Federated Learning With Dynamic Epoch Adjustment and Collaborative Training in Mobile Edge Computing

Tianao Xiang, Yuanguo Bi, *Member, IEEE*, Xiangyi Chen, Yuan Liu, Boyang Wang, Xuemin Shen, *Fellow, IEEE*, and Xingwei Wang

***Abstract*—As a distributed learning paradigm, federated learning (FL) can be applied in mobile edge computing (MEC) to support real-time artificial intelligence by leveraging edge computation resources while preserving data privacy in the end devices. However, the unpredictable wireless connections between end devices and edge servers in MEC (e.g., frequent handovers and unstable wireless channels) may result in the loss of important model parameters, which slows down the FL training process and degrades the quality of the global model. In this paper, we propose an adaptive collaborative federated learning (ACFL) scheme to accelerate the convergence and improve model reliability by mitigating communication-based parameter loss under a three-layer MEC architecture. First, a dynamic epoch adjustment method is proposed to reduce communication rounds by dynamically adjusting the training epochs in end devices. In addition, to accelerate the FL convergence, we present an edge server collaborative training scheme by leveraging a multi-layer computing architecture, where edge servers utilize their maintained data to collaboratively train models with end devices. Finally, extensive simulations are conducted and show that ACFL can efficiently improve model reliability and accelerate the convergence of the FL process in MEC.**

***Index Terms*—Epoch adjustment, mobile edge computing, model reliability, federated learning.**

Manuscript received 18 November 2022; revised 10 May 2023; accepted 8 June 2023. Date of publication 21 June 2023; date of current version 4 April 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFB3102700, in part by the National Natural Science Foundation of China under Grants U1808207 and 62171113, and in part by the Fundamental Research Funds for the Central Universities of China under Grants N2124006-1 and N2116014. Recommended for acceptance by G. Dan. *(Corresponding author: Yuanguo Bi.)*

Tianao Xiang, Yuanguo Bi, Xiangyi Chen, Boyang Wang, and Xingwei Wang are with the School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China, and also with the Engineering Research Center of Security Technology of Complex Network System, Ministry of Education, Shenyang 110169, China (e-mail: 2010652@stu.neu.edu.cn; biyuanguo@mail.neu.edu.cn; xiangyichen3746@gmail.com; 13945918917@163.com; wangxw@mail.neu.edu.cn).

Yuan Liu is with the Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou 511442, China (e-mail: liuyuan@swc.neu.edu.cn).

Xuemin Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: sshen@uwaterloo.ca).

This article has supplementary downloadable material available at https://doi.org/10.1109/TMC.2023.3288392, provided by the authors.

Digital Object Identifier 10.1109/TMC.2023.3288392

I. Introduction

With the developments of smart devices and communication technologies, more than 125 billion devices will be deployed in various areas by 2030, such as industrial Internet, intelligent vehicle, health care, etc., according to IHS Markit forecasts [1]. Compared with traditional multimedia services, the applications in these areas usually have a tight requirement on response latency, which forces us to deploy such services at the edge of the network [2], [3]. Mobile edge computing (MEC) [4] has been emerging as a promising technology to meet the above quality of service (QoS) requirements, and it can provide real-time services with local computation capability, low latency, and large bandwidth communications. Meanwhile, the MEC architecture can enable the physical proximity between the computing servers and the information-generation sources, which facilitates the developments of data-driven artificial intelligence (AI) and significantly improves training efficiency. As a result, combining edge computing and AI has given rise to a new research area, namely, edge intelligence [5]. Rather than relying on the cloud solely, edge intelligence leverages extensive edge resources to gain AI insights, which boosts many AI applications, e.g., cognitive assistance, smart home, Internet of Things, and video analytic.

As a distributed model-level learning paradigm, federated learning (FL) has excellent potential to provide edge intelligence and privacy protection in MEC [6][7]. End devices participating in the FL process collaboratively build a high-performance global model while training data locally. An FL process consists of several training rounds, and only a fraction of end devices are selected to participate in the training process in each round. Each participating end device trains its model with local data, and then the cloud server aggregates the received models to obtain a global model. The principal advantage of FL is decoupling model training from direct access to the raw training data [8]. Without data exchange, an FL process preserves data privacy and reduces the risk of data leakage. Based on this feature, FL is adopted in a variety of areas from commercial companies to academic organizations such as Google [9], Webank [10], and IEEE [11], etc. Meanwhile, FL integrated with MEC has been investigated in numerous research fields, such as blockchain [12][13], deep reinforcement learning [6], and virtual network function auto-scaling [14].

However, due to limited bandwidth, unreliable wireless channels, frequent handover, etc., integrating FL with MEC is confronted with several challenges. On the one hand, the frequent data transmissions and handovers of mobile end devices increase the possibility of packet loss in MEC. Hence, the unstable wireless connections may lead to the loss of important model parameters and even make the model unavailable, degrading the global model's quality. On the other hand, with various user characteristics, the data generated by end devices may be significantly different. Owing to the privacy-preserving feature of FL, devices cannot exchange data to alleviate the difference, which induces the non-independent and identically distributed (non-i.i.d.) issue. The non-i.i.d. issue makes the global model hard to converge and slows down the FL process.

To improve model reliability and training efficiency, we propose a novel hierarchical edge server-assisted FL scheme that is Adaptive Collaborative Learning (ACFL) in MEC. Specifically, we propose a dynamic epoch adjustment method by establishing a formal correlation between communication rounds and the probability of model transmission failure, and present an optimal solution with theoretical convergence analysis based on the minimum communication rounds and the corresponding local training epochs. We then present an approximate solution by scaling critical parameters to calculate the optimal communication rounds and the corresponding epochs, and design a dynamic epoch adjustment method that reduces communication rounds with an approximate solution. Moreover, an auxiliary dataset is constructed by using non-sensitive data generated and maintained in edge servers. Then, we propose an edge server-assisted collaborative training scheme based on the auxiliary dataset, which mitigates the non-i.i.d. issue and further reduces communication rounds to enhance model reliability. The contributions of this paper are four folds as follows.

1) We theoretically derive the optimal communication rounds and the corresponding epochs, and propose a dynamic epoch adjustment method to decrease the probability of losing crucial model parameters due to fragile wireless connections.
2) We present a condition for verifying the validity of auxiliary datasets by analyzing the impact of data heterogeneity, and then propose an edge server collaborative training (ES co-training) scheme, which allows an edge server to collaborate with end devices to train models by using an auxiliary dataset.
3) We develop an adaptive federated learning algorithm with the auxiliary dataset in an edge server by applying the ES co-training scheme and dynamic epochs adjustment method, which minimizes communication rounds and simultaneously mitigates the non-i.i.d. issue.
4) Extensive simulations demonstrate that over 80% of the estimated epochs are close to the optimal epochs in practice, and the proposed ACFL can effectively reduce communication rounds to reach the required accuracy.

The rest of the paper is organized as follows. In Section II, the related works are reviewed. In Section III, we present an FL framework and give the preliminaries for the FL process. In Section IV, the detailed FL procedures of each MEC element are presented. In Section V, the performance evaluations are conducted, followed by concluding remarks in Section VI.

II. Related Work

To improve the efficiency of an FL process, FL frameworks, protocols, and optimization methods have been studied in various situations, e.g., software defined network (SDN), MEC, and blockchain. Meanwhile, some solutions have been proposed to tackle the non-i.i.d. issue [15], [16], [17]. We give a brief review of the existing works as follows.

There are several proposals focusing on designing efficient FL frameworks and protocols. In [18], Liu et al. discuss the advantages of a multi-level federated learning framework, and propose HierFAVG to alleviate the impact of devices offline. In [19], Bonawitz et al. present the training process in a complete federated learning system and give a high-level perspective on the system design of federated learning. Besides the architectural design of the FL process, some works also focus on integrating FL with MEC to provide intelligence. In [20], Lim et al. show the potential of integrating FL with MEC to provide edge intelligence. In [21], Olowononi et al. apply FL in MEC to provide enhanced security in a vehicular cyber-physical system. In [22], Xue et al. present a fully decentralized federated framework to aggregate models of double-deep Q-network in a system by integrating SDN and MEC to balance the resource constraint and the privacy preservation requirement. In [23], Feng et al. present a two-layer blockchain architecture integrated with FL to enhance training accuracy and reduce blockchain time delay.

As a critical research topic, the study of efficient FL optimization has received attention from researchers. In [24], other than coordinate descent methods, Roux et al. propose the first SGD-type method called the stochastic average gradient (SAG) that achieves linear convergence. In [25], Zhou et al. propose a local-SGD algorithm and analyze how the local iterations affects the convergence performance of the local-SGD. In [26], Konevcny et al. divide the federated optimization issue into subproblems by DANE [27], which are optimized by SVRG [28] on a single machine. In [29], Mcmahan et al. propose a distributed algorithm FedAvg that achieves fast convergence in the case of non-i.i.d. data. In [30], Wang et al. present a control algorithm that aims to utilize the limited resources while accelerating the FL process effectively. In [31], Li et al. propose FedProx to improve the FedAvg algorithm under statistical heterogeneity by setting different precisions for devices, which train the model locally according to the precision. In [32], Li et al. study the impact of the essential parameters on convergence and prove the existence of the optimal communication rounds. In [33], Acar et al. propose a dynamic regularizer for each device to align the local models with the optimal global model and reduce transmission costs.

Besides improving training efficiency by designing various optimization methods, some solutions have been presented to solve the non-i.i.d. issue in the FL process. For example, in [34], Sattler et al. extend the top-k sparsification technique by proposing sparse ternary compression, which guarantees effectiveness under non-i.i.d. conditions. In [35], Wu et al.

propose a communication reduction algorithm for FL (FedSCR) by discarding unimportant client updates for the global learning process. In [36], Huang et al. propose a federated attentive message passing scheme (FedAMP) to facilitate similar clients to collaborate and tackle the non-i.i.d. issue in a personalized FL way. In addition, utilizing the shared data to alleviate the non-i.i.d. issue has been studied. In [37], Yue et al. analyze the impact of highly skewed non-i.i.d. data and propose a strategy to tackle the non-i.i.d. issue by sharing a few data between all edge devices. In [38], Chiu et al. propose a new operation called federated swapping (FedSwap), which replaces partial FL operations based on a few shared data to alleviate the impact of weight divergence. In [39], Karimireddy et al. address the non-i.i.d. issue by using a controlled averaging step to stabilize the global model across all participating clients. In [40], Wang et al. propose a two-stage optimization approach that adapts the local objective function of each client to minimize the difference between the local optimal solution and the global optimal solution due to data heterogeneity.

