Communication-Efficient Federated Learning With Adaptive Aggregation for Heterogeneous Client-Edge-Cloud Network

Long Luo, *Member, IEEE*, Chi Zhang, *Student Member, IEEE*, Hongfang Yu, *Member, IEEE*, Gang Sun, *Senior Member, IEEE*, Shouxi Luo, *Member, IEEE*, and Schahram Dustdar, *Fellow, IEEE*

***Abstract*—Client-edge-cloud Federated Learning (CEC-FL) is emerging as an increasingly popular FL paradigm, alleviating the performance limitations of conventional cloud-centric Federated Learning (FL) by incorporating edge computing. However, improving training efficiency while retaining model convergence is not easy in CEC-FL. Although controlling aggregation frequency exhibits great promise in improving efficiency by reducing communication overhead, existing works still struggle to simultaneously achieve satisfactory training efficiency and model convergence performance in heterogeneous and dynamic environments. This paper proposes FedAda, a communication-efficient CEC-FL training method that aims to enhance training performance while ensuring model convergence through adaptive aggregation frequency adjustment. To this end, we theoretically analyze the model convergence under aggregation frequency control. Based on this analysis of the relationship between model convergence and aggregation frequencies, we propose an approximation algorithm to calculate aggregation frequencies, considering convergence and aligning with heterogeneous and dynamic node capabilities, ultimately achieving superior convergence accuracy and speed. Simulation results validate the effectiveness and efficiency of FedAda, demonstrating up to 4% improvement in test accuracy, 6.8× shorter training time and 3.3× less communication overhead compared to prior solutions.**

***Index Terms*—Aggregation frequency, client-edge-cloud, communication, federated learning, training efficiency.**

I. Introduction

FEDERATED Learning (FL) [1] has emerged as an innovative and promising approach for training AI models over massive data generated at the network edge, fostering the advancement of artificial intelligence (AI) systems [2]. Conventionally, FL employs a cloud-centric architecture, where geographically dispersed end devices serve as clients to collectively train AI models. Throughout the training process, clients iteratively update the model parameters using their local datasets and periodically send these updated parameters to a central cloud server. Subsequently, the cloud server consolidates these local updates to improve the global model. However, such a client-cloud FL (CC-FL) system encounters a communication bottleneck arising from constrained communication resources between the remote cloud server and distributed clients [3]. Empowered by edge computing, the recently proposed client-edge-cloud Federated Learning (CEC-FL) system [4] offers a promising solution to alleviate the communication bottleneck encountered by CC-FL. CEC-FL facilitates the aggregation of model updates from multiple clients to edge servers for local aggregation. Subsequently, a single copy of the aggregated result is transmitted to the cloud for global aggregation, effectively reducing communication traffic to the cloud.

However, it is not easy to enhance the training performance of the CEC-FL system by controlling the frequency of model parameter aggregation. For example, to facilitate model convergence, clients may need frequent updates [4], which in turn increases traffic and communication time, consequently slowing down the training speed. On the other hand, reducing aggregation frequency can decrease communication overhead during training, but it may also raise the risk of model overfitting and hence impact model quality. Moreover, the heterogeneity and dynamics of resources among different clients and edges will further exacerbate the complexity of controlling aggregation frequency.

Despite many studies have been proposed to optimize the aggregation frequency, most of them focus on CC-FL [5], [6], [7], [8], [9], [10], while only a few recent solutions are dedicated to CEC-FL [4], [11], [12]. HierFAVG [4] and HFL [11] pioneered the training method with reduced aggregation frequency for CEC-FL. They focus on homogeneous environments and investigate the impact of aggregation frequencies through experiments. Their results show that controlling aggregation frequencies can improve training performance, but the effect varies greatly under different frequencies. They do not provide calculation algorithms and cannot determine appropriate aggregation frequencies for heterogeneous environments. RAF [12], the state-of-the-art heterogeneity-aware solution, adaptively

Manuscript received 22 November 2023; revised 25 April 2024; accepted 2 May 2024. Date of publication 10 May 2024; date of current version 30 December 2024. This work was supported by National Natural Science Foundation of China under Grant 62102066, under Grant 62394324, and in part by the Young Elite Scientists Sponsorship Program by CAST under Grant 2022QNRC001. *(Corresponding author: Hongfang Yu.)*

Long Luo, Chi Zhang, Hongfang Yu, and Gang Sun are with the University of Electronic Science and Technology of China, Chengdu 610054, China (e-mail: llong@uestc.edu.cn; zhangcww@std.uestc.edu.cn; yuhf@uestc.edu.cn; gangsun@uestc.edu.cn).

Shouxi Luo is with Southwest Jiaotong University, Chengdu 610032, China (e-mail: sxluo@swjtu.edu.cn).

Schahram Dustdar is with Distributed Systems Group, TU Wien, 1040 Vienna, Austria (e-mail: dustdar@dsg.tuwien.ac.at).

Digital Object Identifier 10.1109/TSC.2024.3399649

controls aggregation frequencies and presents a heuristic to compute frequencies for different nodes. It aims to alleviate synchronization barriers under resource heterogeneity but lacks guarantees for model convergence. This absence of a solid theoretical foundation may fail to consistently ensure model convergence while achieving high training efficiency in CEC-FL.

In this paper, we propose FedAda, an efficient CEC-FL method that aims to improve training efficiency while maintaining high model quality through adaptive aggregation frequency control. To this end, we explore the joint optimization of local aggregation frequency at the edge and global aggregation frequency in the cloud. Optimizing local aggregation frequency to achieve high training efficiency with convergence guarantee is already challenging for CEC-FL system [11], even under homogeneous resource conditions. Joint optimization of local and global aggregation frequencies further increases the complexity of model training, particularly in networks with dynamic and heterogeneous resources. Although joint optimization has the potential to significantly improve communication efficiency, it should be done with great caution. For FL training that relies on synchronous update mechanisms [1], [4], improper aggregation frequencies for nodes with heterogenous resources can cause severe synchronization barriers when coordinating updates from multiple nodes [3], [12]. These barriers may introduce substantial latency, potentially offsetting or outweighing the communication efficiency gains, ultimately diminishing the overall training efficiency. Therefore, we first quantify the relationship between aggregation frequency and model convergence. Based on this, we compute benchmark aggregation frequencies for model quality and adjust them according to node computational and communication capabilities to improve training efficiency. In summary, this work mainly makes the following contributions:

- We propose FedAda, a novel method that jointly optimizes local and global aggregation frequencies in the client-edge-cloud hierarchical Federated Learning (CEC-FL) system. By optimizing these frequencies, FedAda effectively balances the computational workload of local training nodes, the communication between clients and edges, and the communication between edges and the cloud in resource-constrained dynamic heterogeneous networks.
- We provide a theoretical analysis of the convergence of CEC-FL training when employing optimized aggregation frequencies for efficiency improvement. This convergence analysis lays the foundation for the development of our algorithm.
- Motivated by the convergence analysis, we propose an approximate algorithm that calculates appropriate benchmark aggregation frequencies for model quality improvement. We further propose an online adaptive adjustment strategy to tune these benchmark frequencies according to resource heterogeneity for efficiency improvement.
- Extensive experiments validate the superiority of FedAda in terms of both training efficiency and model performance compared to state-of-the-art methods, with up to 4% improvement in accuracy, $6.8\times$ shorter training time, and $3.3\times$ less communication overhead compared to baselines.

Next, Section II discusses related works and their limitations. Section III presents the learning problem and aggregation frequency control in the CEC-FL system. Section IV formulates the adaptive frequency optimization problem for model training in CEC-FL. The theoretical analysis of convergence upper bound and an approximation algorithm are presented in Sections V and VI, respectively. Finally, Section VII evaluates the proposed algorithm, and Section VIII concludes this work.

II. Related Work

To improve the training efficiency of FL systems, researchers have recently studied the optimization of FL frameworks, communication, *etc*. We briefly review the existing works that are most relevant to this work.

The past few years have seen a lot of work on the communication optimization for the CC-FL framework (i.e., [13], [14], [15], [16]). A body of previous FL research focuses on traditional data compression methods and investigates the application of quantization or sparsification to reduce the data transmitted in each communication connection during training. For example, prior work [17] leverages quantization, using a limited number of bits to represent each model gradient or parameter that needs transmission, thus compressing data and reducing communication overheads. While previous work [18] uses sparsification to decrease communication costs by selectively transmitting a portion of gradients or model parameters.

In addition to data compression techniques, aggregation frequency control is emerging as an effective solution to reduce communication overhead. This approach optimizes how often participating nodes communicate for model aggregation. Straightforwardly, it is feasible and effective to mitigate communication costs by decreasing aggregation frequency to cut down the rounds of communication among nodes [19]. The work [1] demonstrates the possibility of reducing aggregation frequency and verifies the performance improvement over full-batch stochastic gradient descent [20] through numerous experiments. However, this work lacks theoretical proof of the relationship between model convergence and the aggregation frequency, a gap addressed by [21]. Fang et al. [5] argue that it should be careful to reduce the aggregation frequency as decreased communication may impact training performance. Many subsequent studies aim to strike a trade-off between training efficiency and model quality [6]. For example, Malinovsky et al. [6] attempt to trade off model convergence, and communication costs with aggregation frequency control. Luo et al. [7] consider resource limitations in real networks and explore adaptive aggregation frequency control for clients under such constraints. To further enhance training efficiency, recent works such as [8] and [9] propose integrating compression techniques with aggregation frequency control. Liu et al. [10] go a step further by proposing adaptive control of aggregation frequency and batch size, exploiting their interactions to trade off convergence, cost, and training completion time in dynamic environments.

