Convergence Analysis of Hierarchical Split Federated Learning

Hualei Zhang*†, Jun Du*†, Xiangwang Hou*, Chunxiao Jiang†, Jintao Wang*, Dusit Niyato§
*Department of Electronic Engineering, Tsinghua University, Beijing, 100084, China.
†Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing 100084, China.
§ College of Computing and Data Science, Nanyang Technological University, Nanyang Avenue, 639798, Singapore.
E-mail: zhang-hl22@mails.tsinghua.edu.cn, jundu@tsinghua.edu.cn, hxw21@mails.tsinghua.edu.cn, jchx@tsinghua.edu.cn, wangjintao@tsinghua.edu.cn, dniyato@ntu.edu.sg

***Abstract*—Federated Learning (FL) enables distributed intelligence in Internet of Things (IoT) networks, facilitating decentralized machine learning without the need for exchanging raw data. However, the growing complexity of training models significantly hinders their deployment on resource-constrained IoT devices. To address this challenge, Split Federated Learning (SFL) has emerged as a promising solution by partitioning the entire model into client-side and server-side sub-models to alleviate the computational burden on IoT devices. Considering that the client-edge-cloud architecture can enhance data privacy, support connections to a wider range of devices, and reduce communication costs, we explore a hierarchical SFL (HierSFL) system. This system is supported by a HierSFL algorithm that allows for different aggregation frequencies between the client-side and server-side sub-models. Then, we present a convergence analysis of HierSFL that quantifies the effects of client-side and server-side model aggregation on learning performance, providing a theoretical foundation. Empirical experiments verify the theoretical analysis and demonstrate the superiority of the hierarchical architecture within a wireless IoT network. In particular, it is validated that adopting different aggregation frequencies can enhance the training performance. Moreover, the HierSFL algorithm outperforms traditional hierarchical FL algorithm, achieving superior test accuracy in a shorter time.**

I. Introduction

With the proliferation of modern sensors, voluminous datasets has been collected and trained for provisioning a wide range of intelligent services in Internet of Things (IoT) networks. The raw data generated by IoT devices is very often private or sensitive, and transmitting the data to a server for training can be detrimental to privacy protection [1]. As a response, Federated Learning (FL) has been recognized as a promising solution for privacy-preserving IoT applications. In FL, all participating devices perform model training using their locally stored data in parallel. Instead of sharing the raw data with a central server, the devices exchange only the model parameters or the model gradients. However, the practical implementation of FL still faces significant challenges. As machine learning (ML) models become more complex and scale up, training at resource-constrained IoT devices becomes infeasible [2].

Split Learning (SL) [3] is a promising distributed ML framework that has the potential to overcome the limitations of FL. Specifically, SL partitions the global model into client-side and server-side sub-models. These sub-models are deployed on the clients and the central server, respectively, enabling collaborative training between the clients and the server [4]. However, SL employs a sequential training paradigm, where the model is passed from one device to another in a sequential manner. This sequential training mechanism increases the overall training time, reducing the training efficiency [5], [6]. To address this limitation, a new framework called Split Federated Learning (SFL) has been proposed [7]. SFL brings together the benefits of model partitioning from SL and the parallel training approach of FL. In SFL, the training process involves periodic aggregation of the server-side and client-side sub-models, aligning with the design principles of FL. Concurrently, to accommodate a larger number of devices and leverage more data for training, a hierarchical SFL (HierSFL) framework across the cloud-edge-client tiers has gained attention [1], [8], [9]. This approach not only helps preserve data privacy but also reduces the communication overhead during the model training process. By aggregating the client-side and server-side sub-models at the edge server, the amount of data transmitted from the edge to the cloud can be largely minimized, thereby mitigating the communication costs during the training process.

