suppose $n_k = n$ for each $k \in \mathcal{K}$ and changing n does not affect the ranking of model training time among edge servers, we rewrite the objective function (16) as follows:

$$\mathbb{E}[\tilde{c}(e, n, \mathcal{M})] = \frac{\zeta + \alpha_{\bar{M}}en + \beta_{\bar{M}}e + u}{e(\epsilon - D_{\mathcal{M}})} \left(\frac{A}{n} + Be^2 + C_{\mathcal{M}} \right).$$

Let us first prove the above objective function is convex over e for the given n and \mathcal{M} .

We rewrite the objective function into the following simplified form:

$$\mathbb{E}[\tilde{c}(e, n, \mathcal{M})] = (A' + B'e) \left(\frac{C'}{e} + D'e \right), \quad (50)$$

where $A' = \frac{\zeta + u}{(\epsilon - D_{\mathcal{M}})}$, $B' = \frac{n\alpha_{\bar{M}} + \beta_{\bar{M}}}{(\epsilon - D_{\mathcal{M}})}$, $C' \triangleq \frac{A}{n} + C_{\mathcal{M}}$, and $D' = B$ can be considered as positive constants.

To prove the convexity of (50), we take the second order derivative of $\mathbb{E}[\tilde{c}(e, n, \mathcal{M})]$ over e as follows:

$$\frac{\partial^2 \mathbb{E}[\tilde{c}(e, n, \mathcal{M})]}{\partial e^2} = 2B'D' + 2\frac{A'C'}{e^3} > 0, \quad (51)$$

We can therefore claim (50) is a convex function over e .

Let us now prove the objective function in (16) is convex over n under the given e and \mathcal{M} . Similarly, we can rewrite the objective function into the following form:

$$\mathbb{E}[\tilde{c}(e, n, \mathcal{M})] = (A'' + B''n) \left(\frac{C''}{n} + D'' \right) \quad (52)$$

where $A'' = \frac{\zeta + \beta_{\bar{M}}e + u}{e(\epsilon - D_{\mathcal{M}})}$, $B'' = \frac{\alpha_{\bar{M}}}{(\epsilon - D_{\mathcal{M}})}$, $C'' = A$ and $D'' = Be^2 + C_{\mathcal{M}}$ can be considered as positive constants. By taking the second derivative over (52), we can have

$$\frac{\partial^2 \mathbb{E}[\tilde{c}(e, n, \mathcal{M})]}{\partial n^2} = 2\frac{A''C''}{n^3} > 0, \quad (53)$$

which indicates the function in (52) is also convex of n .

According to the definition of biconvexity, we can claim that the objective function (16) is biconvex over e and n . This concludes the proof.

APPENDIX C

PROOF OF LEMMA 2

Let us first define $t_k \leq t$ as the most recent iteration of each edge server k that is updated by the global model coordination, i.e., $t_k \in \mathcal{I}^k$ and $t - t_k \leq e_k$.

We can then decompose term $\|\bar{\mathbf{w}}_t - \mathbf{w}_{k,t}\|^2$ into the following form:

$$\begin{aligned} \|\bar{\mathbf{w}}_t - \mathbf{w}_{k,t}\|^2 &\leq 3(\|\mathbf{w}_{k,t} - \mathbf{w}_{k,t_k}\|^2 + \|\mathbf{w}_{k,t_k} - \bar{\mathbf{w}}_{t_k}\|^2 \\ &\quad + \|\bar{\mathbf{w}}_{t_k} - \bar{\mathbf{w}}_t\|^2), \quad (54) \end{aligned}$$

where $\bar{\mathbf{w}}_t \triangleq \mathbf{w}_0 - \sum_{k=1}^K p_k \sum_{j=0}^{t-1} \eta_j \mathbf{g}_{k,j}$.

We first analyze the first term $\|\mathbf{w}_{k,t} - \mathbf{w}_{k,t_k}\|^2$ on the left-hand-side of equation (54) which characterizes the local model training since last global model coordination at iteration t_k . We can derive the following bound for $\|\mathbf{w}_{k,t} - \mathbf{w}_{k,t_k}\|^2$ as follows:

$$\|\mathbf{w}_{k,t} - \mathbf{w}_{k,t_k}\|^2 = \left\| \sum_{j=t_k}^{t-1} \eta_j \nabla \mathbf{g}_{k,j} \right\|^2 \quad (55)$$

$$\leq \eta_{t_k}^2 (t - t_k)^2 G^2 \quad (56)$$

$$\leq \eta_{t_k}^2 e_k^2 G^2 \quad (57)$$

$$\leq 4\eta_t^2 e_k^2 G^2 \quad (58)$$

where (55) is obtained by the model updating rule in (3), (56) is obtained by Assumption 1, (57) comes from the fact that $t - t_k \leq e_k$ and η_t is non-increasing, and (58) is based on the property of learning rate $\eta_t \leq 2\eta_{H+\tau+t}$ for all $t > 0$.