Unlike CC-FL, which only controls aggregation frequency at the cloud, CEC-FL needs to manage two types of aggregation

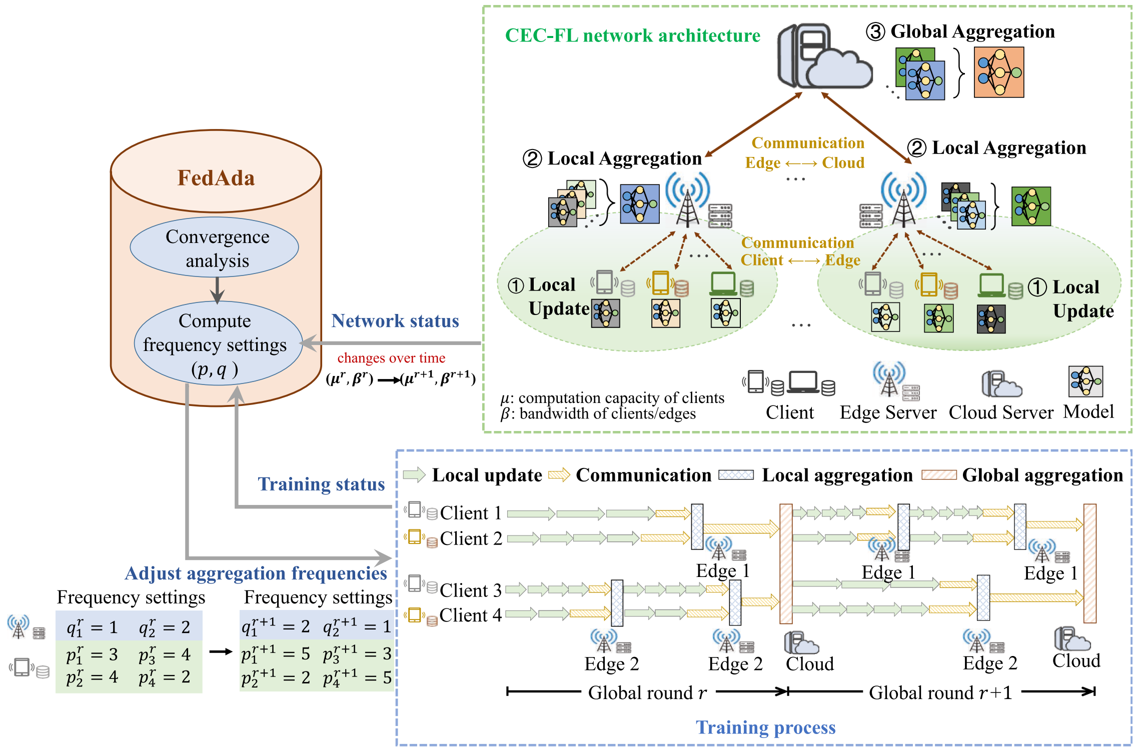


Fig. 1. Illustration of CEC-FL network architecture and training process with FedAda.

frequencies, local at the edge and global at the cloud [22]. The CEC-FL system leverages edge computing by introducing edge servers as local aggregators in addition to the cloud-based global aggregator, thereby implementing a two-tier aggregation mechanism. Model updates from multiple clients undergo local aggregation at the edge server, with only the aggregated result transmitted from the edge to the cloud. This significantly diminishes the traffic load on communication links to the cloud. Liu et al. [4] were the first to propose controlling both local and global aggregation frequencies, investigating training performance under various frequency settings through extensive experiments. Wang et al. [11] theoretically demonstrate the benefits of local aggregation in accelerating model convergence. However, none of the aforementioned approaches address the challenge posed by dynamic heterogeneous networks. Recently, the RAF [12] proposes to adopt a weak synchronization mechanism [23] to mitigate the straggler problem caused by dynamic heterogeneous networks. It also designs a heuristic to calculate the aggregation frequencies for different nodes, however, it lacks convergence guarantees. Wu et al. [24] propose an approach that combines synchronous local aggregation and asynchronous global aggregation. While it decreases communication overhead, it also notably complicates the algorithm [25]. The authors of [26] suggest allowing edge servers with training data to participate in the model training with clients. This approach solely focuses on optimizing local aggregation frequency.

In summary, previous research on aggregation frequency optimization primarily focuses on CC-FL systems. Applying these approaches to CEC-FL systems would fall short in training performance. While some studies exist on aggregation frequency control for CEC-FL, they often concentrate on optimizing one aspect or struggle to adapt to dynamic heterogeneous networks, lacking convergence guarantees. Thus, a gap remains in understanding how to optimize aggregation frequency while ensuring convergence and improving training performance for CEC-FL, especially in heterogeneous and dynamic network scenarios.

III. Overview of CEC-FL System

In this section, we present the overall architecture of FedAda, including the learning problem, training process, and aggregation frequency control.

A. Learning Problem

We consider a typical cloud-edge-end network as shown in Fig. 1, which consists of N distributed end devices, M edge servers, and a cloud server. In this system, all edge servers are interconnected with the cloud server. Each edge server i ($i = 1, 2, \ldots, M$) establishes connections with a distinct group of end devices, denoted as $\mathcal{A}_i$, where the number of end devices in this group is represented by $N_i = |\mathcal{A}_i|$. Consequently, the total number of end devices in this system is given by $N = \sum_{i=1}^{M} N_i$.

The parameter-server (PS) architecture is employed for model training in this cloud-edge-end network to form a client-edge-cloud Federated Learning system. End devices act as clients, and each client j keeps a local dataset $\mathcal{D}_j$ and trains a local model $\boldsymbol{\omega}_j \in \mathbb{R}^d$ over this dataset, where $\boldsymbol{\omega}$ is the model vector and d is the dimension size. Edge servers act as local parameter servers, aggregating local model updates from clients connected to them and maintaining aggregated local models $\boldsymbol{\omega}_i \in \mathbb{R}^d$. Cloud server acts as the centralized PS, globally aggregating model updates from all edge servers and maintaining the global model $\boldsymbol{\omega} \in \mathbb{R}^d$.

The goal of CEC-FL is to obtain an optimal global model $\boldsymbol{\omega}^*$ that minimizes the global loss function $f(\boldsymbol{\omega})$ after a series of global training rounds, which can be expressed as

$$\min_{\boldsymbol{\omega}\in\mathbb{R}^d} f(\boldsymbol{\omega}) \triangleq \frac{1}{N}\sum_{i=1}^{M} N_i f_i(\boldsymbol{\omega}_i)$$
$$\boldsymbol{\omega}^* := \arg\min_{\boldsymbol{\omega}\in\mathbb{R}^d} f(\boldsymbol{\omega}) \tag{1}$$

where $f_i(\boldsymbol{\omega}_i)$ is the loss function corresponding to the i-th edge server, which is defined as the average loss function of clients in $\mathcal{A}_i$ as (2) shows. $F_j(\boldsymbol{\omega}_j)$ is the loss function of the j-th client in $\mathcal{A}_i$.

$$f_i(\boldsymbol{\omega}_i) \triangleq \frac{1}{N_i}\sum_{j\in\mathcal{A}_i} F_j(\boldsymbol{\omega}_j) \tag{2}$$

B. Training Process

Consistent with many previous FL works, we adopt the gradient descent algorithm [27] as the solution to address (1). Each client employs the mini-batch stochastic gradient descent (SGD) for each local iteration of model training. Specifically, in each iteration, the client takes a mini-batch randomly sampled from its local dataset to update the local model. After several local iterations, each client pushes the learned model parameters to the associated edge server. The edge server aggregates the models from all its connected clients to update the aggregated model and distributes it to clients for the next local round of model training. Periodically, after certain local aggregation rounds are completed, edge servers further transmit their aggregated models to the cloud server for aggregation into an improved global model. The cloud then broadcasts the updated global model back to the edges, and each edge distributes the latest global model to its connected clients for the next global round of training. The above local training at clients, periodic local aggregation at edges, and global aggregation at the cloud constitute the process of one global training round considered in this work. This process is iteratively repeated until the global model converges or reaches resource limits. In short, the key steps in each global round are: local model updates by clients, local aggregation by edge servers, and global aggregation by the cloud.

Local update by clients: Let $\boldsymbol{\omega}_j^{(s,r)}(t)$ denotes the local model of client j at the t-th local iteration in the s-th local aggregation round of global round r. The model update in the local training iteration of client j using mini-batch SGD [28] can be expressed by (3).

$$\boldsymbol{\omega}_j^{(s,r)}(t+1) = \boldsymbol{\omega}_j^{(s,r)}(t) - \eta\nabla F_j\left(\boldsymbol{\omega}_j^{(s,r)}(t), \xi_j^{(s,r)}(t)\right) \tag{3}$$

where $\nabla F_j(\boldsymbol{\omega}_j^{(s,r)}(t), \xi_j^{(s,r)}(t))$ is an unbiased gradient estimation and η is the learning rate. $\xi_j^{(s,r)}(t)$ represents the mini-batch randomly sampled from the local dataset $\mathcal{D}_j$.

Local aggregation by edges: For every edge server, it performs a local aggregation after receiving the updated models from all required clients. As mentioned before, each client uploads its learned model to the corresponding edge server after a given number of local iterations. Specifically, we use local aggregation frequency p_j^r to control the number of local iterations of client j in each global round r. Therefore, the model $\boldsymbol{\omega}_i^{(r)}(s)$ updated at the end of s-th local aggregation within global round r at edge server i, which can be expressed by (4).

$$\boldsymbol{\omega}_i^{(r)}(s) = \frac{1}{N_i}\sum_{j\in\mathcal{A}_i}\alpha_j^r \boldsymbol{\omega}_j^{(s,r)}(p_j^r) \tag{4}$$

where α_j^r is the local aggregation weight among clients and $\sum_{j\in\mathcal{A}_i}\frac{\alpha_j^r}{N_i} = 1$. We can see $\boldsymbol{\omega}_j^{(s,r)}(0) = \boldsymbol{\omega}_i^{(r)}(s-1), \forall j \in \mathcal{A}_i$.

Global aggregation by the cloud: While for the cloud server, it performs a global aggregation after receiving the model updates from all required edge servers. Similarly, we adopt global aggregation frequency q_i^r to control the number of local aggregations performed by edge server i in global round r. By the end of the r-th global round, the cloud server follows (5) to update the global model $\boldsymbol{\omega}^{(r)}$.

$$\boldsymbol{\omega}^{(r)} = \frac{1}{M}\sum_{i=1}^{M}\alpha_i^r \boldsymbol{\omega}_i^{(r)}(q_i^r) \tag{5}$$

where α_i^r is the weight for global aggregation among edge servers and $\sum_{i=1}^{M}\frac{\alpha_i^r}{M} = 1$. Similarly, $\boldsymbol{\omega}_i^{(r)}(0) = \boldsymbol{\omega}^{(r-1)}, \forall i = 1,\ldots,M$. Some key notations are listed in Table I.

TABLE I
KEY NOTATIONS USED IN PROBLEM FORMULATION

Notation	Description
N/M	number of clients/edge servers
R	total number of global rounds
$\boldsymbol{\omega}$	global model
$\boldsymbol{\omega}_i$	aggregated model of edge server i
$\boldsymbol{\omega}_j$	local model of client j
$f(\boldsymbol{\omega})$	global loss function
$f_i(\boldsymbol{\omega}_i)$	aggregated loss function of edge server i
$F_j(\boldsymbol{\omega}_j)$	local loss function of client j
t^r	total training time until r-th round
$t_{edge,i}^r$	completion time of edge server i
$t_{client,j}^r(i)$	completion time of client j associated with edge i
W_{edge}^r	average waiting time of edge servers at round r
W_{client}^r	average waiting time of clients at round r
p_j^r	local aggregation frequency of client j
q_i^r	global aggregation frequency of edge server i
$\mu_{client,j}^r$	local computation time of client j at round r
$\beta_{client,j}^r$	communication time of client j at round r
$\beta_{edge,i}^h$	communication time of edge server i at round r
c_j^r	computational resource overhead of client j
b_j^r	communication resource overhead of client j
b_i^r	communication resource overhead of edge server i
C / B	computation / communication resource budget

C. Aggregation Frequency Control: Motivation and Basic Idea

Motivation: Considering the heterogeneity and dynamics of the limited node capacities, we should carefully control the aggregation frequency to improve the training efficiency and performance for CEC-FL. First, clients may have heterogeneous computational and communication capabilities. If all clients are required to perform the same round of local iterations before each local aggregation, this will cause powerful clients to wait for weak clients, thereby introducing non-negligible or even unacceptable latency. A similar synchronization-blocking problem also arises during global aggregation if all the edges are asked to undergo the same round of local aggregations before each global aggregation. In addition, an FL training task usually lasts for a long time, e.g., several hours or even days [29], during which resources may change greatly. It is not suitable to use fixed aggregation frequencies throughout the training process.