From the preceding discussion, it is evident that investigating the client-edge-cloud HierSFL framework is a highly necessary. In this scenario, the essential decisions pertaining to the model aggregation frequency and the model splitting strategy have a significant impact on the model training performance. Notably, the HierSFL approach exhibits distinct convergence characteristics compared to HierFL [10]. This is primarily attributed to the fact that in the HierSFL framework, the server-side sub-models and client-side sub-models can be aggregated at distinct frequencies. In particular, the aggregation of server-side sub-models, which are co-located on the same server, does not incur additional communication overhead, unlike the case for client-side sub-models residing across different devices. Therefore, the design of a HierSFL system is nontrivial. **(1)** If the aggregation frequencies of the client-side and server-side sub-models are disparate, will the entire model still be able to converge? **(2)** In comparison to the synchronous aggregation of client-side and server-side sub-models, does this asynchronous aggregation mechanism offer any performance benefits? **(3)** How does the model splitting points impact the training

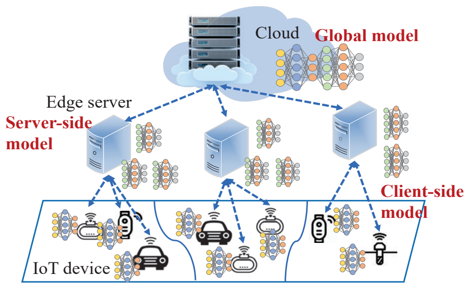


Fig. 1. Illustration of client-edge-cloud architecture for IoT services.

performance in the HierSFL framework?

In this work, we address the above key questions. The first key result is a rigorous proof confirming the convergence of the HierSFL algorithm and demonstrating that the asynchronous aggregation mechanism can achieve better training performance, solving questions **(1)** and **(2)**. Additionally, the convergence analysis demonstrates the impact of model split depth and provides guidance on the appropriate choice of model split points, addressing question **(3)**. Experimental results on the CIFAR-10 dataset validate our theoretical insights and showcase the advantages of the HierSFL framework under different aggregation frequencies and varying model split points.

The rest of the paper is organized as follows. Section II introduces the HierSFL algorithm in the client-edge-cloud IoT scenario. In Section III, the convergence analysis about aggregation strategy is provided. Section IV draws the simulation results, followed by the conclusions in Section V.

II. Hierarchical Split Federated Learning Systems

In this section, we first present the traditional two-layer SFL architecture. Then, we present the HierSFL algorithm with different aggregation frequencies in the client-edge-cloud IoT scenario.

A. Traditional Two-Layer SFL

In the traditional two-layer SFL system, there is one central fed server, one central main server and N clients. Each client $n \in \mathcal{N} = \{1, 2, \ldots, N\}$ possesses its local private dataset $\mathcal{D}_n = \{\mathbf{o}_{n,i}, y_{n,i}\}_{i=1}^{D_n}$, where $\mathbf{o}_{n,i}$ and $y_{n,i}$ represent the i-th input data and its corresponding label, respectively. The global model $\mathbf{x}$ is split at the cut layer into two sub-models as $\mathbf{x} = [\mathbf{x}_r, \mathbf{x}_s]$. The model $\mathbf{x}_r$ pertains to the client-side model, while $\mathbf{x}_s$ corresponds to the server-side model. The clients train models with the help of the fed server and the main server. The main server with higher computing capability is responsible for performing server-side sub-model training, and maintains a separate server-side model corresponding to each client. Meanwhile, the fed server handles the periodic aggregation of client-side sub-models [11].

The objective of SFL is to find the optimal global model $\mathbf{x}^*$ that achieves the minimal expected loss over all clients:

$$\min_{\mathbf{x}} f(\mathbf{x}) \triangleq \min_{\mathbf{x}} \frac{1}{N} \sum_{n=1}^{N} f_n(\mathbf{x}), \tag{1}$$

where $f_n(\mathbf{x})$ represents the local loss function of client n over its local dataset $\mathcal{D}_n$, i.e., $f_n(\mathbf{x}) \triangleq \mathbb{E}_{\xi_n \sim \mathcal{D}_n} \left[\tilde{f}_n(\mathbf{x}; \xi_n)\right]$, and ξ_n denotes the mini-batch data sampled randomly from the local dataset.

B. Client-Edge-Cloud Hierarchical SFL

Given that the client-edge-cloud architecture can help preserve data privacy and support connections to a larger number of devices while reducing communication costs, we explore the HierSFL system. Clients typically represent IoT devices with limited computational capabilities. Edge servers represent IoT gateways equipped with more robust computational capabilities. The cloud refers to a centralized site with ample computational resources. As shown in Fig. 1, we consider a HierSFL system, which consists of one cloud server, Z edge servers indexed by z, and N clients indexed by n. Each edge server oversees a distinct set of clients, denoted as $\{\mathbf{n}^z\}_{z=1}^{Z}$, where $|\mathbf{n}^z| = n^z$ represents the number of clients under each edge server.