Although these works aim to provide efficient and privacy-preserving edge intelligence or sophisticated optimization methods, how to utilize the characteristics of MEC to improve FL training efficiency has not been thoroughly investigated. The mentioned optimization methods and the solutions to the non-i.i.d. issue are usually based on specific situations, and are difficult to adapt to the hierarchical MEC architecture. The main aims of this paper include: i) maintaining high communication efficiency even if the network connection is unstable, and ii) alleviating the non-i.i.d. impact in the FL training process to reduce communication rounds further. Hence, we present the theoretical analysis of the optimal communication rounds and the feasibility of the auxiliary dataset constructed by the non-sensitive data generated and maintained in the edge server. In addition, we design a dynamic epochs adjustment method and an ES co-training scheme to achieve our purposes. Consequently, we propose an enhanced hierarchical FL scheme, i.e., ACFL, in MEC to utilize the features of MEC and improve the training efficiency of FL in MEC.

III. System Architecture and Preliminaries

A. System Architecture

As shown in Fig. 1, we present a three-layer hierarchical FL architecture, including end devices, edge servers, and a cloud server. In the architecture, an enhanced FL scheme, i.e., ACFL, is designed and illustrated as follows.

In ACFL, the network is divided into multiple regions, and each edge server handles the end devices' training tasks in its region. A cloud server manages the FL training process and aggregates a global model. The cloud server initially selects edge servers to participate in the training process. Then, the cloud server receives the region-level aggregated models from edge servers and aggregates them to obtain the global model.

An edge server establishes wireless links with the selected end devices to transmit model parameters. As shown in Fig. 1, the edge server functions as both a trainer and an aggregator in ACFL. As an aggregator, the edge server aggregates the models

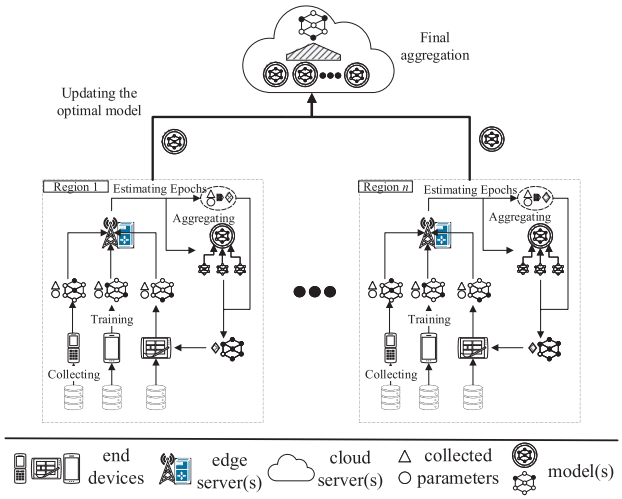


Fig. 1. The three-layer FL architecture in ACFL.

from participating end devices and conducts preliminary model integration. On the other hand, an edge server may also maintain non-sensitive data locally. Unlike the typical FL process, the data maintained in edge servers can be used to construct the auxiliary dataset and allow edge servers to train the model by themselves, improving the training efficiency and robustness of the presented ACFL scheme.

As an essential part of the architecture, an end device participating in the FL process undertakes the training task by training the model with its local data. After individual local training, the end device transmits the updated model to the edge server without external data exchange.

The FL training process of ACFL is shown in Algorithm 1. First, the cloud server selects edge servers and distributes the required accuracy to them. Then, each edge server selects participating end devices and distributes the model to the selected end devices. After that, the end device trains the model with the maintained data and transmits it to the connected edge server. The edge server aggregates the models from participating devices and decides whether to adopt the presented ES co-training scheme to obtain a regional model. If the regional model satisfies the accuracy requirement, the edge server sends it to the cloud server. Then, the cloud server aggregates the received regional models into a global one. Otherwise, the edge server sends the regional model to the selected end devices, which serves as the new initial model in the next round of training. Finally, the FL training process of ACFL is completed.

B. Preliminaries

To further illustrate the FL process in ACFL, the following preliminaries are presented [41], and the relevant symbols and descriptions are presented in Table I. Let $|\cdot|$ denote the cardinality of a set, $\mathbb{P}$ denote the probablity, and $||\cdot||$ denote the L^2 norm. We focus on finding a function $\varphi : \mathcal{X} \to \mathcal{Y}$ to give the prediction of y according to the input $\boldsymbol{x} \in \mathcal{X}$, where $\mathcal{X}$ and $\mathcal{Y}$ represent the input space and the output space, respectively.

Algorithm 1: The Procedures of ACFL.

```
Cloud server selects participating edge servers;
foreach selected edge server do
    Select participating end devices;
    foreach selected end device do
        Train model;
        Send the model to the edge server;
    Aggregate the received models.
    if the condition of the auxiliary dataset is satisfied then
        Edge server trains the aggregated model with the auxiliary dataset;
    Obtain a regional model
    if the accuracy requirement is satisfied then
        Send the regional model to the cloud server;
    Continue;
Aggregate the received regional models;
Obtain a global model;
```

TABLE I
DESCRIPTION OF THE PARAMETERS

Symbols	Description
$\lvert\cdot\rvert$	the cardinality of a set
δ	gradient divergence
D_j	data in region of edge server j
η	step size during the training
e	training epochs per-round
e^*	the corresponding epochs of $R_{\min}$
$\hat{e}$	the estimation of e^*
$\mathbb{R}^d$	d-dimensions space
μ	lower bound of the gap between $F(\boldsymbol{w}(t))$ and $F(\boldsymbol{w}*)$
ν	upper bound of the gap between $\boldsymbol{w}^{(c)}$ and $\boldsymbol{w}*$
n_i	data in end device i
R	communication rounds
$R_{\min}$	the minium of communication rounds R
$\hat{R}_{\min}$	the estimation of $R_{\min}$
$R_{\min}^{(s)}$	$R_{\min}$ by utilizing auxiliary data
T	total training iterations
$\boldsymbol{w}$	the vector of model parameters
$\boldsymbol{w}*$	the optimal solution of objective function
$\hat{\boldsymbol{w}}$	the sub-optimal solution of objective function
$\boldsymbol{w}^{(c)}$	model parameters of the centralized SGD
$\boldsymbol{w}_i$	model parameters of single device i in the region
$\boldsymbol{w}_i^j$	model parameters of device i in region j
$\boldsymbol{w}^j$	model parameters of region j in global aggregation
$\boldsymbol{w}^{(s)}$	model parameters involving auxiliary dataset

Let $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$ denote the sample set for training, the vector $\boldsymbol{x}_i \in \mathcal{X} \in \mathbb{R}^{d_x}$ represents the features of sample i, and $y_i \in \mathcal{Y} \in \mathbb{R}^{d_y}$ is the label of sample i, where $i \in \{1, \ldots, n\}$. To minimize the gap between the prediction φ_i and the label y_i, the empirical risk $R_n(h)$ is formulated as

$$R_n(\varphi) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{P}\left[\varphi_i(\boldsymbol{x}_i) \neq y_i\right], \tag{1}$$

Utilizing different prediction functions for each sample leads to a complicated empirical risk and makes it hard to fit all samples. To generalize prediction function $\varphi_i(\boldsymbol{x})$ to all sample data, φ has a fixed form and is parameterized by a vector $\boldsymbol{w} \in \mathbb{R}^d$. The parameterized prediction function $\varphi(\boldsymbol{x})$ is denoted as

$$\varphi(\cdot;\cdot), \mathbb{R}^{d_x} \times \mathbb{R}^d \to \mathbb{R}^{d_y}. \tag{2}$$

Usually, a loss function $\ell(\varphi(\boldsymbol{x}; \boldsymbol{w}), y)$ is utilized to measure the empirical risk in (1), and we have

$$R_n(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell(\varphi(\boldsymbol{x}_i; \boldsymbol{w}), y_i) = \frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{w}). \tag{3}$$

To simplify the expression, let $f(\boldsymbol{w}) = \ell(\varphi(\boldsymbol{x}_i; \boldsymbol{w}), y_i)$, and the objective function can be expressed as

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} F(\boldsymbol{w}) = \min_{\boldsymbol{w}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{w}). \tag{4}$$

Similarly, the FL loss function $F(\boldsymbol{w})$ consists of loss functions $F(\boldsymbol{w}_i)$ in several end devices. Denote n_i as the data maintained in device i and $|n_i|$ as the cardinality of set n_i, and the FL loss function consisting of m devices is

$$F(\boldsymbol{w}) = \sum_{i=1}^{m} \frac{|n_i|}{\sum_{j=1}^{m} |n_j|} F(\boldsymbol{w}_i), \tag{5}$$

where m is the number of participating devices.

IV. Adaptive Collaborative Federated Learning

In this section, we present the detailed ACFL process under the considered architecture. Denote e as the local epochs, R as the communication rounds between the end devices and the edge server, and T as the total training iterations during the ACFL process. The correlation of R, T, and e can be represented as

$$T = R \cdot e. \tag{6}$$

To minimize empirical risk of FL in (5), we need to find the optimal solution $\boldsymbol{w}^*$, and we have

$$\boldsymbol{w}^* = \arg\min F(\boldsymbol{w}). \tag{7}$$

However, the optimal solution of (7) is hard to obtain. Since theoretically, the true data distribution is required to confirm $F(\boldsymbol{w}^*)$, which is impractical. Hence, instead of obtaining the optimal solution, a sub-optimal solution $\hat{\boldsymbol{w}}$ satisfying the accuracy requirement is admirable, which can be represented as

$$F(\hat{\boldsymbol{w}}) - F(\boldsymbol{w}^*) \leq \Delta. \tag{8}$$

(8) means if Δ is sufficiently small, $\hat{\boldsymbol{w}}$ can be considered as a favorable solution of (7).

Algorithm 2: The Procedures of ACFL in a Cloud Server

```
Initialize accuracy requirement Δ ;
Select participating edge servers ;
Send the accuracy requirement Δ to the selected
  edge servers;
Receive the model parameters returned from edge
  servers;
if the received models satisfy the accuracy requirement
  then
    Aggregate the models and obtain ŵ;
    Broadcast the result to all end devices;
    Training process stops;
```

A. The ACFL Process in Cloud Server

The cloud server aggregates the regional models from the selected edge servers in ACFL. In practice, frequent communications between edge servers and the cloud server usually lead to high communication costs and long communication time, even the risk of privacy leakage. Reducing the communication rounds between edge and cloud servers is a rational way to alleviate these issues. In the proposed architecture, we prove that the communication rounds between edge and cloud servers can be reduced to twice, one for distributing the initial model and the other for aggregating the global model.

Lemma 1: In the three-layer architecture, if the regional aggregated models satisfy the condition in (8), the sub-optimal solution, i.e., the global model $\hat{\boldsymbol{w}}$ can be formulated as

$$F(\boldsymbol{w}^j) - F(\boldsymbol{w}^*) \leq \Delta \Rightarrow F(\hat{\boldsymbol{w}}) - F(\boldsymbol{w}^*) \leq \Delta, \tag{9}$$

where $\boldsymbol{w}^j$ is the parameter of the model aggregated in edge server j and $\boldsymbol{w}^*$) is the optimal solution.