Basic idea: In a nutshell, we propose FedAda to adaptively control aggregation frequencies for participating nodes to address the above challenges in CEC-FL. The core idea is to adjust the aggregation frequency based on model training and network status, as shown in Fig. 1. Local aggregation frequencies (p_j^r) and global aggregation frequencies (q_i^r) can vary across nodes and rounds. As a tie-in to such frequency control approach, FedAda uses weak synchronization mechanism [12], [23] for model aggregation and update. The right bottom of Fig. 1 illustrates the training process using adaptive aggregation frequency control in a toy CEC-FL system with one cloud server, two edge servers (1 and 2), and four clients (1 to 4). In the r-th global round, clients perform varying local iterations (controlled by $p_j^r, j = 1, \ldots, 4$) before pushing updates to edges for local aggregation. Edges perform different rounds of local aggregation (controlled by $q_i^r, i = 1, 2$) before pushing models to the cloud for global aggregation. FedAda may change local and global aggregation frequencies for the next round as the algorithm detailed in Section VI.

IV. Optimizing Aggregation Frequency for CEC-FL: Problem Formulation

In this section, we answer the question of how to determine the appropriate aggregation frequencies for participating nodes in CEC-FL. As local and global aggregation frequencies jointly impact overall model training performance, we propose optimizing both frequencies together to minimize training time. Next, we present the total training time calculation and formulate the time minimization problem under joint frequency optimization in a resource-constrained, dynamic heterogeneous network.

A. Training Time Model

To attain model convergence, a CEC-FL task essentially involves multiple rounds of training, where the duration of training depends on the number of global rounds and the time taken to complete each round. Considering a CEC-FL task with a total number of R global rounds, where the time required to complete the r-th round is denoted as t^r, then the overall training time can be calculated by $T = \sum_{r=1}^{R} t^r$. Now, let's elaborate on the computation of t^r. As previously mentioned (see §III-B), the cloud server cannot globally update the model until receiving local aggregated models from all edge servers in each global round. In this synchronous training manner, t^r is primarily determined by the processing time of the slowest edge server. Let t_i^r denote the round time of edge server i and μ_{cloud}^r denote the computation time for the cloud server performing model aggregation in global round r, t^r can be expressed as follows.

$$t^r = \max_{i=1,\ldots,M} \{t_i^r\} + \mu_{cloud}^r \tag{6}$$

$$t_i^r = \underbrace{q_i^r \times t_{edge,i}^r}_{\text{local aggregation time}} + \underbrace{\beta_{edge,i}^r}_{\text{communication time}} \tag{7}$$

In addition to the communication time $\beta_{edge,i}^r$ required to upload the aggregated model to the cloud server, t_i^r also includes the local aggregation time for gathering the local updated models from all required clients in $\mathcal{A}_i$, as shown in (7). The local aggregation time of edge server i in global round r is the product of the global frequency q_i^r and the completion time $t_{edge,i}^r$ of a local aggregation round.

For local aggregation performed by the edge servers, we still consider a synchronous manner. Therefore, the local aggregation time of an edge server depends on the slowest client associated with it. Let $t_{client,j}^r$ denote the completion time of client j in each local aggregation round and $\mu_{edge,i}^r$ denote the computation time for the edge server i performing local aggregation, $t_{edge,i}^r$ can be computed by (8). The time $t_{client,j}^r$ of client j in each local round depends on the computation time for local training and the communication time $\beta_{client,j}^r$ of pushing the trained local model to the corresponding edge server for local aggregation. Note that as aforementioned, nodes may also transmit their resources or training status for frequency computation. We omit such transmission time because the network status information is only a few bytes, which is negligible compared to models with millions or even tens of millions of bytes. The computation time for local training is the product of the local aggregation frequency p_j^r and the computation time $\mu_{client,j}^r$ for a single local iteration, which can be denoted as (9).

$$t_{edge,i}^r = \max_{j \in \mathcal{A}_i} \left\{t_{client,j}^r\right\} + \mu_{edge,i}^r \tag{8}$$

$$t_{client,j}^r = \underbrace{p_j^r \times \mu_{client,j}^r}_{\text{local training time}} + \underbrace{\beta_{client,j}^r}_{\text{communication time}} \tag{9}$$

To simplify matters, we make two simplifications to the above time computation models as many prior works (i.e., [30], [31], [32]) in this field. Firstly, considering the downstream bandwidth is consistently much higher than the upstream bandwidth in the CEC-FL network [33], we can safely exclude the communication time required for downloading models, as it is not the performance bottleneck. Additionally, we can also exclude the computation times $\mu_{edge,i}^r$ and μ_{cloud}^r associated with model aggregation on both the edge servers and the cloud server, in line with the common practice of existing works [7], [12], [33]. The reason is that the computation time required for updating or aggregating a model on a powerful server is negligible compared to the time involved in training the model on clients with weak capabilities. Formally, the completion time t^r of each global round at the cloud server and the completion time $t_{edge,i}^r$ of each local aggregation round at the edge server i can be rewritten as

(10) and (11), respectively.

$$t^r \geq q_i^r \times t_{edge,i}^r + \beta_{edge,i}^r, \forall i = 1, \ldots, M \tag{10}$$

$$t_{edge,i}^r \geq p_j^r \times \mu_{client,j}^r + \beta_{client,j}^r, \forall j \in \mathcal{A}_i \tag{11}$$

B. Mathematical Formulation

With the above time computation models, we formulate the problem of minimizing the total CEC-FL training time under several necessary practical constraints. To improve the training performance and efficiency, our key idea is to adaptively control the aggregation frequency for both clients and edge servers during the training process to accelerate the model convergence.

Decision Variables: To deal with resource heterogeneity, we allow clients and edge servers to use different local and global aggregation frequencies in each global round. Furthermore, to improve training performance under changing training process and network status, we employ adaptive control that allows participating nodes to adopt different frequencies across various global rounds. To capture these decisions, we introduce p_j^r and q_i^r to denote the local aggregation frequency of client j and global aggregation frequency of edge server i in global round r, respectively.

Convergence Constraint: Model convergence is the primary performance requirement in a CEC-FL task. Let $f(\boldsymbol{\omega}^R)$ and $f(\boldsymbol{\omega}^*)$ denote the global loss function after R global rounds and the optimal global loss function, respectively. To ensure model convergence, it is desirable for $f(\boldsymbol{\omega}^R)$ to approximate $f(\boldsymbol{\omega}^*)$. This can be expressed as (12), where ϵ is a threshold to measure the gap.

$$f(\boldsymbol{\omega}^R) - f(\boldsymbol{\omega}^*) < \epsilon \tag{12}$$

Average Waiting Time Constraint: Considering the discrepancies in computational and communication capabilities between nodes, the presence of a synchronization barrier will have nodes with superior capabilities wait for those with inferior capabilities. To quantify this, we measure the average waiting time of edge servers, denoted as W_{edge}^r, and such that of clients, denoted as W_{client}^r. Moreover, we introduce thresholds ε_e and ε_c to limit the average waiting time to as small as possible. The waiting time of edge server i can be expressed as $(t^r - t_i^r)$ and that of client j can be denoted as $(t_{edge,i}^r - t_{client,j}^r)$. Therefore, the constraints on average waiting time can be formulated as

$$W_{edge}^r = \frac{1}{M}\sum_{i=1}^{M}(t^r - t_i^r) \leq \varepsilon_e \tag{13}$$

$$W_{client}^r = \frac{1}{N_i}\sum_{j\in\mathcal{A}_i}\left(t_{edge,i}^r - t_{client,j}^r\right) \leq \varepsilon_c \tag{14}$$

Objective: To enhance the training efficiency of CEC-FL, we aim to minimize the total time to complete the entire training process as shown in (15). With the above decision variables and constraints, we can formally formulate the problem of minimizing the total time by implementing adaptive control of local and global aggregation under a resource-constrained CEC-FL system, as shown in **P1**.

$$\textbf{P1} \qquad \min \sum_{r=1}^{R} t^r \tag{15}$$

$$s.t. \quad (10) - (14)$$

$$\sum_{r=1}^{R}\sum_{i=1}^{M} q_i^r \times \sum_{j\in\mathcal{A}_i} p_j^r \times c_j^r \leq C \tag{16}$$

$$\sum_{r=1}^{R}\sum_{i=1}^{M}\left(b_i^r + \sum_{j\in\mathcal{A}_i} q_i^r \times b_j^r\right) \leq B$$

$$p_j^r \in \mathbb{N}^+, q_i^r \in \mathbb{N}^+,$$
$$\forall r = \{1,\ldots,R\}, i = \{1,\ldots,M\}, j = \{1,\ldots,N\} \tag{17}$$

Eqs. (10)–(14) are the constraints relating to convergence and waiting time. Besides, the computational and communication resource constraints should not be violated, denoted by (16) and (17), which imply that the computational and communication overheads during the training process cannot exceed the system upper bounds C and B. We adopt a summation approach to define the resource constraints [8] and consider the case where node resources are heterogeneous. To capture client heterogeneity, we introduce parameters c_j^r and b_j^r to express the computational overhead of one local iteration and the communication overhead of model transmission on client j at global round r, respectively. We also consider edge heterogeneity and introduce parameter b_i^r to express the communication overhead of transmitting a full model from edge server i to the cloud.

V. Convergence Analysis

In this section, we analyze the convergence upper bound of the mean square gradient after R global rounds, with the following three assumptions [34].

Assumption 1 (Lipschitz Continuous Gradient): There exists a constant $L > 0$, such that:

$$\|\nabla F(\boldsymbol{\omega}_1) - \nabla F(\boldsymbol{\omega}_2)\| \leq L\|\boldsymbol{\omega}_1 - \boldsymbol{\omega}_2\|, \ \forall \boldsymbol{\omega}_1, \boldsymbol{\omega}_2 \tag{18}$$

Assumption 2 (Unbiased Estimated Gradient): Let ξ be a random sample from local dataset $\mathcal{D}$, then the estimated local gradient is unbiased:

$$\mathbb{E}\left[\nabla F(\boldsymbol{\omega};\xi)\right] = \nabla F(\boldsymbol{\omega}) \tag{19}$$

Assumption 3 (Bounded Variance): There exists a constant σ, such that the variance of the estimated local gradient can be bounded by:

$$\mathbb{E}\left[\|\nabla F(\boldsymbol{\omega};\xi) - \nabla F(\boldsymbol{\omega})\|^2\right] \leq \sigma^2 \tag{20}$$

To obtain the convergence upper bound as described in Theorem 1, we present three important lemmas as follows.