For the second term on the left-hand-side of equation (54), we can derive the following bound:

$$\|\bar{\mathbf{w}}_t - \bar{\mathbf{w}}_{t_k}\|^2 = \left\| \sum_{h=1}^K p_h \sum_{j=t_k}^{t-1} \eta_j \mathbf{g}_{h,j} \right\|^2 \quad (59)$$

$$\leq \eta_{t_k}^2 e_k^2 G^2 \quad (60)$$

$$\leq 4\eta_t^2 e_k^2 G^2 \quad (61)$$

where (59) is the definition of virtual sequence, (60) and (61) follow the same line as (57) and (58).

The third term on the left-hand-side of equation (54) captures the divergence between the virtual sequence and global model at the coordinator. We can write

$$\|\mathbf{w}_{k,t_k} - \bar{\mathbf{w}}_{t_k}\|^2 = \|\mathbf{v}_{k,t_k} - \bar{\mathbf{w}}_{t_k}\|^2 \quad (62)$$

$$= \left\| \sum_{h=1}^K p_h \sum_{j \in \mathcal{W}_{k,t_k}^h} \eta_j \mathbf{g}_{h,j} - \sum_{h=1}^K p_h \sum_{j=0}^{t_k-1} \eta_j \mathbf{g}_{h,j} \right\|^2 \quad (63)$$

$$\leq 16\eta_t^2 \tau^2 G^2. \quad (64)$$

where (62) is based on the model updating rule in (3), (63) follows the definition of \mathbf{v}_{k,t_k} and $\bar{\mathbf{w}}_{t_k}$, (64) is according to constraint in (27) and the property of learning rate used in (58).

Combining (58), (61), and (64), we have

$$\begin{aligned} \|\bar{\mathbf{w}}_t - \mathbf{w}_{k,t}\|^2 &\leq 3(\|\mathbf{w}_{k,t} - \mathbf{w}_{k,t_k}\|^2 + \|\mathbf{w}_{k,t_k} - \bar{\mathbf{w}}_{t_k}\|^2 \\ &\quad + \|\bar{\mathbf{w}}_{t_k} - \bar{\mathbf{w}}_t\|^2) \leq 8\eta_t^2 G^2 (2\tau^2 + e_k^2). \end{aligned}$$

We can also obtain the following results:

$$\mathbb{E} \left[\sum_{k=1}^K p_k \|\bar{\mathbf{w}}_t - \mathbf{w}_{k,t}\|^2 \right] \leq 24\eta_t^2 G^2 \left(2\tau^2 + \sum_{k=1}^K p_k e_k^2 \right).$$

This concludes the proof.

APPENDIX D

PROOF OF THEOREM 2

To derive the convergence upper bound of TS-FL-ASC, we derive the convergence of $\mathbb{E}[\|\bar{\mathbf{w}}_{t+1} - \mathbf{w}_{\mathcal{K}}^*\|^2]$.

Let us first introduce the following result which has already been proved in [44].

Lemma 5. [44, Lemma 1, 2] Suppose the following assumptions hold: (1) F_1, F_2, \dots, F_K are all L -smooth and

μ -convex, i.e., $\frac{\mu}{2}\|\mathbf{w} - \mathbf{w}'\|^2 \leq F_k(\mathbf{w}) - F_k(\mathbf{w}') + (\mathbf{w} - \mathbf{w}')^T \nabla F_k(\mathbf{w}') \leq \frac{L}{2}\|\mathbf{w} - \mathbf{w}'\|^2$ for all $\mathbf{w}, \mathbf{w}' \in \mathbb{R}^d$; (2) data sample x_k is uniformly randomly sampled from \mathcal{D}_k ; and (3) the stochastic gradient satisfies $\mathbb{E}\|\nabla l_k(\mathbf{w}, x_k)\|^2 \leq G^2$ and $\mathbb{E}\|\nabla l_k(\mathbf{w}, x_k) - \nabla F_k(\mathbf{w})\|^2 \leq \sigma_k^2$. By setting $\kappa = \frac{L}{\mu}$, $\gamma = \max\{8\kappa, \tau + H\}$ and $\eta_t = \frac{2}{\mu(\gamma+t)}$, we have

$$\begin{aligned} \mathbb{E}[\|\bar{\mathbf{w}}_{t+1} - \mathbf{w}_{\mathcal{K}}^*\|^2] &\leq (1 - \eta_t \mu) \mathbb{E}[\|\bar{\mathbf{w}}_t - \mathbf{w}_{\mathcal{K}}^*\|^2] \\ &\quad + \eta_t^2 \sum_{k=1}^K \frac{p_k^2 \sigma_k^2}{n_k} + 6\eta_t^2 LD_{\mathcal{K}}^*(F) \\ &\quad + 2\mathbb{E}\left[\sum_{k=1}^K p_k \|\bar{\mathbf{w}}_t - \mathbf{w}_{k,t}\|^2\right], \end{aligned}$$