Before the model training, the cloud initializes the entire global model $\mathbf{x}^0$, and partitions it into client-side model $\mathbf{x}_r^0$ and server-side model $\mathbf{x}_s^0$ at the cut layer. Then, HierSFL takes a total number of K rounds to solve Eq. (1) until the model converges. $\mathbf{w}_{s,n}^{k,t_2,t_1}$ and $\mathbf{w}_{r,n}^{k,t_2,t_1}$ denote the server-side and client-side model parameters, respectively, after k rounds of cloud aggregation, t_2 rounds of edge-aggregation and t_1 local update steps for client n. Similarly, the parameters of the client-side and server-side sub-models on edge server z after k rounds of cloud aggregation and t_2 rounds of edge aggregation are represented by $\mathbf{u}_{r,z}^{k,t_2}$ and $\mathbf{u}_{s,z}^{k,t_2}$, respectively. The parameters of the client-side and server-side sub-models on the cloud after k rounds of cloud aggregation are denoted by $\mathbf{x}_r^k$ and $\mathbf{x}_s^k$, respectively. For a training round $k \in \{1, 2 \ldots, K\}$, the training process of HierSFL is detailed as follows.

1) Model Initialization: Each client downloads the aggregated client-side model $\mathbf{x}_r^k$ and each edge server downloads the aggregated server-side model $\mathbf{x}_s^k$ from the cloud.

2) Client-side Model Forward Propagation: All clients perform forward propagation in parallel with a batch of data samples, and transmit the intermediate features (also known as smashed data) and labels to the corresponding edge servers.

3) Server-side Model Forward Propagation and Backward Propagation: After receiving features from clients, each edge server feeds these data into the server-side sub-models to execute the server-side forward propagation and obtains the predicted values for each client. The predicted values and labels are utilized to calculate loss function values and further complete the backward propagation to update the server-side sub-models. Then, edge servers send the gradients over activation at the cut layer back to the clients. Simultaneously, the edge servers aggregate server-side sub-models to enhance training performance, which does not incur additional communication costs. Therefore, one iteration for the server-side sub-model can be written as

$$\mathbf{w}_{s,n}^{k,t_2,t_1+1} = \mathbf{w}_{s,n}^{k,t_2,t_1} - \eta \frac{1}{n^z} \sum_{i=1}^{n^z} \tilde{\nabla} f_{s,i}\left(\mathbf{w}_{s,i}^{k,t_2,t_1}\right), \tag{2}$$

where $\tilde{\nabla} f_{s,i}\left(\mathbf{w}_{s,i}^{k,t_2,t_1}\right)$ is the stochastic gradient for a mini-batch given the server-side model $\mathbf{w}_{s,n}^{k,t_2,t_1}$ and the output of the client-side model $\mathbf{w}_{r,n}^{k,t_2,t_1}$ of client i. η is the learning rate.

4) Client-side Model Backward Propagation: Each client updates its client-side model parameters based on the received gradients by performing backward propagation. Accordingly, one iteration for the client-side model is expressed as follows:

$$\mathbf{w}_{r,n}^{k,t_2,t_1+1} = \mathbf{w}_{r,n}^{k,t_2,t_1} - \eta \tilde{\nabla} f_{r,n}\left(\mathbf{w}_{r,n}^{k,t_2,t_1}\right), \tag{3}$$

where $\tilde{\nabla} f_{r,n}\left(\mathbf{w}_{r,n}^{k,t_2,t_1}\right)$ is the stochastic gradient given the client-side model $\mathbf{w}_{r,n}^{k,t_2,t_1}$ and the received gradient from the corresponding edge server.

5) Edge Aggregation: After every τ_1 training iterations including steps 2) to 4), clients send the client-side model parameters to the corresponding edge servers, and the edge servers aggregate both client-side models and server-side models. Thus, model aggregation at edge server can be given by

$$\mathbf{w}_{r,n}^{k,t_2+1,0} = \mathbf{u}_{r,z}^{k,t_2+1} = \mathbf{u}_{r,z}^{k,t_2} + \frac{1}{n^z}\sum_{i=1}^{n^z}\left(\mathbf{w}_{r,i}^{k,t_2,\tau_1} - \mathbf{w}_{r,i}^{k,t_2,0}\right), \tag{4}$$

$$\mathbf{w}_{s,n}^{k,t_2+1,0} = \mathbf{u}_{s,z}^{k,t_2+1} = \mathbf{u}_{s,z}^{k,t_2} + \frac{1}{n^z}\sum_{i=1}^{n^z}\left(\mathbf{w}_{s,i}^{k,t_2,\tau_1} - \mathbf{w}_{s,i}^{k,t_2,0}\right). \tag{5}$$