Lemma 1: According to Assumption 1, the expected inner product between full batch gradient and stochastic gradient can be bounded with:

$$\mathbb{E} < \nabla f(\boldsymbol{\omega}^r), g^r >$$

$$\leq \frac{L^2}{2N}\sum_{i=1}^{M}\alpha_i^r\sum_{s=1}^{q}\sum_{j\in\mathcal{A}_i}\alpha_j^r\sum_{t=1}^{p}\|\boldsymbol{\omega}^r-\boldsymbol{\omega}_j^{t,s,r}\|^2$$
$$-\frac{1}{2N}\sum_{i=1}^{M}\alpha_i^r\sum_{s=1}^{q}\sum_{j\in\mathcal{A}_i}\alpha_j^r\sum_{t=1}^{p}\|\nabla f\left(\boldsymbol{\omega}^r\right)^2\|^2$$
$$-\frac{1}{2N}\sum_{i=1}^{M}\alpha_i^r\sum_{s=1}^{q}\sum_{j\in\mathcal{A}_i}\alpha_j^r\sum_{t=1}^{p}\|\nabla F_j\left(\boldsymbol{\omega}_j^{t,s,r}\right)\|^2 \quad (21)$$

where $g^r=\frac{1}{N}\sum_{i=1}^{M}\alpha_i^r\sum_{s=1}^{q}\sum_{j\in\mathcal{A}_i}\alpha_j^r\sum_{t=1}^{p}\nabla F_j(\boldsymbol{\omega}_j^{t,s,r})$.

Lemma 2: Under Assumptions 2 and 3, we have the following bound:

$$\mathbb{E}\|g^r\|^2\leq\frac{1}{N^2}\sum_{i=1}^{M}(\alpha_i^r)^2\sum_{s=1}^{q}\sum_{j\in\mathcal{A}_i}\left(\alpha_j^r\right)^2\sum_{t=1}^{p}\sigma^2$$
$$+\frac{p}{N^2}\sum_{i=1}^{M}(\alpha_i^r)^2\sum_{s=1}^{q}\sum_{j\in\mathcal{A}_i}\left(\alpha_j^r\right)^2\sum_{t=1}^{p}\|\nabla F_j\left(\boldsymbol{\omega}_j^{t,s,r}\right)\|^2 \quad (22)$$

Lemma 3: Under Assumptions 3 and according to [11], we have the following bound:

$$\mathbb{E}\|\boldsymbol{\omega}^r-\boldsymbol{\omega}_j^{t,s,r}\|^2\leq\eta^2\left[\left(\frac{q-1}{Nq}\right)^2+\frac{(p-1)^2}{Np^2q^2}\right]$$
$$\cdot\sum_{i=1}^{M}\sum_{s=1}^{q}\sum_{j\in\mathcal{A}_i}\sum_{t=1}^{p}\left[p\|\nabla F_j\left(\boldsymbol{\omega}_j^{t,s,r}\right)\|^2+\sigma^2\right] \quad (23)$$

Based on the above assumptions and lemmas, we can obtain the convergence upper bound of the mean square gradient as described in Theorem 1.

Theorem 1: Assuming that all nodes are initialized to the same model parameters $\boldsymbol{\omega}^0$, then the upper bound of the mean square gradient after R global rounds can be expressed as:

$$\frac{1}{R}\sum_{r=1}^{R}\|\nabla f\left(\boldsymbol{\omega}^r\right)\|^2\leq\frac{2\left[f\left(\boldsymbol{\omega}^0\right)-f\left(\boldsymbol{\omega}^*\right)\right]}{\eta\alpha_1\alpha_2pqR}+\frac{L\eta\alpha_1\alpha_2}{N}\sigma^2$$
$$+\frac{L^2\eta^2pq}{N}\sigma^2+\frac{L^2\eta^2p}{q}\sigma^2 \quad (24)$$

where $p=\max\{p_j^r\},\ q=\max\{q_i^r\},\ \alpha_1=\max\{\alpha_j^h\},\ \alpha_2=\max\{\alpha_i^h\}$ and the learning rate η satisfy:

$$L^2\eta^2\left[\frac{p^2\left(q-1\right)^2}{Nq}+\frac{(p-1)^2}{q}\right]+\frac{L\eta\alpha_1\alpha_2p}{N}\leq 1 \quad (25)$$

For the proof of Theorem 1, please refer to https://github.com/LesLieZC0324/FedAda. The core idea of FedAda is to adaptively adjust the local and global aggregation frequency for both clients and edge servers. Therefore, according to Theorem 1, we analyze the relationship between the convergence upper bound and the above two frequencies as shown in Theorem 2.

Theorem 2: Let the learning rate $\eta=c_1\frac{q}{\sqrt{RpL}}$, local aggregation weight $\alpha_1=c_2\sqrt{pN/M}$ and global aggregation weight $\alpha_2=c_3\sqrt{qM}$, where c_1, c_2 and c_3 are all constants, then the convergence upper bound can be transformed as:

$$\frac{1}{R}\sum_{r=1}^{R}\|\nabla f\left(\boldsymbol{\omega}^r\right)\|^2\leq\frac{2L\left[f\left(\boldsymbol{\omega}^0\right)-f\left(\boldsymbol{\omega}^*\right)\right]}{c_1c_2c_3q^2\sqrt{pqRN}}$$
$$+\frac{c_1c_2c_3\sqrt{q^3}}{\sqrt{RNp}}\sigma^2+\frac{c_1^2q^3}{RNp}\sigma^2+\frac{c_1^2q}{Rp}\sigma^2 \quad (26)$$

With Theorem 2, FedAda can achieve a linear speedup of convergence rate $\mathcal{O}(1/q\sqrt{pRN})$.

VI. Computing Local and Global Aggregation Frequencies: Algorithm Design

In this section, we propose an adaptive control algorithm to solve problem **P1**, which computes and adapts local and global aggregation frequencies for clients and edge servers with model convergence and resource heterogeneity in mind. Based on this, we give the corresponding procedures at the client, edge server and cloud server sides to show how they collaborate during the training process.

A. Heterogeneity-Aware Computation Strategy

Approximation to P1: It is not hard to see that **P1** is NP-hard and difficult to solve in polynomial time. Thus, we design an approximate algorithm to compute the solution, namely the aggregation frequencies of clients and edge servers per global round, which can improve the training quality while guaranteeing convergence.

According to the convergence analysis in Section V, the training model can eventually converge if the right side of (26) is always smaller than a small constant ρ. Given the learning rate $\eta=c_1\frac{q}{\sqrt{RpL}}$, we can use $q=up$ to express the relationship between the aggregation frequencies of the clients and the edge servers, where $u=\frac{\eta L\sqrt{R}}{c_1}$. Thus, the right side of (26) can be rewritten as

$$\Phi(R,p,u)=\frac{2L\cdot f(\boldsymbol{\omega}^0)}{u^2p^3\sqrt{uRN}}$$
$$+\left(\frac{\sqrt{u^3p}}{\sqrt{RN}}+\frac{u^3p^2}{RN}+\frac{u}{R}\right)\sigma^2\leq\rho \quad (27)$$

where ρ contains c_1, c_2, c_3 and tends to 0. Now, we need to ensure that the value of $\Phi(R,p,u)$ remains consistently smaller than a small constant ρ to ensure convergence. To this end, in each global round, we first compute the optimal values of p^* and u^* that can minimize $\Phi(R,p,u)$ according to the model training information that evolves as the training process. Then, we calculate $q^*=u^*p^*$ accordingly. Theoretically, we can let all clients and edge servers respectively use the optimal results of p^* and q^* as their frequencies during the training process. However, in a heterogeneous scenario, there may be a big gap in the computational and communication capabilities among participating nodes (i.e., edge servers and clients). As a result, the resources of faster nodes may be underutilized as they have to wait for slower nodes because of the synchronization barrier.

Next, we will show how to fully use heterogeneous resources to improve training efficiency.

Awareness of heterogeneity: The key idea is to use the optimal results of p^* and q^* as the benchmarks of local and global aggregation frequencies, thereby ensuring convergence. On this basis, appropriate adjustments are made according to the capability of each node to enhance efficiency by minimizing the waiting time between them. To minimize the waiting time for the local aggregation on each edge server, we try to have all clients managed by that edge server complete local updates and model upload tasks involving a single local aggregation in the same amount of time. Therefore, for each edge server i, we first find a client that can complete p^* local training iterations and upload the local model to this edge server at the fastest speed. Then, we assign p^* as the local aggregation frequency of the fastest client based on the condition that p^* is maximal in Theorem 1 and denote the corresponding completion time of the r-th global round as $\mathit{\Gamma}_i^r = \min_{j\in\mathcal{A}_i}\{p^* \times \mu_{client,j}^r + \beta_{client,j}^r\}$. Finally, we calculate the corresponding number of local iterations for other clients associated with this edge server according to (28). Similarly, to minimize the waiting time for the global aggregation on the cloud server, we use the time $\mathit{\Gamma}^r$ of the fastest edge server that can finish q^* local aggregations as a benchmark and compute the number of local aggregations for other edge servers according to (29). $\mathit{\Gamma}^r$ can be calculated by $\mathit{\Gamma}^r = \min_{i=1,\ldots,M}\{q^* \times \mathit{\Gamma}_i^r + \beta_{edge,i}^r\}$. For other symbols, please review Table I.

$$\text{For client} j \in \mathcal{A}_i: \quad p_j^r = \lfloor \frac{\mathit{\Gamma}_i^r - \beta_{client,j}^r}{\mu_{client,j}^r} \rfloor \tag{28}$$

$$\text{For edge server} i: \quad q_i^r = \lfloor \frac{\mathit{\Gamma}^r - \beta_{edge,i}^r}{\mathit{\Gamma}_i^r} \rfloor \tag{29}$$

With this design, the clients and edge servers with weaker capabilities will perform fewer local updates and local aggregations respectively in each global round, thus mitigating the delay caused by the synchronization barrier. It should be noted that assigning p^* and q^* to the fastest nodes may cause other nodes with poor performance to perform fewer model updates, thereby slowing down the convergence speed to a certain extent. However, this can be compensated because the fastest node in different global rounds often changes in a dynamic network. The experimental results prove that this choice has little impact on the convergence speed (see Section VII).