In Lemma 2, we have derived a bound for the term $\mathbb{E}\left[\sum_{k=1}^K p_k \|\bar{\mathbf{w}}_t - \mathbf{w}_{k,t}\|^2\right]$. Substituting (29) into Lemma 5, we have

$$\mathbb{E}\|\bar{\mathbf{w}}_{t+1} - \mathbf{w}_{\mathcal{K}}^*\|^2 \leq (1 - \eta_t \mu) \mathbb{E}\|\bar{\mathbf{w}}_t - \mathbf{w}_{\mathcal{K}}^*\|^2 + \eta_t^2 B, \quad (65)$$

where

$$B = \sum_{k=1}^K \frac{p_k^2 \sigma_k^2}{n_k} + 6LD_{\mathcal{K}}^*(F) + 48G^2(2\tau^2 + \sum_{k=1}^K p_k e_k^2). \quad (66)$$

Let $\Delta_t = \mathbb{E}[\|\bar{\mathbf{w}}_t - \mathbf{w}_{\mathcal{K}}^*\|^2]$ and $\xi = \max\{\frac{4B}{\mu^2}, \gamma\|\mathbf{w}_0 - \mathbf{w}_{\mathcal{K}}^*\|^2\}$. Based on the definition of ξ , if $t = 0$, we have $\Delta_t \leq \frac{\xi}{\gamma+t}$. We can then write Δ_{t+1} as the following form:

$$\Delta_{t+1} \leq (1 - \eta_t \mu) \Delta_t + \eta_t^2 B \quad (67)$$

$$\leq \left(1 - \frac{2}{\gamma+t}\right) \frac{\xi}{\gamma+t} + \frac{4B}{\mu^2(t+\gamma)^2} \quad (68)$$

$$\begin{aligned} &= \frac{(\gamma+t-1)\xi}{(\gamma+t)^2} + \frac{1}{(\gamma+t)^2} \left(\frac{4B}{\mu^2} - \xi\right) \\ &\leq \frac{(\gamma+t-1)\xi}{(\gamma+t)^2} \quad (69) \end{aligned}$$

$$< \frac{(\gamma+t-1)\xi}{(\gamma+t)^2 - 1} = \frac{\xi}{\gamma+t+1}. \quad (70)$$

where (67) comes directly from (65), (68) is based on the property of learning rate defined in Theorem 2, (69) comes from the fact that $\frac{4B}{\mu^2} \leq \xi$.

According to the definition of ξ , we have the inequality $\xi = \max\{\frac{4B}{\mu^2}, \gamma\Delta_0\} \leq \frac{4B}{\mu^2} + \gamma\Delta_0$.

Substituting the above inequality into (70), we have

$$\mathbb{E}\|\bar{\mathbf{w}}_t - \mathbf{w}_{\mathcal{K}}^*\|^2 \leq \frac{1}{\gamma+t} \left(\frac{4B}{\mu^2} + \gamma\Delta_0\right). \quad (71)$$

Using the property of L -smooth of $F(\cdot)$, we can obtain

$$\begin{aligned} \mathbb{E}[F(\bar{\mathbf{w}}_t) - F^*] &\leq \frac{L}{2} \mathbb{E}\|\bar{\mathbf{w}}_t - \mathbf{w}_{\mathcal{K}}^*\|^2 \\ &\leq \frac{L}{2} \frac{1}{\gamma+t} \left(\frac{4B}{\mu^2} + \gamma\Delta_0\right) \\ &\leq \frac{\kappa}{\gamma+t} \left(\frac{2B}{\mu} + \frac{\mu\gamma}{2} \Delta_0\right) \quad (72) \end{aligned}$$

where $\kappa = \frac{L}{\mu}$. This concludes the proof.

APPENDIX E PROOF OF THEOREM 3

According to Theorem 2, the model trained by a set \mathcal{M} of edge servers can always converge to the optimal weight $\mathbf{w}_{\mathcal{M}}^*$, and we can write

$$\mathbb{E}(\|\bar{\mathbf{w}}_{\mathcal{M},t} - \mathbf{w}_{\mathcal{M}}^*\|^2) \leq \Delta, \quad (73)$$

where

$$\begin{aligned} \Delta &= \frac{4}{\mu^2(\gamma+t)} \left(\sum_{k \in \mathcal{M}} \frac{p_k^2 \sigma_k^2}{n_k} + 48G^2(2\tau^2 + \sum_{k=1}^K p_k e_k^2) \right. \\ &\quad \left. + 6LD_{\mathcal{M}}^*(F) + \frac{\mu^2(\gamma+1)}{4} \|\mathbf{w}_0 - \mathbf{w}_{\mathcal{M}}^*\|^2 \right) \quad (74) \end{aligned}$$

Combining (74) with (49), we have

$$\mathbb{E}(F(\bar{\mathbf{w}}_{\mathcal{M},t})) - F^* \leq L\Delta + L\|\mathbf{w}^* - \mathbf{w}_{\mathcal{M}}^*\|^2. \quad (75)$$

This concludes the proof.