According to Lemma 1, if the selected edge servers can provide the model satisfying the accuracy requirement in (8), the cloud server only performs one global model aggregation to obtain a sub-optimal solution to satisfy (8). The ACFL process in the cloud server is presented in Algorithm 2. The cloud server initially initializes Δ as the accuracy requirement for the sub-optimal solution. Then, it distributes the accuracy requirement to the selected edge servers and waits to receive the regional models that satisfy the accuracy requirement from edge servers. After receiving the regional models, the cloud server aggregates them and obtains a global model. Finally, the cloud server broadcasts the global model to all end devices through edge servers, and then the FL training process is terminated.

B. The ACFL Process in Edge Server

In this subsection, we illustrate the ACFL process in edge servers. Two improved schemes are designed to decrease the communication rounds between end devices and the edge server in a region and improve training efficiency and model reliability. According to Lemma 1, the aggregation process in the cloud server is significantly decreased if each regional model satisfies the accuracy requirement in (8). Hence, our goal is to train a regional model that satisfies the accuracy requirement in each edge server, and we have

$$\boldsymbol{w}^j = \arg\min F(\boldsymbol{w}^j(T)). \tag{10}$$

To simplify the expression, let $\boldsymbol{w}_i(t)$ denote $\boldsymbol{w}_i^j(t)$, which means device i connects to the edge server in the region j, i.e., edge server j. Similar to FedAvg, we maintain a local model $\boldsymbol{w}_i(t)$ at device i, where $t = \{0, 1, 2, 3 \cdots, n\}$ represents the index of local epochs executing in the device i. To avoid the impact of different initial models, end devices in a region are assigned to the same initial model at time $t = 0$. An end device updates its model based on a gradient descent method, and we have

$$\boldsymbol{w}_i(t) = \boldsymbol{w}_i(t-1) - \eta \nabla F(\boldsymbol{w}_i(t-1)), \tag{11}$$

where $\eta > 0$ is the step size. After e epochs, each end device uploads its model to the corresponding edge server, and then the edge server aggregates the modes of end devices to obtain a regional model. The aggregation process can be formulated as

$$\boldsymbol{w}^j(t) = \sum_{i=1}^{n_{sum}^j} \frac{|n_i^j|}{\sum_{i'=1}^{n_{sum}^j} |n_{i'}^j|} \boldsymbol{w}_i(t), \tag{12}$$

where n_{sum}^j is the number of end devices in region j. The edge server aggregates models only at moment $t = ke$, where $k \in \{1, 2, \ldots, R\}$ denotes the index of communication round. We make the following assumptions on function $F(\cdot)$ to perform the convergence analysis.

Assumption 1: For device i, suppose that the loss function $F(\boldsymbol{w})$ is convex for all $\boldsymbol{w}$ and $\boldsymbol{v}$, i.e.,

$$\nabla F(\boldsymbol{w})(\boldsymbol{v} - \boldsymbol{w}) \leq F(\boldsymbol{v}) - F(\boldsymbol{w}).$$

Assumption 2: For device i, suppose that the loss function $F(\boldsymbol{w})$ is L-lipschitz continuous, i.e.,

$$||F(\boldsymbol{v}) - F(\boldsymbol{w})|| \leq L||\boldsymbol{v} - \boldsymbol{w}||.$$

Assumption 3: For device i, suppose that the loss function $F(\boldsymbol{w})$ is β-smooth, i.e.,

$$F(\boldsymbol{v}) \leq F(\boldsymbol{w}) + (\boldsymbol{v} - \boldsymbol{w})\nabla F(\boldsymbol{w}) + \frac{L}{2}||\boldsymbol{v} - \boldsymbol{w}||^2.$$

In addition, we define gradient divergence to represent the divergence between the gradient of a local loss function and the regional loss function.

Definition 1 (Gradient Divergence): For any device i and its model parameter $\boldsymbol{w}_i$, define δ_i as the upper bound of $||\nabla F(\boldsymbol{w}) - \nabla F_i(\boldsymbol{w}_i)||$

$$||\nabla F(\boldsymbol{w}) - \nabla F(\boldsymbol{w}_i)|| \leq \delta_i, \tag{13}$$

where $\nabla F(\boldsymbol{w})$ denotes the aggregated model. Similar to (12), let δ^j denote the gradient divergence in the region of edge server j as

$$\delta^j = \sum_{i=1}^{n_{sum}^j} \frac{|n_i^j|}{\sum_{i'=1}^{n_{sum}^j} |n_{i'}^j|} \delta_i. \tag{14}$$

Because the following analysis is based on one edge server and the corresponding end devices, we simplify δ^j to δ for convenience. The degree of divergence in data distribution of end devices determines the non-i.i.d. impact on FL convergence.

To describe the FL convergence, we utilize Theorem 1 of [30] and transform it into (15). Let the step size satisfy $\eta \leq \frac{1}{\beta}$, the convergence upper bound of the FL process after T iterations is presented as follow

$$F(\boldsymbol{w}(T)) - F(\boldsymbol{w}^*) \leq \frac{1}{T\left(\nu\eta\left(1-\frac{\beta\eta}{2}\right)-\frac{Lh(e)}{e\mu^2}\right)}, \quad (15)$$

where ν is minimum of $\frac{1}{||\boldsymbol{w}^c(t)-\boldsymbol{w}^*||^2}$ in the kth round, $\boldsymbol{w}^c(t)$ is the training solution by utilizing the centralized SGD as auxiliary variables, μ is the lower bound of empirical risk loss between the sub-optimal and optimal solutions, and $h(e) = \frac{\delta}{\beta}(\eta\beta+1)^e - \frac{\delta}{e}\eta$. At the beginning of each round, $\boldsymbol{w}^c(t)$ is initialized to $\boldsymbol{w}^j(t)$, and the auxiliary variable $\boldsymbol{w}^c(t)$ is updated by

$$\boldsymbol{w}^c(t) = \boldsymbol{w}^c(t-1) - \eta\nabla F(\boldsymbol{w}^c(t-1)). \quad (16)$$

Combining (15), we can rearrange the accuracy requirement in (8) as

$$\frac{1}{T\left(\nu\eta\left(1-\frac{\beta\eta}{2}\right)-\frac{Lh(e)}{e\mu^2}\right)} \leq \Delta. \quad (17)$$

1) Model Reliability Improvement With Communication Rounds Optimization: In this part, we aim to reduce the model degradation caused by packet loss in the communications between end devices and the edge server. The quality of the aggregated model in the edge server depends on the models transmitted by end devices. Hence, the packet loss in the communication process may lead to model parameters loss, which degrades the quality of the aggregated model in the edge server, may slow down the training process, and even cause the FL process fail to converge.

To analyze the possibility of packet loss in the FL process, we model the wireless link between end device j and the edge server as a tuple (ϵ_j, p_j), where ϵ_j is the packet length and p_j is the packet successful transmission probability. Since the model updating process in FL is constrained by the packet length ϵ_j, we divide the model parameter w_j of device j into multiple packets. Assuming that no redundant packets are sent, the local model can be divided into $\lceil\frac{|w_j|}{\epsilon_j}\rceil$ packets. Thus, the number of packets required for a successful model update is denoted by $N_j^u = \lceil|w_j|/\epsilon_j\rceil$. These transmissions are independently and identically distributed following the *Geometric* $(p = 1 - p_j)$ distribution, shown as below

$$\mathbb{P}(X \geq N_j^u) = p_j^{N_j^u} = p_j^{\lceil|w_j|/\epsilon_j\rceil}. \quad (18)$$

The probability of device j successfully transmitting its model in one round can denote by $p_{sj} = p_j^{\lceil|w_j|/\epsilon_j\rceil}$. Assuming device j participates in R training rounds, we can also model the probability of stable and successful model updates as a *Geometric* $(p = 1 - p_{sj})$ distribution. The probability of successful model transmissions in R rounds is denoted as

$$\mathbb{P}(X \geq R) = p_{sj}^R = p_j^{R\cdot N_j^u} = p_j^{R\cdot\lceil|w_j|/\epsilon_j\rceil}. \quad (19)$$

(19) shows that the wireless channel conditions and the number of communication rounds constrain the model update process of each device. Assume that the packet successful transmission probability p_j and the packet length ϵ_j are constant for device j. The successful model transmission probability is determined by communication rounds R. Hence, we aim to minimize the communication rounds R to decrease the probability of packet loss in model transmission and enhance the model transmission reliability.

Eq. (17) shows that the convergence upper bound depends on the total training iterations T, as well as the sub-optimal solution accuracy requirement Δ. According to the definition of total iterations $T = R \cdot e$, tuning the local training epochs e can affect communication rounds R, when the accuracy requirement Δ is fixed. Hence, combining the accuracy requirement in (17), the optimization problem can be formulated as

$$\min R \quad (20a)$$

$$\text{s.t. } T = Re \quad (20b)$$

$$\frac{1}{T\left(\nu\eta\left(1-\frac{\beta\eta}{2}\right)-\frac{Lh(e)}{e\mu^2}\right)} \leq \Delta. \quad (20c)$$

Combining (20b) and (20c), we obtain a new constraint

$$Re\left(\nu\eta\left(1-\frac{\beta\eta}{2}\right)-\frac{Lh(e)}{e\mu^2}\right) \geq \frac{1}{\Delta}. \quad (21)$$

(21) presents the correlation between the local epochs e and communication rounds R, which means R is inversely proportional to e with a constant accuracy requirement Δ. We define the function $H(e)$ to model the correlation in (21) as follow

$$H(e) \triangleq e\left(\nu\eta\left(1-\frac{\beta\eta}{2}\right)-\frac{Lh(e)}{e\mu^2}\right). \quad (22)$$

Take $H(e)$ into (21), we can simplify the objective function as

$$\min R$$
$$\text{s.t. } \frac{1}{\Delta H(e)} \leq R. \quad (23)$$

According to (23), the minium communication rounds R depends on the maximum of $H(e)$. Let e^* as the optimal value of e, where $H_{\max} = H(e^*)$, the optimal solution of (23) can be obtained in Lemma 2.