B. Dynamic Adjustment During Training Process

Due to the differences in performance between participating nodes, we need to dynamically select the local and global aggregation frequency for the nodes within each global round. At the same time, due to the dynamic changes of the network, we dynamically adjust the above two frequencies across global rounds. The main idea is to first estimate the values of p^* and q^* that minimize $\Phi(R, p, u)$ according to the training status in the current global round and then adapt them according to (28) and (29) to obtain the aggregation frequency of every client and edge server applied for the next global round. Meanwhile, we

Algorithm 1: Procedure at Client j ($\forall j \in \mathcal{A}_i$).

```
   Input: Learning rate $\eta$
   Output: Local model $\boldsymbol{\omega}_j^{(s,r)}(p_j^r)$
1  Estimate the resource overhead $c_j^r$ and $b_j^r$;
2  Receive $\boldsymbol{\omega}^{(r-1)}$, $\boldsymbol{\omega}_i^{(r)}(s-1)$ and $p_j^r$ from the
     corresponding edge server $i$;
3  Set $\boldsymbol{\omega}_j^{(s,r)}(0) \leftarrow \boldsymbol{\omega}_i^{(r)}(s-1)$;
4  for $t = 1, 2, ..., p_j^r$ do
5      Perform local update $\boldsymbol{\omega}_j^{(s,r)}(t) =$
         $\boldsymbol{\omega}_j^{(s,r)}(t-1) - \eta \cdot \nabla F\left(\boldsymbol{\omega}_j^{(s,r)}(t-1), \xi^{(s,r)}(t-1)\right)$;
6  end
7  if $r > 1$ then
8      Estimate Lipschitz constant
         $L_j \leftarrow \frac{\|\nabla F_j(\boldsymbol{\omega}_j^{(s,r)}(p_j^r)) - \nabla F_j(\boldsymbol{\omega}^{(r-1)})\|}{\|\boldsymbol{\omega}_j^{p_j^r,s,r} - \boldsymbol{\omega}^{(r-1)}\|}$;
9      Estimate gradient estimation variance
         $\sigma_j^2 \leftarrow \|\nabla F_j(\boldsymbol{\omega}^{(r-1)}, \xi_j^{(r-1)}) - \nabla F_j(\boldsymbol{\omega}^{(r-1)})\|^2$;
10 end
11 Compute resource consumption $C_j^r \leftarrow p_j^r \cdot c_j^r$;
12 Send $\boldsymbol{\omega}_j^{(s,r)}(p_j^r)$, $C_j^r$, $b_j^r$, $\mu_{client,j}^r$, $\beta_{client,j}^r$, $L_j$ and $\sigma_j^2$ to
     the corresponding edge server $i$.
```

assume that the network status information received in the r-th global round is relatively stable during the $(r+1)$-th global round. Specifically, we let the cloud server control the optimal calculation of frequencies p^* and q^* and the adjustment of the global aggregation frequency for edge servers, while each edge server controls the adjustment of the local aggregation frequency for clients.

Next, we explain how to dynamically adjust the local and global aggregation frequencies in detail by illustrating the procedures at the sides of the client (Algorithm 1), edge server (Algorithm 2), and cloud server (Algorithm 3) in each global round.

At the client side: Algorithm 1 gives the procedure of each client. Client j first estimates its own computational and communication resource overhead, and then receives global model, local aggregated model and local aggregation frequency from the associated edge server i. Subsequently, client j iteratively performs local update (line 5), estimates Lipschitz constant L_j and gradient estimation variance σ_j^2 locally (lines 8-9), then informs the associate edge server i of the estimated results along with its local model, resource consumption, computational and communication capability, with the given frequency p_j^r.

At the edge server side: The edge server sits between the cloud server and the clients, and Algorithm 2 shows its workflow in each global round. Considering that the time of each global round is relatively short, we assume that the resources of all clients are relatively stable within each global round, but fluctuate between global rounds. Each edge server i aggregates local models from clients by the end of each local aggregation and gets the training process parameters $L_i^s, (\sigma^2)_i^s$ and status information sent by clients in $\mathcal{A}_i$ (lines 8-10). Meanwhile, it adaptively adjusts the local aggregation frequency for clients following

Algorithm 2: Procedure at Edge Server i ($\forall i \in M$).

```
Input: Procedure at clients (Alg.1)
Output: Local aggregated model $\boldsymbol{\omega}_i^{(r)}(q_i^r)$
1  Estimate the communication overhead $b_i$;
2  Receive $\boldsymbol{\omega}^{(r-1)}$, $p^{r-1}$ and $q_i^{r-1}$ from the cloud server;
3  Set $\boldsymbol{\omega}_i^{(r)}(0) \leftarrow \boldsymbol{\omega}^{(r-1)}$, $p^r \leftarrow p^{r-1}$, $q_i^r \leftarrow q_i^{r-1}$;
4  for $s = 1, 2, ..., q_i^r$ do
5      Send $\boldsymbol{\omega}^{(r-1)}$, $\boldsymbol{\omega}_i^{(r)}(s-1)$ and $p_j^r$ to clients in $\mathcal{A}_i$;
6      Procedure at clients (Alg.1);
7      Receive $\{\boldsymbol{\omega}_j^{(s,r)}(p_j^r), C_j^r, b_j^r, \mu_{client,j}^r, \beta_{client,j}^r, L_j, \sigma_j^2\}$ from clients in $\mathcal{A}_i$;
8      Local aggregation:
       $\boldsymbol{\omega}_i^{(r)}(s) = \frac{1}{N_i}\sum_{j\in\mathcal{A}_i} \alpha_j^r \cdot \boldsymbol{\omega}_j^{(s,r)}(p_j^r)$;
9      Update $C_i^{s,r} \leftarrow \sum_{j\in\mathcal{A}_i} C_j^r$ and $B_i^{s,r} \leftarrow \sum_{j\in\mathcal{A}_i} b_j^r$;
10     Calculate parameters:
       $L_i^s \leftarrow \sum_{j\in\mathcal{A}_i} L_j$ and $(\sigma^2)_i^s \leftarrow \sum_{j\in\mathcal{A}_i} \sigma_j^2$;
11     Calculate
       $\Gamma_i^r = \min_{j\in\mathcal{A}_i}\left\{p^r \times \mu_{client,j}^r + \beta_{client,j}^r\right\}$;
12     Calculate local aggregation frequency:
       $p_j^r \leftarrow \lfloor \frac{\Gamma_i^r - \beta_{client,j}^r}{\mu_{client,j}^r} \rfloor, \forall j \in \mathcal{A}_i$;
13 end
14 Count $L_i \leftarrow \sum_s L_i^s$, $\sigma_i^2 \leftarrow \sum_s (\sigma^2)_i^s$;
15 Count $C_i^r \leftarrow \sum_s C_i^{s,r}$, $B_i^r \leftarrow b_i^r + \sum_s B_i^{s,r}$;
16 Send $\boldsymbol{\omega}_i^{(r)}(q_i^r)$, $C_i^r$, $B_i^r$, $\Gamma_i^r$, $\beta_{edge,i}^r$, $L_i$ and $\sigma_i^2$ to the cloud server.
```

the heterogeneity-aware computation strategy presented in Section VI-A (lines 11-12). Then, before completing the required rounds of local aggregation q_i^r, the edge server i distributes the aggregated models back to clients to continue training for the next local round. Otherwise, edge server i aggregates the training parameters and status information to obtain L_i, σ_i^2, C_i^r and B_i^r, and subsequently uploads them and the aggregated model to the cloud server.

At the cloud server side: As Algorithm 3 shows, the cloud server initializes the local and global aggregation frequencies based on the maximum settings of local SGD in [11] and initializes the search space to ensure q is smaller than p to accelerate convergence [4]. In each global round, the cloud server first updates the global model and the resource status according to the updated models and network information received from edge servers (lines 8-9). Subsequently, it aggregates parameters L, σ^2 received from edge servers (lines 10). On the basis of estimated L, σ^2, the optimal value of p^r and u is determined by minimizing $\Phi(R, p, u)$ in a specified search space, and accordingly the benchmark global aggregation frequency q^r for the next global round is calculated (lines 16-17). Finally, it adaptively adjusts the global aggregation frequency for edge servers following the heterogeneity-aware computation strategy presented in §VI-A (lines 18-19). The training process will be stopped if the resource constraints cannot be satisfied or the model reaches convergence (line 21). It should be noted that p^r and u are obtained by traversing the search under the powerful arithmetic of the cloud server, and the computation time is negligible.

Algorithm 3: Procedure at the Cloud Server.

```
Input: Global round $R$, number of clients $N$ and edge servers $M$, resource constraints $C$ and $B$, Procedure at edge servers (Alg.2)
Output: Global model $\boldsymbol{\omega}^R$
1  Initialize model parameters $\boldsymbol{\omega}^0$ for all nodes;
2  Initialize $C^0 = 0$, $B^0 = 0$, $p^0 = 50$, $q^0 = 1$;
3  Initialize $p^{\min} = 1$, $p^{\max} = 50$, $u_{\min} = 0$, $u_{\max} = 1$;
4  for $r = 1, ..., R$ do
5      Send $\boldsymbol{\omega}^{(r-1)}$, $p^{r-1}$ and $q_i^{r-1}$ to edge server $i$;
6      Procedure at edge servers (Alg.2);
7      Receive $\{\boldsymbol{\omega}_i^{(r)}(q_i^r), C_i^r, B_i^r, \Gamma_i^r, \beta_{edge,i}^r, L_i, \sigma_i^2\}$ from all edge servers;
8      Global aggregation: $\boldsymbol{\omega}^{(r)} = \frac{1}{M}\sum_{i=1}^M \alpha_i^r \boldsymbol{\omega}_i^{(r)}(q_i^r)$;
9      Update $C^r \leftarrow C^{r-1} + \sum_{i=1}^M C_i^r$ and
       $B^r \leftarrow B^{r-1} + \sum_{i=1}^M B_i^r$;
10     Calculate parameters:
       $L \leftarrow \sum_{i=1}^M L_i/N$ and $\sigma^2 \leftarrow \sum_{i=1}^M \sigma_i^2/N$;
11     Obtain $t^r = \max_i\left\{q_i^r \times \Gamma_i^r + \beta_{edge,i}^r\right\}$;
12     if $r \le 1$ then
13         Set $p^r = p^{r-1}$, $q_i^r = q_i^{r-1}$;
14     end
15     else
16         Search the optimal $p^r \in [p^{\min}, p^{\max}]$,
           $u \in [u_{\min}, u_{\max}]$ to minimize $\Phi(R, p, u)$;
17         Calculate $q^r = u \times p^r$;
18         Calculate $\Gamma^r = \min_i\left\{q^r \times \Gamma_i^r + \beta_{edge,i}^r\right\}$;
19         Calculate global aggregation frequency:
           $q_i^r \leftarrow \lfloor \frac{\Gamma^r - \beta_{edge,i}^r}{\Gamma_i^r} \rfloor, \forall i$;
20     end
21     Stop the training process if $C^r > C$ or $B^r > B$ or model converged.
22 end
23 return $\boldsymbol{\omega}^R$.
```

VII. Performance Evaluation

In this section, extensive experiments are conducted to evaluate the performance of FedAda, and the simulation results demonstrate its effectiveness and efficiency.