6) Cloud Aggregation: After every τ_2 edge aggregations, each edge server transmits the aggregated model parameters to the cloud for aggregation, which can be represented as

$$\mathbf{x}_r^{k+1} = \mathbf{x}_r^k + \sum\nolimits_{z=1}^{Z} \frac{n^z}{N}\left(\mathbf{u}_{r,z}^{k,\tau_2} - \mathbf{x}_r^k\right), \tag{6}$$

$$\mathbf{x}_s^{k+1} = \mathbf{x}_s^k + \sum\nolimits_{z=1}^{Z} \frac{n^z}{N}\left(\mathbf{u}_{s,z}^{k,\tau_2} - \mathbf{x}_s^k\right). \tag{7}$$

More specifically, the client-side model undergoes edge aggregation every τ_1 training iterations and cloud aggregation every $\tau_1\tau_2$ iterations. In contrast, the server-side model undergoes edge aggregation every training iteration and cloud aggregation every $\tau_1\tau_2$ iterations. The detail of the HierSFL algorithm is summarized in Algorithm 1.

III. Convergence Analysis of HierSFL

In this section, we provide the convergence analysis of HierSFL that characterizes the effect of client-side model aggregation frequency and server-side model aggregation frequency on the training performance. Due to space limitations, the detailed proof is available at https://github.com/zhl606/HierSFL/blob/main/proofHierSFL.pdf. Due to the large number of nonlinear layers in deep learning models, such as the ReLU activation function, the loss functions are typically non-convex. To ensure wider applicability, we present the convergence analysis for non-convex loss functions.

A. Preliminaries

Assumption 1. *(L-Smoothness). The loss function $f(x)$ is L-smooth with the Lipschitz constant $0<L<\infty$, i.e., for any x and y, we have*

$$\|\nabla f(x) - \nabla f(y)\| \le L\|x-y\|, \tag{8}$$

Algorithm 1 Hierarchical Split Federated Learning (HierSFL)

```
1: Initialize the entire model on the cloud $\mathbf{x}^0 = [\mathbf{x}_r^0, \mathbf{x}_s^0]$;
2: for $k = 0 : K-1$ do
3:   for $z = 1 : Z$ in parallel do
4:     Set the server-side model same as the cloud model $\mathbf{x}_s^k$, i.e., $\mathbf{u}_{s,z}^{k,0} = \mathbf{x}_s^k$;
5:     Set the aggregated client-side model stored in the edge server same as the cloud model $\mathbf{x}_r^k$, i.e., $\mathbf{u}_{r,z}^{k,0} = \mathbf{x}_r^k$;
6:     for $t_2 = 0 : \tau_2 - 1$ do
7:       for client $n \in \mathbf{n}^z$ in parallel do
8:         Set the client-side model same as the associated client-side model stored in the edge server, i.e., $\mathbf{w}_{r,n}^{k,t_2,0} = \mathbf{u}_{r,z}^{k,t_2}$;
9:         for $t_1 = 0 : \tau_1 - 1$ do
10:          Perform the forward and backward propagation as described in steps 2) to 4) in Section II-B.
11:        end for
12:        Client $n$ sends $\left(\mathbf{w}_{r,n}^{k,t_2,\tau_1} - \mathbf{w}_{r,n}^{k,t_2,0}\right)$ to its associated edge server;
13:      end for
14:      Edge server $z$ performs client-side model aggregation and stores the aggregated client-side model;
15:    end for
16:    Edge server $z$ sends $\left[\mathbf{u}_{r,z}^{k,\tau_2} - \mathbf{x}_r^k, \mathbf{u}_{s,z}^{k,\tau_2} - \mathbf{x}_s^k\right]$ to the cloud;
17:  end for
18:  Cloud aggregates the received models from all edge servers;
19: end for
```