Lemma 2: With the constraint of (21), the minimum communication rounds $R_{\min}$ can be obtained as follow

$$R_{\min} = \frac{1}{\Delta H_{\max}}. \quad (24)$$

According to the monotonicity of function $H(e)$, the corresponding epochs e^* to $R_{\min}$ can be formulated as

$$e^* = \frac{\ln\left(\eta\beta + \frac{\nu\eta\mu^2\beta\left(1-\frac{\beta\eta}{2}\right)}{L\delta}\right) - \ln\ln(\eta\beta+1)}{\ln(\eta\beta+1)}, \quad (25)$$

and $H_{\max}$ is formulated as

$$H_{\max} = \frac{L\delta}{\mu^2\beta}((e^*\ln(\eta\beta+1)-1)(\eta\beta+1)^{e^*}). \quad (26)$$

Proof: The proof of the Lemma is provided in Appendix B, available online. □

2) Design of a Dynamic Epoch Adjustment Method: In this part, an estimation of the optimal epoch e^* corresponding to $R_{\min}$ is presented, and the dynamic epoch adjustment is designed. The optimal number of local epochs e^* is formulated as

$$e^* = \frac{\ln\left(\eta\beta + \frac{\nu\eta\mu^2\beta\left(1-\frac{\beta\eta}{2}\right)}{L\delta}\right) - \ln\ln(\eta\beta+1)}{\ln(\eta\beta+1)}. \tag{27}$$

However, ν and μ^2 in (27) cannot be obtained in practice. Therefore, we need to give a computable form of the optimal solution. According to the definitions of ν and μ^2 in (15), $\nu\mu^2$ can be expressed as

$$\nu\mu^2 = \frac{\min ||F(\boldsymbol{w}^c(t)) - F(\boldsymbol{w}^*)||^2}{\max_k ||\boldsymbol{w}^c(t) - \boldsymbol{w}^*||^2}, \tag{28}$$

where $\max_k||\boldsymbol{w}^c(t) - \boldsymbol{w}^*||^2$ is the maximum of differences between the solutions with the centralized SGD and the optimal solution in the kth communication round. Since $F(\boldsymbol{w})$ has the L-lipschitz continuous assumption, we have $||F(\boldsymbol{w}) - F(\boldsymbol{w}^*)|| \leq L||\boldsymbol{w} - \boldsymbol{w}^*||$ for each $\boldsymbol{w}$. Utilizing the L-lipschitz condition of $F(\cdot)$, (28) can be scaled as

$$\nu\mu^2 \leq \frac{\min L^2||\boldsymbol{w}^c(t) - \boldsymbol{w}^*||^2}{\max_k ||\boldsymbol{w}^c(t) - \boldsymbol{w}^*||^2}. \tag{29}$$

Here the notation ζ is used to simplify the expression form. Denote ζ as

$$\zeta = \frac{\min ||\boldsymbol{w}^c(t) - \boldsymbol{w}^*||^2}{\max_k ||\boldsymbol{w}^c(t) - \boldsymbol{w}^*||^2} \leq 1. \tag{30}$$

Hence, (29) can be formulated as

$$\nu\mu^2 \leq \frac{\min L^2||\boldsymbol{w}^c(t) - \boldsymbol{w}^*||^2}{\max_k ||\boldsymbol{w}^c(t) - \boldsymbol{w}^*||^2} = \zeta L^2. \tag{31}$$

According to (31) and the relation $\eta \leq \frac{1}{\beta}$, $\eta\beta \in (0, 1]$, and we can have

$$\begin{aligned}\eta\beta + \frac{\nu\eta\mu^2\beta\left(1-\frac{\beta\eta}{2}\right)}{L\delta} \\ = 1 + \frac{\zeta L}{\delta}\left(\eta\beta - \frac{(\eta\beta)^2}{2}\right) \leq 1 + \frac{L\zeta}{2\delta}.\end{aligned} \tag{32}$$

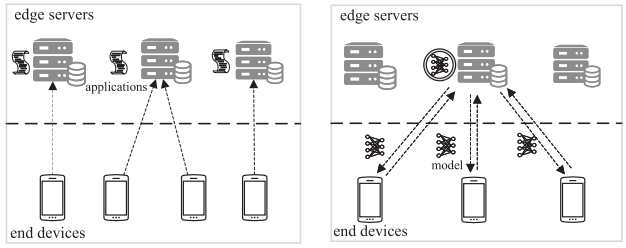
Combining (27) and (32), we can have the estimation of e^* as

$$e^* \leq \hat{e} = \frac{\ln\left(\frac{L\zeta}{2\delta} + 1\right) - \ln\ln(\eta\beta+1)}{\ln(\eta\beta+1)}. \tag{33}$$

The parameters in (33) can be estimated by the local data [30], including δ, L, and β. Hence, the accuracy of the approximate solution $\hat{e}$ depends on the parameter ζ. With a suitable ζ, $\hat{e}$ can achieve the optimal solution e^*. We call such an estimating process as dynamic epochs adjustment.

Since $R_{\min} = \frac{1}{\Delta H_{\max}}$, $R_{\min}$ can be obtained with $H_{\max}$. When $H(e) = H_{\max}$, $\nabla H(e) = 0$ can be derived, and the expression of $H_{\max}$ is given as

$$H_{\max} = \frac{L\delta}{\mu^2\beta}((e^*\ln(\eta\beta+1) - 1)(\eta\beta+1)^{e^*}). \tag{34}$$



(a) The applications running in edge servers may generate data that is called edge server data.

(b) The edge server data can be utilized to construct an auxiliary dataset to improve the federated learning efficiency.

Fig. 2. Sources of auxiliary datasets in the edge servers of ACFL.

Combining the lower bound of convex optimization in [42] represented as Lemma 3, μ can be expressed by a more explicit form.

Lemma 3 (lower bound of convex optimization [42]): For $1 \leq T \leq \frac{1}{2}(d-1)$ and $\boldsymbol{w}_0 \in \mathbb{R}^d$, there exists a smooth convex function F such that for any first order method, we have

$$F(\boldsymbol{w}) - F(\boldsymbol{w}^*) \geq \frac{3L||\boldsymbol{w}_0 - \boldsymbol{w}^*||^2}{32(T+1)^2} = \hat{\mu}, \tag{35}$$

where T is the number of total iterations in the whole optimization process and d is the dimensions of parameter $\boldsymbol{w}$.

To utilize the Lemma 3 to estimate parameter μ in practice, we need to differentiate the dimensions d and the number of features in $\boldsymbol{w}$. We can consider $\boldsymbol{w}$ as the vector from a higher dimension space than the features of models. The features can be considered as the necessary dimensions for the training process, but other dimensions are meaningless for the training process. As a result, the requirement $1 \leq T \leq \frac{1}{2}(d-1)$ can be ignored when utilizing Lemma 3 in practice. Utilizing $\hat{\mu}$, $R_{\min}$ can be estimated as $\hat{R}_{\min}$, and we have

$$\hat{R}_{\min} = \frac{\hat{\mu}^2\beta}{\Delta L\delta[(\hat{e}\ln(\eta\beta+1) - 1)(\eta\beta+1)^{\hat{e}}]}. \tag{36}$$

3) ES Co-Training Scheme With Auxiliary Dataset: In this part, we propose an edge server collaborative training scheme, which utilizes the data maintained in edge servers to mitigate the non-i.i.d. issue and decrease the communication rounds.

In the MEC environment, collaborative paradigms are adopted where end devices collaborate with edge servers to perform tasks like model training [43], [44], [45], computation offloading [46], and service migration [47]. As shown in Fig. 2(a), an edge server may run multiple applications, and it can generate and maintain its own data locally that can be utilized to improve the training efficiency. With these edge server data, we can then build an auxiliary dataset to mitigate the non-i.i.d. issue and improve training efficiency, as depicted in Fig. 2(b). Currently, solutions [37], [38] reduce the impact of skewed data to improve FL efficiency involving edge server data. However, utilizing edge server data may not always improve the training performance due to the uncertain distribution of the edge server data. Therefore, we derive a condition that can evaluate the

feasibility of using edge server data as an auxiliary dataset in FL and demonstrate the corresponding theoretical analysis.

Based on Lemma 2, the lower bound of $R_{\min}$ directly relates to $H(e)$. As a part of $H(e)$, $h(e)$ in (22) denotes the impact of non-i.i.d. issue, and it is related to the number of epochs e and the gradient divergence δ. Therefore, reducing the gradient divergence δ can alleviate the impact of the non-i.i.d. issue and further decrease the communications rounds. We define the global data as the total data distributed in end devices and edge servers, and a global dataset means that it has the same data distribution as the global data. Intuitively, training a model with a dataset with a similar data distribution to the global data than the data of end devices leads to a minor gradient divergence δ. In addition, compared with the data maintained in edge servers, the data maintained in each end device may have more skewed data.

With proper distribution of the auxiliary dataset, the non-i.i.d. issue can be alleviated. Hence, we can obtain a tighter convergence upper bound, which further enhances the effectiveness of the dynamic epoch adjustment method. In order to better illustrate this issue, the following analysis focuses on a region with K end devices and an edge server, and the cross-entropy loss function is adopted as an instance to demonstrate the effect of data distribution [37] as follow

$$\ell(\boldsymbol{w}) = \mathbb{E}_{\boldsymbol{x},y\sim p}\left[\sum_{i=1}^{C}\log f_i(\boldsymbol{x},\boldsymbol{w})\right] = \sum_{i=1}^{C}\mathbb{E}_{\boldsymbol{x},y\sim p}\left[\log f_i(\boldsymbol{x},\boldsymbol{w})\right] \tag{37}$$

where C is the data class among global data. In order to illustrate how the non-i.i.d. issue affects the gradient divergence δ, let $p^{(k)}$ represent the data distribution of device k. Combining Definition 1 and the loss function in (37), the device k's gradient divergence δ_k and the total gradient divergence δ can be formulated as

$$\delta_k = g_{\max}\sum_{i=1}^{C}||p^{(k)}(y=i) - p(y=i)|| \tag{38}$$

$$\delta = \sum_{i=1}^{K}\frac{n_i}{\sum_{j=1}^{K}n_j}\delta_i, \tag{39}$$

where $g_{\max}$ is denoted as

$$g_{\max} \triangleq \max_{i=\{1,2,\ldots,C\}}||\mathbb{E}_{x,y\sim p}[\log f_i(x,\boldsymbol{w})]||. \tag{40}$$

With a specific form of the gradient divergence in (38) and (39), we formally illustrate the correlation between the gradient divergence and the heterogeneous data distributions of end devices. Consequently, training on an auxiliary dataset with a similar data distribution with the global data can efficiently decrease the gradient divergence and thus mitigate the impact of the non-i.i.d. issue. Hence, we propose a collaborative training process with the auxiliary dataset in the edge server to decrease the gradient divergence.

The collaborative training process with the auxiliary dataset is shown in Fig. 3. The edge server sends the initial regional model to the end devices. Then each end device completes training and delivers its model to the edge server. The edge server aggregates the received models, and then trains the aggregated model with the auxiliary dataset to obtain the regional model in the round. Such a process is called the edge server collaborative training scheme (ES co-training).

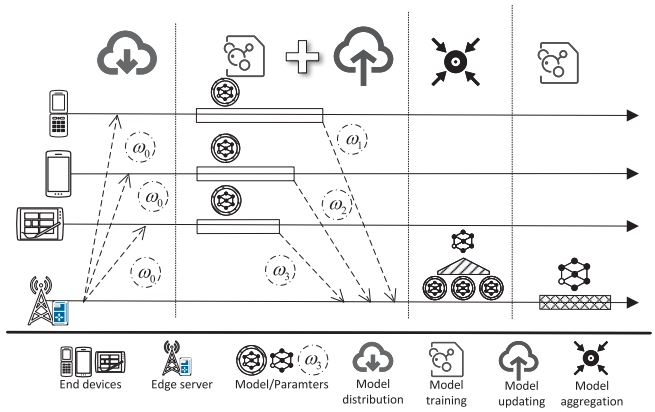


Fig. 3. The edge server collaborative training scheme with auxiliary dataset.