A. Methodology

Datasets and Models: We use two real-world datasets for FL training, Fashion-MNIST [35] and CIFAR-10 [36], both datasets contain 10 different classes. Fashion-MNIST comprises 60,000 28×28 10-class grey-scale images, including 50,000 training samples and 10,000 testing samples. While CIFAR-10 contains samples of 32×32 three-channel color images, with the same total number of image samples, image classification, and dataset partitioning as Fashion-MNIST. We conduct experiments on both IID data and Non-IID data, except for Non-IID data, all datasets are distributed uniformly across clients by default. To generate Non-IID data, we introduce a parameter $\varphi \in [0.1, 1]$ to represent the proportion of dominant classes within the dataset. The remaining samples were randomly and uniformly selected from the other nine classes. The parameter φ reflects the heterogeneity, where the heterogeneity of the dataset enlarges as φ increases. Based on the above real-world datasets, we train two popular models, CNN and ResNet9. Both CNN and ResNet9 are lightweight neural network architectures, which are well-suited for CEC-FL systems characterized by clients with limited computational resources. The datasets and models we chose are consistent with the experimental settings used in many prior Federated Learning studies such as [4], [8], [37]. The former is a CNN model specialized for Fashion-MNIST dataset, which has about 0.58 M parameters with two convolutional layers, two pooling layers, and one fully-connected layer. It is more challenging to train a model on CIFAR-10 dataset. The well-known ResNet9 model (about 2.45 M parameters) is used for it.

Baselines: We compare FedAda with state-of-the-art CEC-FL solutions that employ aggregation frequency control to enhance training performance like this work. Details are described below.

- *HierFAVG* [4] is the first CEC-FL solution that adopts a two-layer aggregation style to our knowledge. To make the results comparable, the frequency setting of ($p = 6$, $q = 10$) that yields the best training efficiency in their paper is used in experiments.
- *HFL* [11] also provides a solution based on two-level aggregation, along with a theoretical analysis of the effectiveness of local aggregation. Like HierFAVG, it lets participating nodes in the same layer adopt the same aggregation frequency. When evaluating performance, the frequency setting ($p = 5$, $q = 50$) they reported to have the best training efficiency is used in the experiments.
- *RAF* [12] allows participating nodes with different capabilities to use different aggregation frequencies. It employs a heuristic to compute aggregation frequencies, setting the frequency of the slowest client and edge server to one and adjusting the frequencies of other nodes based on these benchmarks.

Similar to FedAda, both HierFAVG and HFL offer theoretical analysis and convergence guarantees, although they focus on homogeneous resource scenarios. In contrast, RAF focuses on dynamic and heterogeneous networks as this work. These three baselines are currently the closest counterparts to this work.

Performance metrics: In the evaluation we mainly consider the following performance metrics.

- *Test accuracy:* measures the ability of a trained model to make correct predictions and is calculated as the ratio between the number of accurate predictions in the test dataset and the total number of that dataset. It is the primary metric of interest for an FL solution.
- *Total time:* includes the overall computation and communication time that an FL task takes for the model to reach the target or convergence accuracy. It reflects the convergence speed of an FL algorithm.
- *Average waiting time:* measures the ability of an algorithm to mitigate the delay caused by the synchronization barrier, which can be calculated via (13) and (14). This is often a concern for heterogeneous scenarios, as it reflects whether an FL solution effectively uses limited network capacities.
- *Communication overhead:* measures the amount of traffic load incurred by an FL training task to achieve the target or convergence accuracy, which is also a major concern in resource-constrained systems.

Simulation setup: We implement FedAda and all compared algorithms on the most popular FL framework, PyTorch. To conduct experiments, we simulate a CEC-FL system on a deep learning workstation equipped with an AMD Ryzen 9 5950X 16-Core CPU, 2 NVIDIA GeForce RTX 3090 GPUs, and 128 GB RAM and deploy the system using Docker. This system consists of one cloud server, four edge servers and 20 clients, with five clients under each edge server. In each global round, we randomly select 10 out of 20 nodes as clients to participate in the training process. This selection method simulates, to a certain extent, the intermittent availability of clients in real-world scenarios. All comparison algorithms are subjected to the same heterogeneity and training settings, which are described as follows.

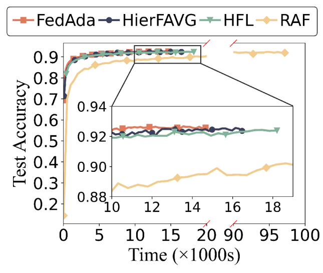
For performance evaluation under heterogeneous resources, we assume that the computation time to complete one local iteration varies across models and clients and that network bandwidth may also vary between nodes. To simulate this heterogeneity, we first give an average value, and then based on this, use the factor $\gamma \in [0, 1)$ to adjust the degree of heterogeneity of resources between nodes. Specifically, following prior related work [4], the average computation time μ is 0.5 s and 4.0 s for a local training iteration over CNN (Fashion-MNIST) and ResNet9 (CIFAR-10), respectively. The average bandwidth β available to clients and edges for transmitting model parameters is 4Mbps [12]. The local iteration computation time follows a uniform distribution of $\mu \sim U((1-\gamma)\mu, (1+\gamma)\mu)$ and the bandwidth fluctuates between $(1-\gamma)\beta$ Mbps and $(1+\gamma)\beta$ Mbps [8]. Note that $\gamma = 0$ indicates that available resources are homogeneous among all nodes. To simulate network dynamics, we change the values of local iteration computation time and bandwidth at regular intervals.

For model training, three different scenarios are considered: (i) IID data and homogeneous resource, (ii) IID data and heterogeneous resource, and (iii) Non-IID data and heterogeneous resource. The number of global rounds for training CNN on Fashion-MNIST are 400, while for training ResNet9 on CIFAR-10 are 400, 800, and 600, respectively. Besides, we use the momentum-SGD algorithm as an optimizer for all compared algorithms, where the momentum is set to 0.9. The local and global aggregation weights are averaged. The learning rate and batch size are set as 0.01 and 32 for all algorithms, respectively.

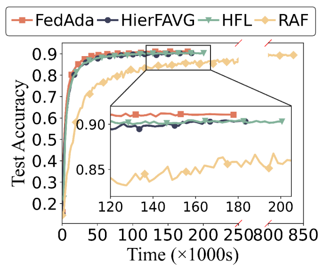
B. General Performance

First, we investigate the performance in terms of test accuracy, training time, average waiting time, and communication overhead of all compared algorithms under a wide spectrum of scenarios.

Accuracy and training time: Fig. 2 and Table II show the convergence results of all compared CEC-FL algorithms on IID data



(a) CNN on Fashion-MNIST



(b) ResNet9 on CIFAR-10

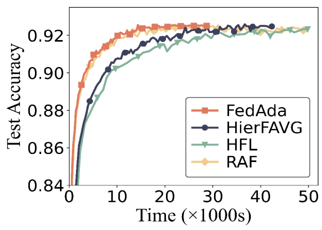
Fig. 2. Test accuracy (IID data, homogeneous resource).

TABLE II
TOTAL TIME (S) NEEDED TO ACHIEVE SPECIFIED ACCURACY (IID DATA, HOMOGENEOUS RESOURCE).

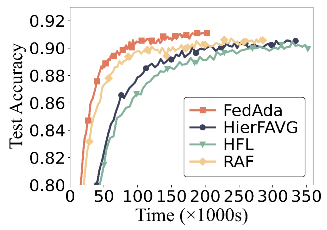
Algorithms	Accuracy (CNN)			Accuracy (ResNet9)		
	86%	89%	92%	80%	85%	90%
HierFAVG	745	1,821	7,532	17,097	30,035	126,148
HFL	1,092	2,184	7,697	18,053	30,088	106,312
RAF	4,201	10,921	47,139	69,494	127,715	734,371
FedAda	**532**	**1,350**	**6,949**	**13,128**	**21,514**	**60,449**

distribution in the homogeneous resource scenario. Fig. 2 shows the convergence process. FedAda always yields the best test accuracy for all trained models. HierFAVG, HFL, and RAF come second and have similar test accuracy. However, RAF has a much slower convergence speed. As Fig. 2(a) shows, FedAda HierFAVG and HFL converge to 92% accuracy at approximately 18,000 seconds, while RAF has not yet converged at that time and only achieves 90% accuracy. A similar situation occurs when training ResNet9 on CIAFR-10, and it is more obvious. At around 200,000 seconds, RAF is nearly 5% lower than the other three algorithms. Fig. 2(b) also shows that FedAda improves the test accuracy by 0.6% compared to baselines. Table II records the total time needed to achieve specified accuracy. FedAda has the fastest convergence speed, taking $6,949\,s$ to reach 92% accuracy for CNN on Fashion-MINIST, while HierFAVG, HFL, and RAF take $7,532\,s$, $7,697\,s$ and $47,139\,s$, respectively. While for ResNet9 on CIFAR-10, FedAda achieves a much faster convergence speed than HierFAVG, HFL, and RAF, with a speedup of about $2.1\times$, $1.8\times$, and $6.8\times$, respectively, to achieve 90% accuracy. When the computation and bandwidth resources are homogeneous, RAF will degenerate into HierFAVG with the aggregation frequency of $(p = 1, q = 1)$. As a result, it incurs frequent communication and produces the slowest convergence speed.

We also investigate the heterogeneous resource scenario, and Fig. 3 and Table III report the simulation results. In this part, we set $\gamma = 0.8$ to generate heterogeneous computation and communication capacities for the participating nodes. The results show that FedAda still achieves the highest test accuracy in this scenario, achieving around 92% and 91% test accuracy on CNN and ResNet9, respectively. We can also see from Fig. 3(b) that FedAda improves up to 0.3% test accuracy, compared with the other three baselines. It is worth noting that, unlike the homogeneous resource scenario, RAF now achieves faster convergence speed. This improvement should be due to the heterogeneous-aware adjustment mechanism of RAF, which helps it benefit from local aggregation. Table III shows that FedAda maintains the fastest convergence speed on both CNN and ResNet9. Specifically, FedAda reaches 92% accuracy on CNN with $9,895\,s$, $2.04\times$ and $2.88\times$ faster than HierFAVG and RAF, respectively. Regarding the training of RestNet9, it achieves a speedup of $2.76\times$ and $4.27\times$ to reach 90% accuracy compared to HierFAVG and HFL, respectively. We can also observe that both HierFAV, HFL, and FedAda take longer to achieve the specified accuracy in heterogeneous resources compared to homogeneous scenarios (see Table II). However, the increase of FedAda is minimal.



(a) CNN on Fashion-MNIST

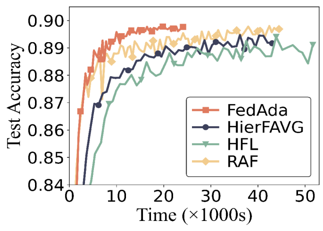


(b) ResNet9 on CIFAR-10

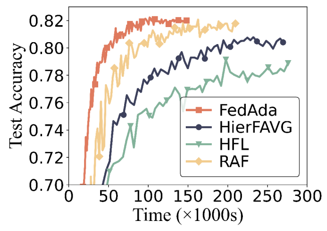
Fig. 3. Test accuracy (IID data, heterogeneous resource).

TABLE III
TOTAL TIME (S) NEEDED TO ACHIEVE SPECIFIED ACCURACY (IID DATA, HETEROGENEOUS RESOURCE).