Assumption 2. *(Variance of SGD). The variances of stochastic gradients for client-side model and server-side model across each layer are bounded*

$$\mathbb{E}_{\xi_n\sim\mathcal{D}_n}\left\|\tilde{\nabla} f_{r,n}(\mathbf{w}) - \nabla f_{r,n}(\mathbf{w})\right\|^2 \le \sum\nolimits_{l=1}^{L_c} \sigma_l^2, \forall \mathbf{w}, \forall n, \tag{9}$$

$$\mathbb{E}_{\xi_n\sim\mathcal{D}_n}\left\|\tilde{\nabla} f_{s,n}(\mathbf{w}) - \nabla f_{s,n}(\mathbf{w})\right\|^2 \le \sum\nolimits_{l=L_c+1}^{L^T} \sigma_l^2, \forall \mathbf{w}, \forall n, \tag{10}$$

where L_c is the number of client-side model layers, also referred to as the model split depth, and L^T represents the total number of layers for the global model. σ_l^2 denotes the bounded variance for the l-th layer of model.

Assumption 3. *(Second moments). The expectation of the squared l_2-norm of the gradients for client-side models and server-side models across each layer are bounded*

$$\mathbb{E}_{\xi_n\sim\mathcal{D}_n}\left\|\nabla f_{r,n}(\mathbf{w})\right\|^2 \le \sum\nolimits_{1\le l\le L_c} G_l^2, \forall \mathbf{w}, \forall n, \tag{11}$$

$$\mathbb{E}_{\xi_n\sim\mathcal{D}_n}\left\|\nabla f_{s,n}(\mathbf{w})\right\|^2 \le \sum\nolimits_{L_c+1\le l\le L^T} G_l^2, \forall \mathbf{w}, \forall n, \tag{12}$$

where G_l^2 is the second moments for l-th layer of model.

B. Convergence Proof Outline

The proof is structured as follows: We begin by leveraging the property of L-smooth functions to establish a bound on the evolution of the cloud model parameter $\mathbf{x}^k$, which depends on three terms, i.e., $\mathbb{E}\left[\langle\nabla f(\mathbf{x}^k), \mathbf{x}^{k+1}-\mathbf{x}^k\rangle\right]$, $\mathbb{E}\left[\|\mathbf{x}_r^{k+1}-\mathbf{x}_r^k\|^2\right]$, and $\mathbb{E}\left[\|\mathbf{x}_s^{k+1}-\mathbf{x}_s^k\|^2\right]$, as detailed in Lemma 1. In Lemmas 2 to 4, we derive the upper bounds for each of the three terms, respectively, and characterize how they relate to the model split depth L_c and the aggregation parameters τ_1 and τ_2.

Lemma 1. *(One Round of Cloud Aggregation). With Assumption 1, the following relationship between $\mathbf{x}^{k+1}$ and $\mathbf{x}^k$:*

$$\begin{aligned}\mathbb{E}\left[f(\mathbf{x}^{k+1})\right] &\leq \mathbb{E}\left[f(\mathbf{x}^k)\right] + \mathbb{E}\left[\langle\nabla f(\mathbf{x}^k), \mathbf{x}^{k+1}-\mathbf{x}^k\rangle\right] \\ &+\frac{L}{2}\mathbb{E}\left[\|\mathbf{x}_r^{k+1}-\mathbf{x}_r^k\|^2\right]+\frac{L}{2}\mathbb{E}\left[\|\mathbf{x}_s^{k+1}-\mathbf{x}_s^k\|^2\right].\end{aligned} \tag{13}$$

Lemma 1 follows from the property of the L-smoothness of the loss function $f(x)$. We next bound the three terms on the right hand side of Eq. (13).

Lemma 2. *With Assumptions 1 to 3, $\mathbb{E}\left[\|\mathbf{x}_r^{k+1}-\mathbf{x}_r^k\|^2\right]$ is bounded as follows:*

$$\begin{aligned}&\mathbb{E}\|\mathbf{x}_r^{k+1}-\mathbf{x}_r^k\|^2 \\ &\leq \frac{\eta^2}{N}\sum_{i=1}^{N}\tau_1\tau_2\sum_{\alpha=0}^{\tau_2-1}\sum_{\beta=0}^{\tau_1-1}\mathbb{E}\|\nabla f_{r,i}\left(\mathbf{w}_{r,i}^{k,\alpha,\beta}\right)\|^2+\eta^2\tau_1^2\tau_2^2\sum_{l=1}^{L_c}\sigma_l^2.\end{aligned} \tag{14}$$

Through Eq. (14), we can observe that the upper bound of this term is related to the number of layers on the client-side.