Let the epochs performed on an end device be e_1, and the epochs performed on the edge server be e_2. Intuitively, increasing e_2 can facilitate the federated learning process when the auxiliary dataset has a data distribution similar to the global data. Since the data distribution of the auxiliary dataset is uncertain, increasing e_2 may degrade the training efficiency. Subsequently, we formally analyze the effect of the data distribution of the auxiliary dataset in the training process and demonstrate the theoretical result in Lemma 4.

Lemma 4: If Assumptions 1, 2, and 3 hold for function $F(\boldsymbol{w})$, then the difference between the solution with the ES co-training scheme and that with the centralized SGD can be denoted as

$$\begin{aligned}&||\boldsymbol{w}^{(s)}(t) - \boldsymbol{w}^{c}(t)||\\ &\le \frac{\delta^{(s)}}{\beta}((\eta\beta+1)^{e_2}-1)(\eta\beta+1)^{e_1} + h(e_1)\\ &= h(e_1) - h^{(s)}(e_1) + h^{(s)}(e) + e_2\delta^{(s)},\end{aligned} \tag{41}$$

where $\delta^{(s)}$ is the gradient divergence of the training process with the auxiliary dataset, $\boldsymbol{w}^{c}(t)$ is the solution with the centralized SGD, $\boldsymbol{w}^{(s)}(t)$ is the solution involving the auxiliary dataset, and $e = e_1 + e_2$ is the local training epochs.

Proof: The proof of the Lemma is provided in Appendix C, available online. □

Lemma 4 demonstrates the difference between the solution with the auxiliary dataset and that with the centralized SGD, which can be considered as the term $h(e)$ in (15). Thus the convergence upper bound of the FL process with the ES co-training scheme can be formulated as in Theorem 1.

Theorem 1: If Assumptions 1, 2, and 3 hold for function $F(\cdot)$, the convergence upper bound of the FL process with the ES co-training scheme after T iterations is given by

$$F(\boldsymbol{w}(T)) - F(\boldsymbol{w}^*) \le \frac{1}{T\left(\nu\eta\left(1-\frac{\beta\eta}{2}\right) - \frac{L\rho(e)}{e\mu^2}\right)}, \tag{42}$$

where $\rho(e)$ can be formulated as

$$\rho(e) \triangleq h(e_1) - h^{(s)}(e_1) + h^{(s)}(e) + \eta\delta^{(s)}e_2. \quad (43)$$

Compared with the upper bound in (15), the upper bound in Theorem 1 only changes $h(e)$ to $\rho(e)$. In order to prove that training process with the ES co-training scheme achieves a tighter boundary in Theorem 1, $h(e) \geq \rho(e)$ should be satisfied. The corresponding condition is demonstrated in Lemma 5.

Lemma 5: The upper bound in Theorem 1 is tighter than the bound in (15), if $\delta^{(s)}$ satisfies the following condition

$$\delta^{(s)} < \left(1 - \frac{1}{\ln(\eta\beta+1)} + \frac{1}{(\eta\beta+1)\ln(\eta\beta+1)}\right)\delta. \quad (44)$$

Proof: The proof of the Lemma is provided in Appendix D, available online. □

Lemma 5 can be considered as the feasible condition of using an auxiliary dataset. The condition validates that the ES co-training is efficient if the data distribution of the auxiliary dataset is similar to the distribution of global data.

If the condition in Lemma 5 is satisfied, let $e = e^*$, the optimal communication rounds obtained by the dynamic epoch adjustment method can be formulated as

$$R_{\min}^{(s)} = \frac{1}{\Delta e\left(\nu\eta\left(1 - \frac{\beta\eta}{2}\right) - \frac{L\rho(e)}{e\mu^2}\right)} \leq R_{\min}, \quad (45)$$

where $R_{\min}^{(s)}$ denotes the communication rounds when using the auxiliary dataset with epochs e^*. For the fixed accuracy requirement Δ, utilizing auxiliary datasets in edge servers achieves lower communication rounds $R_{\min}^{(s)}$ than $R_{\min}$. Hence, adapting the ES co-training scheme with a feasible auxiliary dataset can further reduce the communication rounds while alleviating the effect of non-i.i.d. data. In addition, model training in a powerful edge server consumes less time than in end devices, further speeding up the training process in practice. Therefore, the edge servers involved in the training process can mitigate end device burdens.

4) Adaptive Federated Learning With Auxiliary Dataset: According to the theoretical results, both the dynamic epochs adjustment method and the ES co-training scheme can accelerate the training process and reduce the communication rounds. As a critical part of MEC, edge servers are located between the cloud server and end devices, which makes them easy to manage and collaboratively train models with other parts. Therefore, combining the proposed dynamic epochs adjustment method and the ES co-training scheme, we propose an adaptive federated learning algorithm with auxiliary dataset in edge servers, as shown in Algorithm 3.

As a critical part of ACFL, the adaptive federated learning algorithm with auxiliary dataset establishes the connection between an edge server and the corresponding end devices in the FL process, which can be considered as the ACFL under a two-layer architecture. The edge server sets the number of epochs executing in end devices to minimize the communication rounds according to the dynamic epoch adjustment method. Meanwhile, the ES co-training scheme is also designed to mitigate the non-i.i.d. issue and further reduce the communication rounds while improving the training efficiency of ACFL. The dynamic epochs adjustment method is illustrated in lines 13 to 18, where the edge server utilizes the received parameters to obtain $\hat{\beta}$ and $\hat{L}$, and computes $\hat{e}$ as an estimation of e^*. The ES co-training scheme is shown in lines 20 to 24. If the auxiliary dataset satisfies Lemma 5, the edge server can utilize its auxiliary dataset and control the epochs executing in the edge server with

Algorithm 3: The Procedures of ACFL in an Edge Server (i.e., the Adaptive Federated Learning Algorithm With Auxiliary Dataset).

Input: control parameter ζ, epochs division ratio α, Scaling ratio of the estimated optimal communication rounds γ
Output: $\boldsymbol{w}(t)$

1 Initialize $e \leftarrow$ initial value, $t \leftarrow 0$, $R_{count} \leftarrow 0$, $\hat{R}_{\min} \leftarrow 0$, $R_{limit} \leftarrow 0$;
2 Recieve the accuracy requirement Δ and initialize $\boldsymbol{w}(0)$ as initial model;
3 Select m participating end devices;
4 Send $\boldsymbol{w}(0)$ and e to the participating end devices;
5 **repeat**
6 $t_0 \leftarrow t$;
7 $t \leftarrow t + e$;
8 Receive $\boldsymbol{w}_i(t), \beta_i, \nabla F(\boldsymbol{w}_i(t))$ and L_i from end device i;
9 Compute $\boldsymbol{w}(t)$ according to Eq. (12);
10 **if** $||F(\boldsymbol{w}(t)) - F(\boldsymbol{w}(t_0))|| < \Delta$ **then**
11 Return $\boldsymbol{w}(t)$ to the cloud server and Stop;
12 **else**
13 Estimate $\hat{\beta} \leftarrow \sum_{i=1}^{m}\left[n_i\beta_i/(\sum_{j=1}^{m} n_j)\right]$;
14 Estimate $\hat{L} \leftarrow \sum_{i=1}^{m}\left[n_iL_i/(\sum_{j=1}^{m} n_j)\right]$;
15 Compute $\nabla F(\boldsymbol{w}(t)) \leftarrow \sum_{i=1}^{m}\left[n_i\nabla F(\boldsymbol{w}_i(t))/(\sum_{j=1}^{m} n_j)\right]$;
16 Compute $\delta_i \leftarrow ||\nabla F(\boldsymbol{w}(t)) - \nabla F(\boldsymbol{w}_i(t))||$ for each device i;
17 Compute $\delta \leftarrow \sum_{i=1}^{m}\left[n_i\delta_i/(\sum_{j=1}^{m} n_j)\right]$;
18 Compute $\hat{e}$ according to Eq. (33) ;
19 Compute $\delta^{(s)}$ of the auxiliary dataset;
20 **if** $\delta^{(s)}$ *meets Lemma 5* **then**
21 $e \leftarrow (1-\alpha)\hat{e}$;
22 Train $\boldsymbol{w}(t)$ epochs $\alpha\hat{e}$ by utilizing the auxiliary dataset;
23 **else**
24 $e \leftarrow \hat{e}$;
25 Compute $\hat{R}_{\min}$ according to Eq. (36);
26 **if** $R_{limit} < \hat{R}_{\min}$ **then**
27 $R_{limit} = \hat{R}_{\min}$;
28 $R_{count} \leftarrow R_{count} + 1$;
29 **if** $R_{count} \geq \gamma R_{limit}$ **then**
30 Stop;
31 Send e and $\boldsymbol{w}(t)$ to the end devices ;
32 **until**

Algorithm 4: The Procedures of ACFL in an end Device.

1 Initialize $t \leftarrow 0$;
2 Receive $\boldsymbol{w}(t)$ and e from the edge server;
3 Set $\boldsymbol{w}_i(t) \leftarrow \boldsymbol{w}(t)$ and $\boldsymbol{w}_0 \leftarrow \boldsymbol{w}(t)$;
4 **repeat**
5 $\quad t \leftarrow t+1$;
6 $\quad$ Perform local update and obtain $\boldsymbol{w}_i(t)$ according to Eq. (11);
7 **until** $t \geq e$
8 Estimate $\hat{\beta}_i \leftarrow ||\nabla F(\boldsymbol{w}_i(t)) - \nabla F(\boldsymbol{w}_0)||/||\boldsymbol{w}_i(t) - \boldsymbol{w}_0||$;
9 Estimate $\hat{L}_i \leftarrow ||F(\boldsymbol{w}_i(t)) - F(\boldsymbol{w}_0)||/||\boldsymbol{w}_i(t) - \boldsymbol{w}_0||$;
10 Send $\boldsymbol{w}_i(t)$, $\hat{\beta}_i$, $\nabla F(\boldsymbol{w}_i(t))$ and $\hat{L}_i$ to the edge server;

the epoch division ratio α to alleviate the non-i.i.d. issue and improve the training efficiency. The edge server chooses α and sends $(1-\alpha)\hat{e}$ to the selected end devices as the local epochs in the next round. Then, the edge server trains the model $\boldsymbol{w}(t)$ with $\alpha\hat{e}$ epochs with the auxiliary dataset. Otherwise, the edge server sends $\hat{e}$ to the selected end devices as the local epochs in the next round.