Algorithms	Accuracy (CNN)			Accuracy (ResNet9)		
	86%	89%	92%	80%	85%	90%
HierFAVG	2,042	4,945	20,163	40,116	64,843	181,591
HFL	3,257	7,116	28,453	44,821	75,687	281,435
RAF	903	2,456	11,370	19,694	34,415	100,662
FedAda	**809**	**2,153**	**9,895**	**15,406**	**26,581**	**65,848**



(a) CNN on Fashion-MNIST



(b) ResNet9 on CIFAR-10

Fig. 4. Test accuracy (Non-IID data, heterogeneous resource).

TABLE IV
TOTAL TIME (S) NEEDED TO ACHIEVE SPECIFIED ACCURACY (NON-IID DATA, HETEROGENEOUS RESOURCE).

Algorithms	Accuracy (CNN)			Accuracy (ResNet9)		
	85%	87%	89%	78%	80%	82%
HierFAVG	3,621	6,378	19,142	101,027	164,335	-
HFL	5,415	9,197	32,668	197,566	-	-
RAF	1,556	3,064	7,878	50,810	69,382	161,929
FedAda	**1,324**	**2,993**	**5,723**	**31,112**	**45,885**	**77,322**

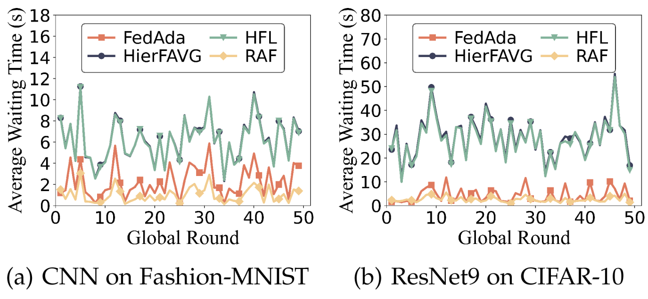


Fig. 5. Average waiting time of local aggregation.

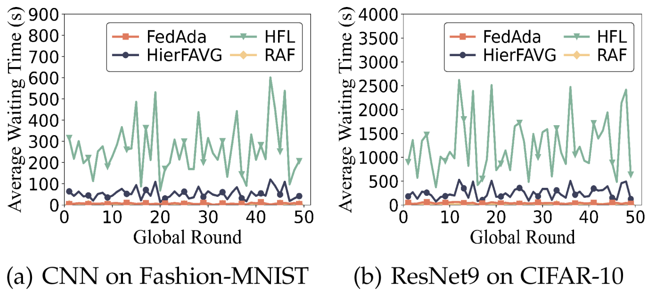


Fig. 6. Average waiting time of global aggregation.

Further, we evaluate algorithm performance under a more heterogeneous scenario, namely Non-IID data and heterogeneous resource scenario. Fig. 4 and Table IV show the results. Unless otherwise stated, the experiments in this part use $\varphi = 0.6$ and $\gamma = 0.8$ to simulate the heterogeneity of data and resource, respectively. We can see from Fig. 4(a) and (b) that the test accuracy of all algorithms drops compared with the first two scenarios. However, FedAda still achieves the highest accuracy, around 89% for CNN and 82% for ResNet9. Meanwhile, its performance is more stable while the accuracy of HFL, HierFAVG, and RAF decreases more drastically, especially training ResNet9 on CIFAR-10 (see Fig. 4(b)). In addition, as shown in Table IV, FedAda has the fastest convergence speed and RAF ranks second. Compared to HierFAVG and HFL, FedAda significantly improves the convergence speed by achieving a speedup of $3.3\times$ and $5.7\times$ respectively for CNN on Fashion-MNIST and $3.3\times$ and $6.4\times$ respectively for ResNet9 on CIFAR-10. Even compared to the RAF, FedAda is still $1.4\times$ and $1.6\times$ faster. The superior performance over RAF lies in the frequency computation of FedAda takes into account model convergence and not just resource heterogeneity.

Average waiting time: In the above experiments, we also record the average waiting time when CEC-FL uses different algorithms to perform local aggregation and global aggregation at the edge and cloud, respectively. Figs. 5 and 6 show the average waiting time in the scenario where both data and resource are heterogeneous. The result of the heterogeneous resource scenario with IID data is similar, we omit the presentation to save space. As expected, FedAda and RAF perform similarly, far better than HierFAVG and HFL. The reason is that both FedAda and RAF adaptively control aggregation frequency to match the computational and communication capabilities of nodes.

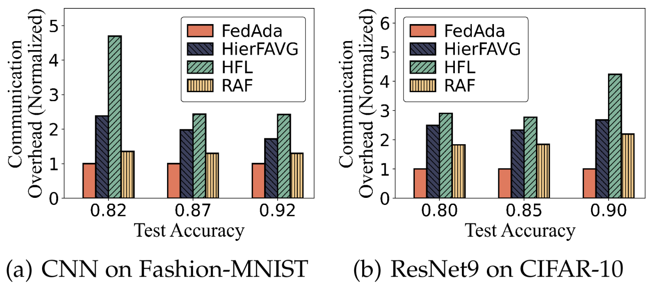


Fig. 7. Communication overhead (Normalized by FedAda).

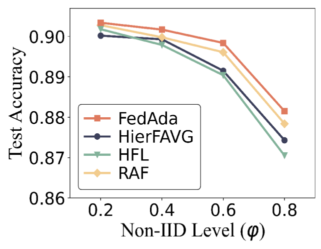
However, HierFAVG and HFL allow nodes to use the same and fixed frequency during training, resulting in long waiting times when resources between nodes are heterogeneous. In fact, heterogeneous resources are very common in real client-edge-cloud networks, limiting the practicality of HierFAVG and HFL. We can also see from Fig. 6(a) and (b) that for all compared algorithms, the waiting time of global aggregation is much longer than that of local aggregation. The reason is that before each global aggregation, the edge server may perform multiple rounds of local aggregation, resulting in accumulated waiting time.

Communication overhead: Further, we analyze the communication overhead of different algorithms to achieve a specified test accuracy. Fig. 7 shows the results normalized by FedAda, where FedAda outperforms all baselines. Specifically, FedAda saves $0.7\times$, $1.4\times$, and $0.3\times$ communication overhead for CNN on Fashion-MNIST reaching 92% accuracy compared to HierFAVG, HFL, and RAF, respectively. The communication overhead for FedAda to reach 90% accuracy on ResNet9 with CIFAR-10 saves $1.7\times$, $3.3\times$, and $1.2\times$ compared to HierFAVG, HFL, and RAF, respectively. These results are consistent with the convergence speed of FedAda because FedAda introduces fewer rounds of communication in the entire process, significantly reducing communication overhead and accelerating convergence. Overall, FedAda outperforms all compared baselines for CEC-FL training in a variety of scenarios, both homogeneous and heterogeneous. The most important reason is that it adaptively adjusts the aggregation frequency throughout the training process, by considering both model convergence and resource heterogeneity. This facilitates better utilization of node computing and communication capabilities, as well as improved model quality and convergence speed.

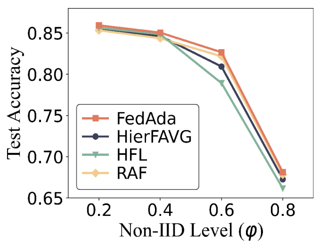
C. Performance Under Different Heterogeneity Levels

The above experiments show that the training performance has a great relationship with data and resource heterogeneity. Concretely, data heterogeneity can impact model accuracy and resource heterogeneity largely impacts convergence speed. Therefore, we further investigate how these two heterogeneities impact algorithm performance.

Impact of data heterogeneity: First, we run FL training tasks separately under different levels (φ) of data heterogeneity. Fig. 8 shows the experiment results that all the algorithms suffer from an increasing loss in test accuracy as the Non-IID level grows.

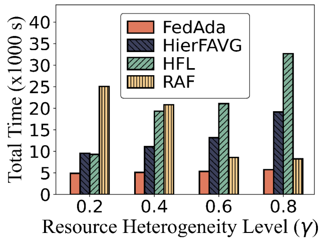


(a) CNN on Fashion-MNIST

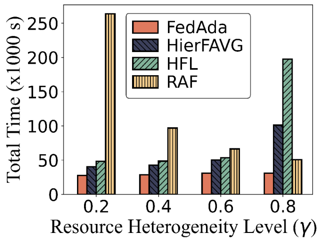


(b) ResNet9 on CIFAR-10

Fig. 8. Convergence accuracy under different data heterogeneity levels.



(a) CNN on Fashion-MNIST

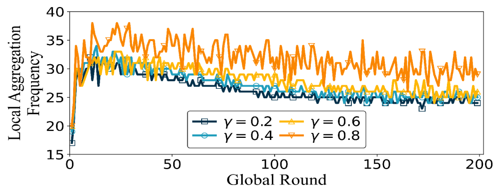


(b) ResNet9 on CIFAR-10

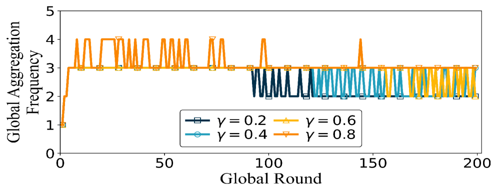
Fig. 9. Total time to reach target accuracy under different resource heterogeneity levels.

However, FedAda is more robust than other baselines. We also see that the accuracy loss of CNN on Fashion-MINIST is less than that of ResNet9 on CIFAR-10. For CNN on Fashion-MINIST, when φ increases from 0.2 to 0.6, the accuracy drops by less than 1%, and when φ increases from 0.6 to 0.8, the loss increases to $2\% - 3\%$. While for ResNet9 on CIFAR-10, we can see more loss, nearly 20% when the φ increases from 0.2 to 0.8. Nevertheless, FedAda maintains a high standard when the level of data heterogeneity is relatively low (i.e., $\varphi \leq 0.6$). However, the results indicate that high-level data heterogeneity will seriously affect model quality and needs to be handled more carefully. Fortunately, we have seen many techniques [38], [39], [40] proposed to deal with the performance degradation issue of Non-IID data. In the future, we will further investigate how to combine these techniques with adaptive frequency control to address the challenge of Non-IID data with high-level heterogeneity.

Impact of resource heterogeneity: Then, we rerun FL training tasks under different levels (γ) of resource heterogeneity. Fig. 9(a) and (b) show the total time for CNN and ResNet 9 to achieve the specified accuracy of 89% and 78% at different levels of resource heterogeneity, respectively. The data heterogeneity is set to $\varphi = 0.6$ in these experiments. The results demonstrate that FedAda still maintains the shortest completion time and exhibits stable performance in all cases. As expected, HierFAVG and HFL increase as the resource heterogeneity level grows. Since both algorithms allow all nodes in the same layer to use the same frequency, the synchronization-blocking effect will become more severe as the degree of resource heterogeneity increases. As a result, the completion time of each local and global aggregation round increases with the degree of resource heterogeneity, as does the total training time. Interestingly, RAF has a decreasing completion time as the level of resource heterogeneity rises. This is because when the heterogeneity level is low, RAF incurs frequent communication that takes up a large portion of the total time, whereas as the heterogeneity level increases, more local aggregations are allowed and result in a speedup of convergence. This again confirms the benefit of local aggregation in accelerating model convergence.