Lemma 3. *With Assumptions 1 to 3, $\mathbb{E}\left[\|\mathbf{x}_s^{k+1}-\mathbf{x}_s^k\|^2\right]$ is bounded as follows:*

$$\begin{aligned}&\mathbb{E}\|\mathbf{x}_s^{k+1}-\mathbf{x}_s^k\|^2 \\ &\leq \frac{\eta^2}{N}\sum_{i=1}^{N}\tau_1\tau_2\sum_{\alpha=0}^{\tau_1\tau_2-1}\mathbb{E}\|\nabla f_{s,i}\left(\mathbf{w}_{s,i}^{k,\alpha}\right)\|^2+\eta^2\tau_1^2\tau_2^2\sum_{l=L_c+1}^{L^T}\sigma_l^2.\end{aligned} \tag{15}$$

Through Eq. (15), it is shown that the upper bound of this term is dependent on the number of layers on the server-side.

Lemma 4. *With Assumptions 1 to 3, $\mathbb{E}\left[\langle\nabla f(\mathbf{x}^k), \mathbf{x}^{k+1}-\mathbf{x}^k\rangle\right]$ is bounded as follows:*

$$\begin{aligned}&\mathbb{E}\left[\langle\nabla f(\mathbf{x}^k), \mathbf{x}^{k+1}-\mathbf{x}^k\rangle\right] \\ &\leq -\frac{\eta\tau_1\tau_2}{2}\mathbb{E}\|\nabla f(\mathbf{x}^k)\|^2-\frac{\eta\tau_1\tau_2}{2}\mathbb{E}\|\frac{1}{N}\sum_{j=1}^{N}\nabla f_j(\mathbf{w}_j^{k,\alpha,\beta})\|^2 \\ &\quad+\frac{L^2\eta^3}{2}\tau_1\tau_2 C\left(\tau_1,\tau_2\right)\sum\nolimits_{l=1}^{L^T}\left(G_l^2+\sigma_l^2\right) \\ &\quad+\frac{L^2\eta^3}{2}\tau_1\tau_2 D\left(\tau_1,\tau_2\right)\sum\nolimits_{l=1}^{L_c}\left(G_l^2+\sigma_l^2\right),\end{aligned} \tag{16}$$

where $C\left(\tau_1,\tau_2\right)$ and $D\left(\tau_1,\tau_2\right)$ are functions of τ_1 and τ_2,

$$\begin{aligned}C\left(\tau_1,\tau_2\right) &= \frac{\tau_1^2\left(\tau_2-1\right)\left(2\tau_2-1\right)}{6} \\ &+\frac{\left(\tau_1-1\right)\left(2\tau_1-1\right)}{6}+\frac{\tau_1\left(\tau_2-1\right)\left(\tau_1-1\right)}{2},\end{aligned} \tag{17}$$

$$\begin{aligned}D\left(\tau_1,\tau_2\right) &= \tau_1^2\left(\tau_2-1\right)\left(2\tau_2-1\right)/6 \\ &+\frac{\left(\tau_1-1\right)\left(2\tau_1-1\right)}{6}-\frac{\tau_1\left(\tau_2-1\right)\left(\tau_1-1\right)}{2}.\end{aligned} \tag{18}$$

By combining Lemmas 1 to 4, we now have the following theorem.

Theorem 1. *(Convergence of HierSFL for Non-Convex Loss Functions). The mean square gradient of cloud model after K aggregation rounds is bounded:*

$$\begin{aligned}&\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\|\nabla f(\mathbf{x}^k)\|^2 \leq \frac{2}{\eta\tau_1\tau_2K}\left(f(\mathbf{x}^0)-f^*\right)-\sum_{l=1}^{L^T}G_l^2 \\ &+L^2\eta^2C\left(\tau_1,\tau_2\right)\sum_{l=1}^{L^T}\left(G_l^2+\sigma_l^2\right)+L\eta\tau_1\tau_2\sum_{l=1}^{L^T}\left(G_l^2+\sigma_l^2\right) \\ &+L^2\eta^2D\left(\tau_1,\tau_2\right)\sum_{l=1}^{L_c}\left(G_l^2+\sigma_l^2\right),\end{aligned} \tag{19}$$

where f^* represents the minimum value of problem (1).