Lines 25 to 30 show the procedures that when the algorithm fails to converge. According to (36), the edge server computes $\hat{R}_{\min}$, and utilizes the maximum of $\hat{R}_{\min}$ as the upper bound of communication rounds R_{limit}. Since $\hat{e} \geq e^*$ in (33), the estimation of the optimal communication rounds $\hat{R}_{\min}$ is smaller than $R_{\min}$. To avoid the unexpected termination due to the underestimation of $R_{\min}$, a parameter $\gamma > 1$ is utilized to scale the upper bound of communication rounds R_{limit} to γR_{limit}. If the communication rounds R_{count} is larger than γR_{limit}, the edge server considers that the algorithm cannot converge and terminates the training process.

C. The ACFL Process in End Device

This subsection presents the detailed ACFL process in end devices under the considered architecture. In ACFL, end devices generate and maintain data, and are also responsible for the training tasks. Hence, the selected end devices are required to train thier models on local data and transmit these models to the edge server for aggregation.

Since the edge server is required to estimate some critical parameters such as β, L, and δ, end devices need to estimate these parameters and transmit these parameters to the corresponding edge server, which is different from the typical FL process. The detailed ACFL process in an end device is given in Algorithm 4.

At the beginning of each round, an end device receives the initial model $\boldsymbol{w}(t)$ and local epochs e from the edge server. Then, the end device trains its model $\boldsymbol{w}(t)$ according to (11). After e epochs, the end device estimates L_i, β_i as $\hat{L}_i$, $\hat{\beta}_i$. Then, the model and estimated parameters are sent to the edge server. Then, the ACFL process in the end device stops, and it cannot continue the training process unless it is selected by the edge server again.

TABLE II
SIMULATION PARAMETERS

Parameter	Description	Variable
batch size	size of data in each epoch	64
devices per round	number of training devices	10
learning rate	step size in optimization	0.1
data class	global data categories	10
step decay	step change strategy	0.99

V. SIMULATION RESULTS

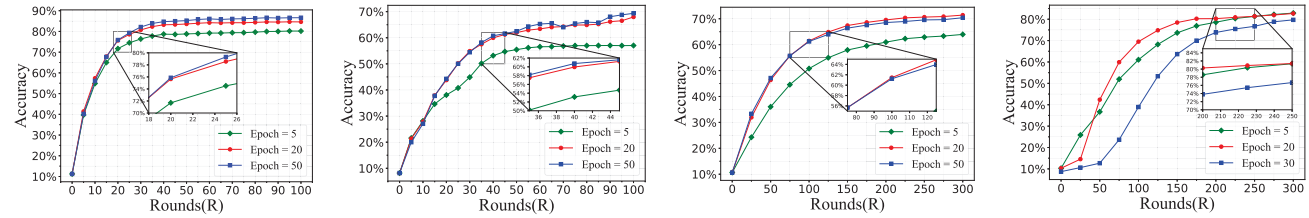
In this section, we conduct extensive simulations to evaluate the proposed ACFL scheme from three aspects: i) the existence of the optimal communication rounds and the validity of the estimation $\hat{e}$; ii) the effect of the auxiliary dataset in improving the performance of ACFL, and the impacts of data distribution of auxiliary dataset and training epochs on the auxiliary dataset; and iii) to verify that ACFL achieves a better performance both in two-layer and three-layer architectures compared with FedProx [31], FedDyn [33], Scaffold [39], and FedNova [40].

We use four datasets to verify the validity of the above theoretical results, and construct the Synthetic dataset using the same manner as reference [32]. The MNIST [48], FEMNIST [49], and CIFAR-10 [50] datasets are utilized to evaluate the performance of the proposed ACFL scheme.

To investigate the non-i.i.d. issue in the FL process, we refer to the following division methods of the MNIST dataset. We divide the MNIST dataset into 10 classes (0-9) according to different data types. The whole dataset is distributed to 100 end devices in a non-i.i.d. manner, and each end device maintains two kinds of data. In addition, we also unbalance the amounts of data in different end devices following a power law [32]. We keep the high volume data (i.e., data classes 0-9) and utilize the original division of FEMNIST (i.e., 100 clients, and each client has 600 data samples). We also distribute 60000 samples in CIFAR-10 to 100 clients, each of which has two kinds of data to simulate the non-i.i.d. issue. The two-layer fully connected neural network (i.e., 2NN) and logistics are trained models on MNIST and FEMNIST. The convolutional neural network (i.e., CNN) is selected to train models in CIFAR-10. Important parameters in the simulations are shown in Table II.

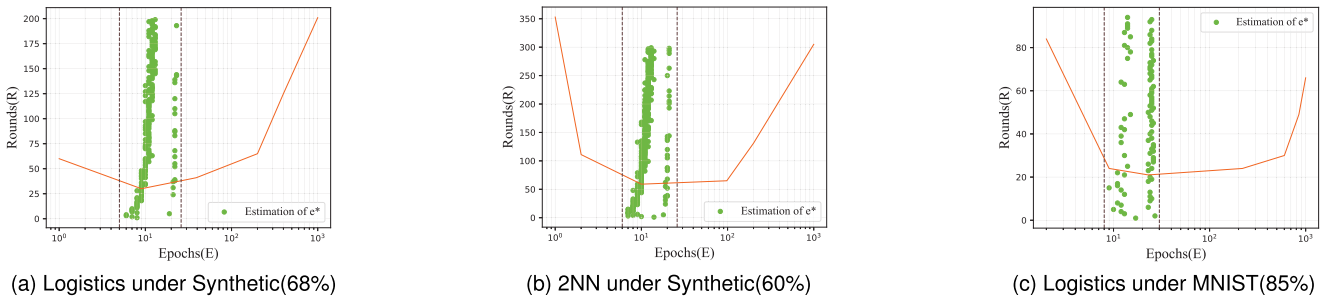
A. Existence of Optimal Solution

According to (21), the communication rounds R can be reduced by adjusting local training epochs to minimize the model degradation possibility due to packet loss. The results in Fig. 4 show the impact of epochs on accuracy during the training process. For example, with the increase of epochs from 5 to 20, the accuracy of the logistics model improves from 72% to 76% within 22 rounds, and the accuracy of 2NN improves from 54% to 58% during 30 rounds. However, increasing epochs from 20 to 50 in Fig. 4(a) and (b) just slightly improves less than 1% accuracy after the same rounds in both 2NN and logistics. In Fig. 4(c) and (d), we also observe the similar trends as those in Fig. 4(a) and (b), increasing epochs cannot linearly improve or even slow down the FL efficiency.



(a) The logistics model under MNIST (b) The 2NN model under Synthetic (c) The CNN model under CIFAR-10 (d) The 2NN model under FEMNIST

Fig. 4. Impact of increasing epochs on training efficiency.



(a) Logistics under Synthetic(68%) (b) 2NN under Synthetic(60%) (c) Logistics under MNIST(85%)

(d) 2NN under MNIST(80%) (e) Logistics under FEMNIST(80%) (f) CNN under CIFAR-10(80%)

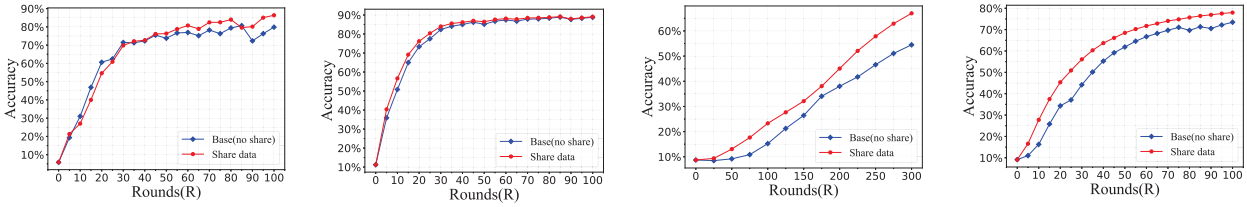
Fig. 5. The existence of optimal communication rounds.

In order to explicitly illustrate the impact of epochs on the model training, more simulations are conducted. Fig. 5 shows the rounds required to achieve a fixed accuracy with epochs increasing from 1 to 1000. We utilize 3 kinds of models and 4 kinds of datasets to verify the relation between epochs and communication rounds. The rounds to achieve a fixed accuracy in each sub-figure of Fig. 5 show the similar trend. With epochs increasing from 1 to 30, the training rounds achieving the required accuracy decrease, and the minimal communication rounds $R_{\min}$ are obtained. With the epochs gradually increasing to 1000, the training rounds achieving the required accuracy also go up.

In Fig. 5, green dots correspond to the estimation of epochs e^* by (33) during the training process. The estimations of e^* in Fig. 5(a) and (b) are mainly distributed in interval [8,13]. In Fig. 5(a), the rate of estimations distributed in interval [8,13] is 86%, and the rate is 85.5% in Fig. 5(b). There are 84% estimations distributed in [12,14] and [23,26] in Fig. 5(c). In Fig. 5(d), estimations distributed in intervals [8,9] and [18,19] are over 89%. The cause of such estimation distributions in Fig. 5(c) and (d) is the fluctuating δ per-round in the proposed algorithm. Due to a more unbalanced division of MNIST, δ fluctuates from 0.8×10^{-7} to 0.3, which leads to a shift of estimations. In Fig. 5(e), 87% epoch estimations distributed in the internals [9,21], and in Fig. 5(f) over 90% estimations are in intervals [10,37]. Therefore, the trends in Fig. 5 prove the existence of the optimal communication rounds in (27). Meanwhile, the above simulation results demonstrate that the estimation in (33) is accurate. By dynamically adjusting the epochs to estimate e^*, the communication rounds can be reduced in ACFL.

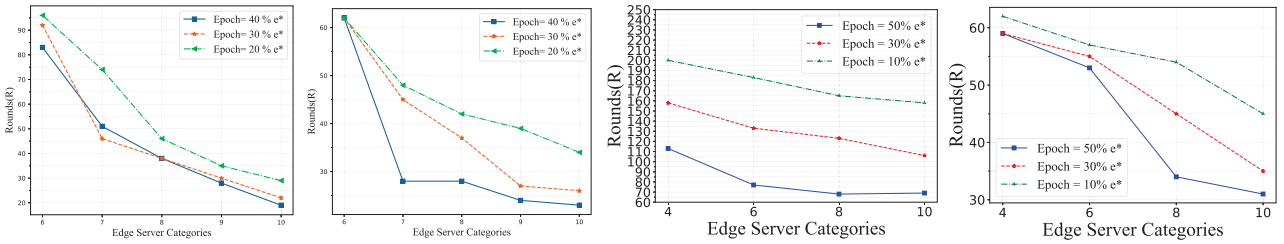
B. *Effectiveness of ES Co-Training Scheme With Auxiliary Dataset*

In Fig. 6, in order to verify the effectiveness of the ES co-training scheme with the auxiliary dataset in decreasing the communication rounds in ACFL, we conduct two simulations: two-layer ACFL (the adaptive federated learning algorithm with auxiliary dataset) and FedAvg in edge servers. The data in end devices and epochs executing in end devices are same as that in previous simulations, and six categories of data are constructed in edge servers. The results of logistics and 2NN show that training with the auxiliary dataset leads to higher accuracy with the same communication rounds. In Fig. 6, by



(a) 2NN model under MNIST (b) Logistics model under MNIST (c) 2NN model under FEMNIST (d) Logistics model under FEMNIST

Fig. 6. Effectiveness of auxiliary dataset on training efficiency.