(a) Local aggregation frequency p



(b) Global aggregation frequency q

Fig. 10. Aggregation frequency over time under different resource heterogeneity levels.

Finally, we conduct experiments under different levels of resource heterogeneity and see how the aggregation frequency changes as training proceeds. Fig. 10 depicts how the aggregation frequency changes over time, in terms of global training rounds. In these experiments, we use Non-IID data with a heterogeneity level of $\varphi = 0.6$ for FL training. Fig. 10(a) shows that the local aggregation frequency p generally decreases as the number of global rounds increases, and increases as the resource heterogeneity γ increases. The reason why local aggregation frequency decreases as training progresses is that the deviation of the local model from the global model gradually decreases, and fewer local iterations help prevent model overfitting. The reason why p increases with the resource heterogeneity is that as γ increases, the number of local iterations of clients with few resources decreases, resulting in insufficient local training and an increasing deviation from the global model. Therefore, when γ increases, p needs to be increased to ensure that poor clients also experience enough local iterations to ensure the quality of the global model. Compared with p, Fig. 10(b) shows that the global aggregation frequency q is more stable and only fluctuates between 1 and 4 throughout the training process. This shows that FedAda effectively exploits the advantage of edge computing, allowing more local aggregation and less global aggregation, thereby fully obtaining the benefits of local aggregation in promoting model convergence and improving communication efficiency.

VIII. Conclusion

This article investigates the aggregation frequency control problem for efficient FL training in dynamic heterogeneous and resource-constrained client-edge-cloud networks. To solve this problem, we formulate it as an optimization problem and theoretically analyze the influence of the aggregation frequency on model convergence. Based on our convergence analysis, we present an adaptive aggregation frequency adjustment method to improve training quality and speed. Extensive evaluation results demonstrate the effectiveness and efficiency of the proposed method.

References

[1] B. McMahan et al., "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[2] M. Waqas et al., "The role of artificial intelligence and machine learning in wireless networks security: Principle, practice, and challenges," *Artif. Intell. Rev.*, vol. 55, no. 7, pp. 5215–5261, 2022.

[3] H. Zhou, Z. Li, H. Yu, L. Luo, and G. Sun, "NBSync: Parallelism of local computing and global synchronization for fast distributed machine learning in wans," *IEEE Trans. Services Comput.*, vol. 16, no. 6, pp. 4115–4127, Nov/Dec. 2023.

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

[5] X. Fang and M. Ye, "Robust federated learning with noisy and heterogeneous clients," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10072–10081.

[6] G. Malinovskiy et al., "From local SGD to local fixed-point methods for federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 6692–6701.

[7] B. Luo, X. Li, S. Wang, J. Huang, and L. Tassiulas, "Cost-effective federated learning design," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.

[8] Y. Xu, Y. Liao, H. Xu, Z. Ma, L. Wang, and J. Liu, "Adaptive control of local updating and model compression for efficient federated learning," *IEEE Trans. Mobile Comput.*, vol. 22, no. 10, pp. 5675–5689, Oct. 2023.

[9] M. K. Nori, S. Yun, and I. -M. Kim, "Fast federated learning by balancing communication trade-offs," *IEEE Trans. Commun.*, vol. 69, no. 8, pp. 5168–5182, Aug. 2021.

[10] W. Liu et al., "AdaCoOpt: Leverage the interplay of batch size and aggregation frequency for federated learning," in *Proc. IEEE/ACM 31st Int. Symp. Qual. Service*, 2023, pp. 1–10.

[11] J. Wang et al., "Demystifying why local aggregation helps: Convergence analysis of hierarchical SGD," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 8548–8556.

[12] L. Yang, Y. Gan, J. Cao, and Z. Wang, "Optimizing aggregation frequency for hierarchical model training in heterogeneous edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 7, pp. 4181–4194, Jul. 2023.

[13] T. Zhang, L. Gao, C. He, M. Zhang, B. Krishnamachari, and A. S. Avestimehr, "Federated learning for the Internet of Things: Applications, challenges, and opportunities," *IEEE Internet Things Mag.*, vol. 5, no. 1, pp. 24–29, Mar. 2022.

[14] P. Qi et al., "Model aggregation techniques in federated learning: A comprehensive survey," *Future Gener. Comput. Syst.*, vol. 150, pp. 272–293, 2023.

[15] W. J. Yun et al., "SlimFL: Federated learning with superposition coding over slimmable neural networks," *ACM Trans. Netw.*, vol. 31, pp. 2499–2514, 2023.

[16] A. A. Al-Saedi et al., "Reducing communication overhead of federated learning through clustering analysis," in *Proc. IEEE Symp. Comput. Commun.*, 2021, pp. 1–7.

[17] Y. Ren et al., "Two-layer accumulated quantized compression for communication-efficient federated learning: TLAQC," *Sci. Rep.*, vol. 13, no. 1, 2023, Art. no. 11658.

[18] S. Lu et al., "Top-k sparsification with secure aggregation for privacy-preserving federated learning," *Comput. Secur.*, vol. 124, 2023, Art. no. 102993.

[19] W. Sun, S. Lei, L. Wang, Z. Liu, and Y. Zhang, "Adaptive federated learning and digital twin for industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5605–5614, Aug. 2021.

[20] M. Zinkevich et al., "Parallelized stochastic gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, vol. 2, pp. 2595–2603.

[21] S. U. Stich, "Local SGD converges fast and communicates little," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 3514–3530.

[22] Y. Cui, K. Cao, J. Zhou, and T. Wei, "Optimizing training efficiency and cost of hierarchical federated learning in heterogeneous mobile-edge cloud computing," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 42, no. 5, pp. 1518–1531, Mar. 2023.

[23] Z. Li, H. Zhou, T. Zhou, H. Yu, Z. Xu, and G. Sun, "ESync: Accelerating intra-domain federated learning in heterogeneous data centers," *IEEE Trans. Services Comput.*, vol. 15, no. 4, pp. 2261–2274, Jul./Aug. 2022.

[24] Q. Wu et al., "HiFlash: Communication-efficient hierarchical federated learning with adaptive staleness control and heterogeneity-aware client-edge association," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 5, pp. 1560–1579, May 2023.

[25] Z. Feng, X. Chen, Q. Wu, W. Wu, X. Zhang, and Q. Huang, "FedDD: Toward communication-efficient federated learning with differential parameter dropout," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 5366–5384, May 2024.

[26] T. Xiang et al., "Federated learning with dynamic epoch adjustment and collaborative training in mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 4092–4106, May 2024.

[27] S. Arora et al., "Understanding gradient descent on the edge of stability in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 948–1024.

[28] Y. Xiong, R. Wang, M. Cheng, F. Yu, and C. -J. Hsieh, "FedDM: Iterative distribution matching for communication-efficient federated learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 16323–16332.

[29] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Convergence time optimization for federated learning over wireless networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 4, pp. 2457–2471, Apr. 2021.

[30] Z. Zhao et al., "Towards efficient communications in federated learning: A contemporary survey," *J. Franklin Inst.*, vol. 360, pp. 8669–8703, 2023.

[31] Y. J. Cho, J. Wang, T. Chirvolu, and G. Joshi, "Communication-efficient and model-heterogeneous personalized federated learning via clustered knowledge transfer," *IEEE J. Sel. Topics Signal Process.*, vol. 17, no. 1, pp. 234–247, Jan. 2023.

[32] C. Zhao et al., "FedSup: A communication-efficient federated learning fatigue driving behaviors supervision approach," *Future Gener. Comput. Syst.*, vol. 138, pp. 52–60, 2023.

[33] M. Chen et al., "Communication-efficient federated learning," *Proc. Nat. Acad. Sci. USA*, vol. 118, no. 17, 2021, Art. no. e2024789118.

[34] F. Haddadpour et al., "Federated learning with compression: Unified analysis and sharp guarantees," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2021, pp. 2350–2358.

[35] G. Ayache, V. Dassari, and S. E. Rouayheb, "Walk for learning: A random walk approach for federated learning from heterogeneous data," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 929–940, Apr. 2023.

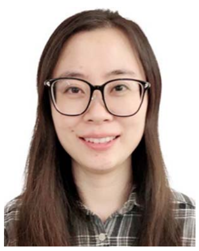
[36] J. Du, B. Jiang, C. Jiang, Y. Shi, and Z. Han, "Gradient and channel aware dynamic scheduling for over-the-air computation in federated edge learning systems," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 1035–1050, Apr. 2023.

[37] A. Panda et al., "Sparsefed: Mitigating model poisoning attacks in federated learning with sparsification," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2022, pp. 7587–7624.

[38] X. Mu et al., "Fedproc: Prototypical contrastive federated learning on non-iid data," *Future Gener. Comput. Syst.*, vol. 143, pp. 93–104, 2023.

[39] Y.-T. Cao et al., "Knowledge-aware federated active learning with non-IID data," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2023, pp. 22279–22289.

[40] S. Arisdakessian, O. A. Wahab, and A. Mourad, "Coalitional federated learning: Improving communication and training on non-IID data with selfish clients," *IEEE Trans. Services Comput.*, vol. 16, no. 4, pp. 2462–2476, Jul./Aug. 2023.



Long Luo (Member, IEEE) received the MS and PhD degrees in communication engineering from the University of Electronic Science and Technology of China, in 2015 and 2020, respectively. She is an Associate Professor with the University of Electronic Science and Technology of China (UESTC). Her research interests include networking and distributed systems.

Chi Zhang (Student Member, IEEE) is currently working toward the master's degree with the School of Information and Communication Engineering, University of Electronic Science and Technology of China. His research interests include distributed computing and federated learning.

Hongfang Yu (Member, IEEE) received the BS degree in electrical engineering from Xidian University, in 1996, and the MS and PhD degrees in communication and information engineering in from UESTC, 1999 and 2006, respectively. She is a professor with the University of Electronic Science and Technology of China (UESTC). Her research interests include AI network systems and network security.

Gang Sun (Senior Member, IEEE) is a professor with the University of Electronic Science and Technology of China. His research interests include network virtualization, cloud computing, high-performance computing, parallel and distributed systems, ubiquitous/pervasive computing and intelligence and cyber security.

Shouxi Luo (Member, IEEE) received the MS and PhD degrees in communication engineering from the UESTC, in 2011 and 2016, respectively. He is an associate professor with the School of Computing and Artificial Intelligence, Southwest Jiaotong University. His research interests include data center networks, software-defined networking, and networked systems.

Schahram Dustdar (Fellow, IEEE) received the PhD degree in business informatics from the University of Linz, Austria, in 1992. He is currently a full professor of computer science (informatics) with a focus on internet technologies heading the Distributed Systems Group, TU Wien, Wein, Austria. He has been the Chairman of the Informatics Section of the Academia Europaea, a member of the IEEE Conference Activities Committee, the Section Committee of Informatics of the Academia Europaea, and the Academia Europaea: The Academy of Europe, Informatics Section. He was a recipient of the ACM Distinguished Scientist Award and the IBM Faculty Award.