Theorem 1 reveals that when the cut layer is shallower, i.e., when L_c is smaller, the convergence bound is lower. This observation aligns with the experimental results illustrated in Fig. 2.

Corollary 1. *When both the server-side model and client-side model perform aggregation at the edge server every τ_1 training iterations, the mean square gradient of cloud model after K aggregation rounds is bounded:*

$$\begin{aligned}&\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\|\nabla f(\mathbf{x}^k)\|^2 \leq \frac{2}{\eta\tau_1\tau_2K}\left(f(\mathbf{x}^0)-f^*\right)-\sum_{l=1}^{L^T}G_l^2 \\ &+L^2\eta^2E\left(\tau_1,\tau_2\right)\sum_{l=1}^{L^T}\left(G_l^2+\sigma_l^2\right)+L\eta\tau_1\tau_2\sum_{l=1}^{L^T}\left(G_l^2+\sigma_l^2\right),\end{aligned} \tag{20}$$

where $E\left(\tau_1,\tau_2\right)$ is a function of τ_1 and τ_2,

$$E\left(\tau_1,\tau_2\right) = \frac{\tau_1^2\left(\tau_2-1\right)\left(2\tau_2-1\right)}{3}+\frac{\left(\tau_1-1\right)\left(2\tau_1-1\right)}{3}. \tag{21}$$

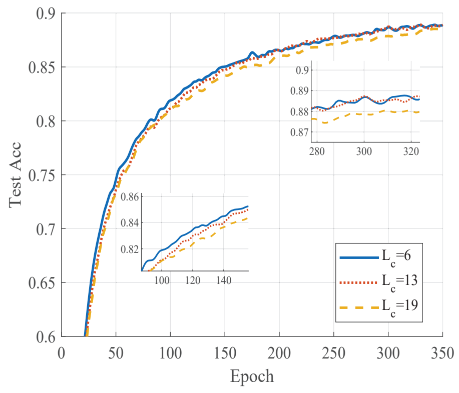
By comparing Eqs. (19) and (20), we can conclude that aggregating the server-side models at the edge server after each training iteration leads to better convergence performance. This finding is further corroborated by the experimental result depicted in Fig. 3.

IV. Simulation Results

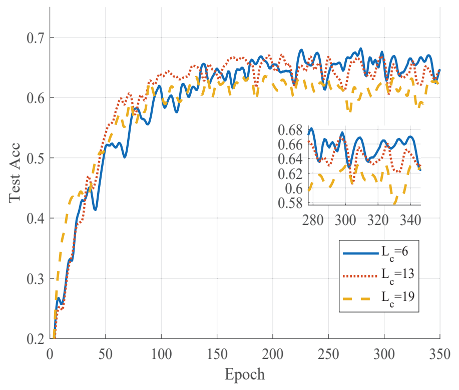
In this work, we provide simulation results for HierSFL to validate the findings from the above convergence analysis and demonstrate the superiority of the HierSFL system.

A. Simulation settings

In this simulation, we consider a HierSFL system with 20 clients, 4 edge servers and a cloud, assuming that each edge server serves 5 clients, each with the same amount of training data. Additionally, we set the client-side model edge aggregation frequency as $\tau_1 = 10$, and the cloud aggregation frequency as $\tau_2 = 3$. While the proposed algorithm and the



(a) CIFAR-10 under IID setting.



(b) CIFAR-10 under non-IID setting.

Fig. 2. CIFAR-10 dataset under IID and non-IID setting with different model split depth.

convergence analysis are developed for the IID data case, we also evaluate its effectiveness under the non-IID setting. To replicate non-IID data distribution, the data shards are distributed among multiple clients. Each client is allocated two specific classes from the dataset, and the data is partitioned and assigned through a random selection process.

As for the learning task, we execute the HierSFL framework on the CIFAR-10 dataset. Moreover, we use the Convolutional Neural Network (CNN) with 3 convolutional blocks as in [10]. The local training employs SGD with a batch size B_a of 16. More specifically, this model consists of a total of 26 layers. The learning rates of the client-side model and the server-side model are consistent. The learning rate is initialized as 0.004 and then set as 0.002, and 0.001 at the 400-th, and 800-th round.