(a) The 2NN model under MNIST(85%) (b) The logistics model under MNIST(85%) (c) The 2NN model under FEMNIST(75%) (d) The logistics model under FEMNIST(75%)

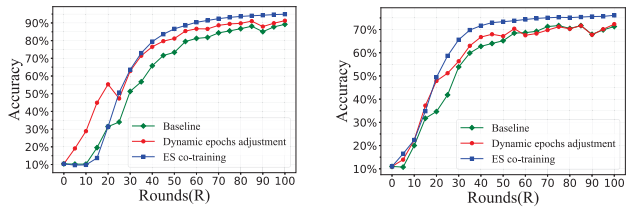
Fig. 7. Impact of data categories in edge servers and training epochs executing in edge servers.

utilizing the auxiliary dataset, the accuracy of logistics improves 5.74%, and the accuracy of 2NN improves 6.87% during 100 rounds. Furthermore, in the results of logistics and 2NN with FEMNIST as shown in Fig. 6(c) and (d), the accuracy of logistics improves 2.1% during 100 rounds, and the accuracy of 2NN improves 15.41% during 300 rounds. Observations in Fig. 6 confirm the result of (45) that training with the auxiliary dataset in edge servers is feasible to reduce the communication rounds, achieving a fixed accuracy.

Fig. 7 shows the impact of data distribution on the efficiency of the ES co-training scheme. We use various kinds of data in the auxiliary dataset to demonstrate the data distribution. For example, in Fig. 7, we randomly select 6 to 10 digits classes, i.e., data categories, from MNIST and FEMNIST to construct the auxiliary dataset and fix the size of the auxiliary dataset. Each curve in Fig. 7 shows the number of rounds to achieve a fixed accuracy with the increasing data types in the auxiliary dataset. With the data categories increasing from 6 to 10, the communication rounds are reduced by nearly 70 rounds.

In addition, Fig. 7 also shows the effect of epochs with the auxiliary dataset training on communication rounds. In the simulations on MNIST, we set α in Algorithm 3 to 20%, 30%, and 40%, respectively. In the simulations on FEMNIST, we set α to 10%, 30%, and 50% to clearly illustrate the impact of the auxiliary dataset. The curves in Fig. 7 show that increasing epochs with the auxiliary dataset reduces the communication rounds. Upon analyzing the curves in Fig. 7(c), we observe that with consistent data types in the auxiliary dataset, a 20% increase in the number of epochs can lead to a reduction in 50 communication rounds during the training process.

The simulations show that training on the auxiliary dataset with more global distribution or epochs can reduce the communication rounds to achieve the same accuracy. Such results prove



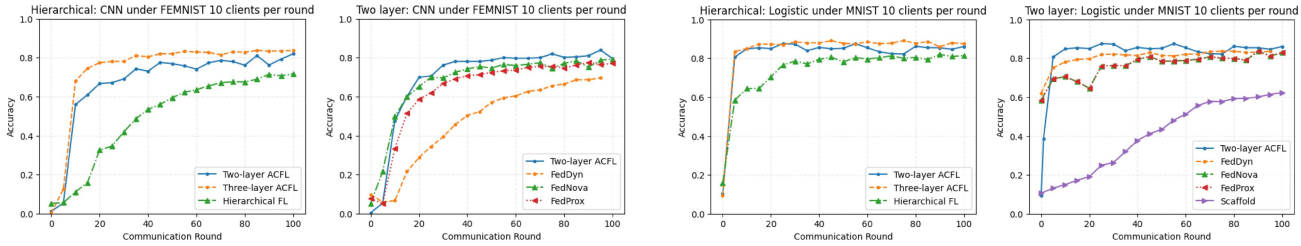
(a) The CNN model under FEMNIST (b) The logistics model under FEMNIST

Fig. 8. Comparisons between ES co-training scheme and dynamic epoch adjustment method.

the analysis of the auxiliary dataset and Theorem 1. Theoretically, the auxiliary dataset with a similar distribution to the global data distribution leads to a small $\delta^{(s)}$. Furthermore, increasing epochs executing on the auxiliary dataset can minimize $\rho(e)$ in Theorem 1. Thus, the ES co-training scheme in ACFL can lead to a tighter upper bound than the typical FL.

To further evaluate the performance of the ES co-training scheme and the performance of the dynamic epoch adjustment method on the accuracy, we respectively adopt them in Algorithm 3, and the original FedAvg is used as our baseline. Fig. 8 shows the simulations results of training CNN and logistics on the FEMNIST dataset. We set the epochs $e = 5$ to eliminate the impact of the dynamic epoch adjustment method in two-layer ACFL. Meanwhile, to maximize the effect of the ES co-training scheme, we use the whole FEMNIST as the auxiliary dataset in the edge server according to the results of Fig. 7. We evaluate the effect of dynamic epoch adjustment method without utilizing the ES co-training scheme.

For example, in Fig. 8(b), the dynamic epoch adjustment method achieves 70% accuracy 5 rounds earlier than the



(a) Performance comparison of different algorithms and architectures under FEMNSIT and CNN

(b) Performance comparison of different algorithms and architectures under MNSIT and Logistic

Fig. 9. Performance comparison of different algorithms and architectures.

baseline. After 100 rounds, the accuracy of simulations adopting the auxiliary dataset increased by 3.6% utilizing CNN and 2.7% utilizing logistics than the baseline. Both results in Fig. 8(a) and (b) show that dynamic epoch adjustment method achieves the same accuracy faster than the baseline. The dynamic epoch adjustment method can be considered an accelerated method to reduce the communication rounds. However, it cannot reduce the original gap between the optimal and sub-optimal solutions. As a result, it finally achieves the same accuracy as the baseline. On the other hand, the ES co-training scheme achieves higher accuracy than both the baseline and the dynamic epoch adjustment method. Hence, such improvement can reduce the gap between the optimal and sub-optimal solutions, according to (45).

C. Performance Comparison

We conduct simulations to evaluate the performance of the proposed ACFL on the FEMNIST and MNIST datasets, utilizing CNN and logistic models. To represent two-layer ACFL, we adopt the adaptive federated learning algorithm with auxiliary dataset in the two-layer FL architecture. We compare its performance with the state-of-the-art (SOTA) methods, including FedProx, FedNova, and FedDyn, on FEMNIST using CNN models. To further demonstrate the efficiency of ACFL under a three-layer architecture, we use a FedAvg-based hierarchical FL as the baseline. We evaluate the performance of the proposed scheme on FEMNIST using CNN models. Fig. 9(a) shows the results of our evaluations.

In addition to the SOTA methods demonstrated in Fig. 9(a), we also evaluate the performance of ACFL using the Scaffold method with a logistic model on MNIST. Using a FedAvg-based hierarchical FL as the baseline, we evaluate the performance of ACFL on MNIST with a logistic model. Fig. 9(b) shows the results of our evaluations on MNIST with a logistic model. Two edge servers are employed in the three-layer FL architecture for three-layer ACFL and the baseline. Considering the fairness of comparison, ten end devices are selected for all methods per round. In Fig. 9(a), the epochs on the auxiliary dataset are set to 10. In contrast, in Fig. 9(b), they are set to 5.

The simulation results demonstrate the effectiveness of the proposed ACFL scheme in reducing communication rounds. In terms of accuracy under the FEMNIST and MNIST datasets, using both CNN and logistic models, the proposed ACFL scheme is superior to the other comparison methods in both three-layer and two-layer architectures. For example, on the right side of Fig. 9(b), with the two-layer architecture, ACFL achieves 85% accuracy on MNIST within 15 rounds, whereas FedDyn, which achieves the best performance among other comparison methods, achieves only 83% in 30 rounds. Correspondingly, on the right side of Fig. 9(a), ACFL with 20 rounds produces 67% accuracy on the FEMNIST dataset, whereas FedProx, which achieves the best performance among other comparison methods, can only achieve 58%. In addition, the ACFL scheme shows superior performance in the three-layer architecture compared to the two-layer ACFL and the baseline. For instance, in Fig. 9(a), the three-layer ACFL has higher accuracy compared to the two-layer ACFL under the FEMNIST dataset and CNN model. Precisely, the three-layer ACFL acquires 70% accuracy in only 15 communication rounds compared with that the two-layer ACFL requires nearly 30 rounds to reach the same accuracy.

Additionally, the three-layer ACFL has higher accuracy than the two-layer ACFL even after 100 rounds. As depicted in Fig. 9(a), at 100th round, the three-layer ACFL achieves 83.6% accuracy, which is higher than the 82% accuracy of the two-layer ACFL. Such observations can be attributed to the fact that the three-layer ACFL consists of multiple edge servers that have auxiliary datasets. Therefore, the three-layer ACFL can reduce the data heterogeneity more efficiently than the two-layer ACFL, resulting in faster convergence. In conclusion, our experiments show the proposed ACFL can improve communication efficiency and alleviate data heterogeneity in federated learning.

VI. Conclusion

This paper presents an efficient federated learning scheme ACFL to improve training efficiency and alleviate the impact of the non-i.i.d. issue under the proposed hierarchical FL architecture. We have theoretically derived the optimal communication rounds and the corresponding epochs and proposed a dynamic epochs adjustment method to eliminate the probability of model parameter loss due to fragile wireless connections. Then, we have analyzed the feasibility of utilizing the auxiliary dataset to improve training efficiency and proposed the ES co-training scheme according to the theoretical analysis. Furthermore, based on the ES co-training scheme and dynamic epochs adjustment method, the adaptive federated learning algorithm with auxiliary

dataset has been designed to improve the communication efficiency and stability of ACFL. Finally, extensive simulations have been conducted, which show that ACFL can effectively reduce the communication rounds to achieve the required accuracy and improve training efficiency.

REFERENCES

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surv. Tut.*, vol. 17, no. 4, pp. 2347–2376, Fourth Quarter 2015.

[2] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, Jan. 2018.

[3] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[4] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," ETSI, Sophia Antipolis, France, White Paper, 2015.

[5] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," in *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.

[6] Z. Zhu, S. Wan, P. Fan, and K. B. Letaief, "Federated multi-agent actor-critic learning for age sensitive mobile edge computing," 2020, *arXiv:2012.14137*.

[7] R. Zeng, S. Zhang, J. Wang, and X. Chu, "FMore: An incentive scheme of multi-dimensional auction for federated learning in MEC," in *Proc. IEEE 40th Int. Conf. Distrib. Comput. Syst.*, Singapore, 2020, pp. 278–288.

[8] P. Kairouz et al., "Advances and open problems in federated learning," 2019, *arXiv:1912.04977*.

[9] J. Konečỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016, *arXiv:1610.05492*.

[10] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, *Federated Learning*. San Rafael, CA, USA: Morgan & Claypool, 2019.

[11] IEEE guide for architectural framework and application of federated machine learning, IEEE Standard 3652.1–2020, 2021.

[12] U. Majeed and C. S. Hong, "FLchain: Federated learning via MEC-enabled blockchain network," in *Proc. 20th Asia-Pacific Netw. Operations Manage. Symp.*, Matsue, Japan, 2019, pp. 1–4.

[13] Y. Zhao et al., "Privacy-preserving blockchain-based federated learning for IoT devices," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 1817–1829, Feb. 2021.

[14] T. Subramanya and R. Riggio, "Centralized and federated learning for predictive VNF autoscaling in multi-domain 5G networks and beyond," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 1, pp. 63–78, Mar. 2021.

[15] G. Zhu, Y. Du, D. Gündüz, and K. Huang, "One-bit over-the-air aggregation for communication-efficient federated edge learning," in *Proc. IEEE Glob. Commun. Conf.*, Taipei, Taiwan, China, 2020, pp. 1–6.

[16] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Wireless communications for collaborative federated learning," *IEEE Commun. Mag.*, vol. 58, no. 12, pp. 48–54, Dec. 2020.

[17] D. Li and J. Wang, "FedMD: Heterogenous federated learning via model distillation," 2019, *arXiv:1910.03581*.

[18] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *Proc. IEEE Int. Conf. Commun.*, Dublin, Ireland, 2020, pp. 1–6.

[19] K. Bonawitz et al., "Towards federated learning at scale: System design," 2019, *arXiv:1902.01046*.

[20] W. Y. B. Lim et al., "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surv. Tut.*, vol. 22, no. 3, pp. 2031–2063, Third Quarter 2020.

[21] F. O. Olowononi, D. B. Rawat, and C. Liu, "Federated learning with differential privacy for resilient vehicular cyber physical systems," in *Proc. IEEE 18th Annu. Consum. Commun. Netw. Conf.*, Las Vegas, NV, USA, 2021, pp. 1–5.

[22] Z. Xue et al., "A resource-constrained and privacy-preserving edge-computing-enabled clinical decision system: A federated reinforcement learning approach," *IEEE Internet Things J.*, vol. 8, no. 11, pp. 9122–9138, Jun. 2021.

[23] L. Feng, Z. Yang, S. Guo, X. Qiu, W. Li, and P. Yu, "Two-layered blockchain architecture for federated learning over mobile edge network," *IEEE Netw.*, vol. 36, no. 1, pp. 45–51, Jan./Feb. 2022.

[24] N. Roux, M. Schmidt, and F. Bach, "A stochastic gradient method with an exponential convergence rate for finite training sets," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, Lake Tahoe, NV, USA, 2012, pp. 2663–2671.

[25] F. Zhou and G. Cong, "On the convergence properties of a K-step averaging stochastic gradient descent algorithm for nonconvex optimization," 2017, *arXiv:1708.01012*.

[26] J. Konečnỳ, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," 2015, *arXiv:1511.03575*.

[27] O. Shamir, N. Srebro, and T. Zhang, "Communication-efficient distributed optimization using an approximate Newton-type method," in *Proc. Int. Conf. Mach. Learn.*, Beijing, China, 2014, pp. 1000–1008.

[28] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, Lake Tahoe, NV, USA, 2013, pp. 315–323.

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

[30] S. Wang et al., "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *Proc. IEEE Conf. Comput. Commun.*, Honolulu, HI, USA, 2018, pp. 63–71.

[31] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2018, *arXiv:1812.06127*.

[32] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," 2019, *arXiv:1907.02189*.

[33] D. A. E. Acar, Y. Zhao, R. M. Navarro, M. Mattina, P. N. Whatmough, and V. Saligrama, "Federated learning based on dynamic regularization," in *Proc. 9th Int. Conf. Learn. Representations*, 2021, pp. 1–36.

[34] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3400–3413, Sep. 2020.

[35] X. Wu, X. Yao, and C.-L. Wang, "FedSCR: Structure-based communication reduction for federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1565–1577, Jul. 2021.

[36] Y. Huang et al., "Personalized cross-silo federated learning on non-IID data," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 7865–7873.

[37] Z. Yue, L. Meng, L. Liangzhen, S. Naveen, C. Damon, and C. Vikas, "Federated learning with Non-IID data," 2018, *arXiv:1806.00582*.

[38] T.-C. Chiu, Y.-Y. Shih, A.-C. Pang, C.-S. Wang, W. Weng, and C.-T. Chou, "Semisupervised distributed learning with non-IID data for AIoT service platform," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9266–9277, Oct. 2020.

[39] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 5132–5143.

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

[41] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Rev.*, vol. 60, no. 2, pp. 223–311, 2018.

[42] Y. Nesterov, *Introductory Lectures on Convex Optimization: A. Basic Course*, vol. 87. Berlin, Germany: Springer, 2003.

[43] D. Liu, G. Zhu, J. Zhang, and K. Huang, "Data-importance aware user scheduling for communication-efficient edge machine learning," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 1, pp. 265–278, Mar. 2021.

[44] S. Huang, S. Wang, R. Wang, M. Wen, and K. Huang, "Reconfigurable intelligent surface assisted edge machine learning," in *Proc. IEEE Int. Conf. Commun.*, Montreal, QC, Canada, 2021, pp. 1–6.

[45] A. Albaseer, B. S. Ciftler, M. Abdallah, and A. Al-Fuqaha, "Exploiting unlabeled data in smart cities using federated edge learning," in *Proc. Int. Wireless Commun. Mobile Comput.*, Piscataway, NJ, USA, 2020, pp. 1666–1671.

[46] K. Guo, R. Gao, W. Xia, and T. Q. S. Quek, "Online learning based computation offloading in MEC systems with communication and computation dynamics," *IEEE Trans. Commun.*, vol. 69, no. 2, pp. 1147–1162, Feb. 2021.

[47] J. Liu and Q. Zhang, "To improve service reliability for AI-powered time-critical services using imperfect transmission in MEC: An experimental study," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9357–9371, Oct. 2020.

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

[49] S. Caldas et al., "Leaf: A benchmark for federated settings," 2018. *arXiv:1812.01097*.

[50] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," M.S. thesis, Univ. Tront, 2009.

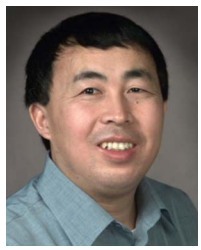
Tianao Xiang received the BE degree in software engineering from Northeastern University, Shenyang, China, in 2018. He is currently working toward the PhD degree in computer science with Northeastern University. His research interests include mobile edge computing, Internet of Vehicles and federated learning, etc.

Yuanguo Bi (Member, IEEE) received the PhD degree in computer science from Northeastern University, Shenyang, China, in 2010. He was a visiting PhD degree with the BroadBand Communications Research (BBCR) lab, Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada from 2007 to 2009. He is currently a professor with the School of Computer Science and Engineering, Northeastern University. He has authored/coauthored more than 50 journal/conference papers, including high quality journal papers, such as *IEEE Journal on Selected Areas in Communications*, *IEEE Transactions on Wireless Communications*, *IEEE Transactions on Intelligent Transportation Systems*, *IEEE Transactions on Vehicular Technology*, *IEEE IoT Journal*, *IEEE Communications Magazine*, *IEEE Wireless Communications*, IEEE Network, and Mainstream Conferences, such as IEEE Global Communications Conference, IEEE International Conference on Communications. His research interests include medium access control, QoS routing, multihop broadcast, and mobility management in vehicular networks, software-defined networking, and mobile edge computing. He has served as an editor/guest editor for IEEE Communications Magazine, IEEE Wireless Communications, IEEE ACCESS. He has also served as the Technical Program Committee member for many IEEE conferences.

Xiangyi Chen received the MS degree in computer science from Northeastern University, Shenyang, China, in 2019. She is currently working toward the PhD degree in computer science with Northeastern University. Her research interests include mobile edge computing, software-defined networking and network function virtualization, etc.

Yuan Liu received the BSc degree in the honor school, Harbin Institute of Technology, China, in 2010 and the PhD degree in the School of Computer Engineering from Nanyang Technological University (NTU), Singapore, in 2014. She is a professor with the Cyberspace Institute of Advanced Technology of Guangzhou University in Guangdong, China. She was an Associate Professor with northeastern university, China from 2015 to 2021. From 2014 to 2015, and she ever worked as research fellow with Joint NTU-UBC Research Center of Excellence in Active Living for the Elderly (LILY), NTU, Singapore. Her research interests include trust-based incentive mechanism design, federated learning, trust management, blockchain consensus protocols, and blockchain powered artificial intelligence.

Boyang Wang received the BS degree in computer science and technology from Northeast Petroleum University, Daqing, China, in 2019. She is currently working toward the MS degree in computer science with Northeastern University, Shenyang, China. Her research interests include Internet of Vehicles, federated learning and intrusion detection, etc.



Xuemin (Sherman) Shen (Fellow, IEEE) is an university professor and an associate chair for graduate studies with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research focuses on wireless resource management, wireless network security, social networks, smart grid, and vehicular ad hoc and sensor networks. He is the IEEE Communications Society President, was an elected member of IEEE ComSoc Board of Governor, and the chair of Distinguished Lecturers Selection Committee. He served as the technical program committee chair/co-chair for IEEE Globecom'16, Infocom'14, IEEE VTC'10 Fall, and Globecom'07, the Symposia chair for IEEE ICC'10, the tutorial chair for IEEE VTC'11 Spring and IEEE ICC'08, the general co-chair for ACM Mobihoc'15, and the chair for IEEE Communications Society Technical Committee on Wireless Communications, and P2P Communications and Networking. He also serves/served as an editor-in-chief for the IEEE Internet of Things Journal, and the IEEE Network, a founding area editor for the IEEE Transactions on Wireless Communications, and an associate editor for the IEEE Transactions on Vehicular Technology and the IEEE Wireless Communications, etc. He received the IEEE ComSoc Education Award, the Joseph LoCicero Award for Exemplary Service to Publications, the Excellent Graduate Supervision Award in 2006, and the Premiers Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada. He is a registered professional engineer of Ontario, Canada, a fellow of the IEEE, an Engineering Institute of Canada fellow, a Canadian Academy of Engineering fellow, a Royal Society of Canada fellow, and a distinguished Lecturer of IEEE Vehicular Technology Society and Communications Society.

Xingwei Wang received the BS, MS, and PhD degrees in computer science from Northeastern University, in 1989, 1992, and 1998, respectively. He is currently a professor with the School of Computer Science and Engineering, Northeastern University. His research interests include cloud computing and future Internet, among others. He has published more than 100 journal articles, books and book chapters, and refereed conference papers. He has received several best paper awards.