B. Effect of Model Split Depth

Model split depth L_c, as the core parameter in the proposed algorithm, influences the convergence performance of the system. We explore the impact of parameter L_c on the convergence of the proposed HierSFL framework. It should be noted that in the simulation experiments, the global model is split after each convolutional block, i.e., $L_c = 6, 13, 19$.

As shown in Fig. 2, the test accuracy decrease with the deepening of model split point L_c, consistent with the derived convergence bound in Eq. (19). This is because deeper model

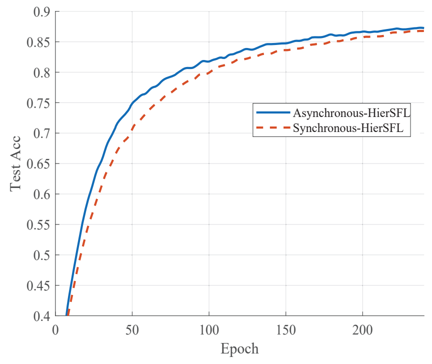


Fig. 3. Comparison of different server-side model aggregation frequency.

split points result in the smaller portion of the entire model being aggregated per training iteration, thus leading to a lower convergence performance. Furthermore, the impact of model split depth on convergence performance is more significant under non-IID than IID settings. This can be attributed to the fact that, in the non-IID setting, the variances in local dataset distributions have a greater impact on the convergence bound, as demonstrated in Eq. (19).

C. Comparison of Different Server-side Model Aggregation Frequency

In this part, we aim to validate the finding presented in Corollary 1. Asynchronous-HierSFL refers to the training mechanism introduced in this work. In this mechanism, the client-side model is aggregated at the edge server every τ_1 training iterations, while the server-side model is aggregated at the edge server after every training iteration. On the other hand, Synchronous-HierSFL represents the training mechanism where both the client-side and server-side models are aggregated at the edge server every τ_1 training iterations. As shown in Fig. 3, Asynchronous-HierSFL is observed to perform better than Synchronous-HierSFL, consistent with the derived convergence bounds in Eqs. (19) and (20). This performance enhancement is attributed to the more frequent aggregation of the server-side model, which allows the edge server to better utilize the information from the corresponding clients, thereby enhancing the training performance of HierSFL.

D. Comparison of Different Training Methods under IoT Wireless Network

To demonstrate the benefits of the HierSFL system, we compare the proposed algorithm with HierFL [10] and analyze the impact of different model split points under a wireless network. The clients upload the client-side models to the edge server through a wireless channel with a bandwidth of B_1, which is equal to 5 kHz. The channel gain is denoted as h_1. The edge servers upload the entire model to the cloud through a wireless channel with a bandwidth of B_2, which is equal to 0.8 MHz. The channel gain is denoted as h_2. The client transmitter power, p_c, is fixed at 0.1 W. Moreover, the transmitter powers from the edge server to the client, denoted

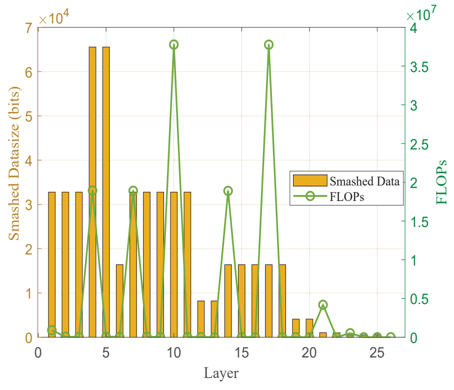


Fig. 4. Smashed datasize and FLOPs of CNN.

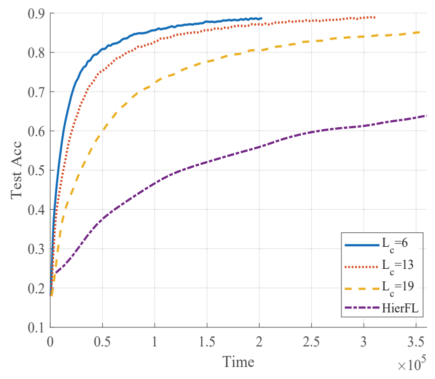


Fig. 5. Test accuracy of different training methods.

as p_{e1}, and from the edge server to the cloud, denoted as p_{e2}, are fixed at 0.5 W and 1.2 W, respectively. For the computation model, we consider that the client's CPU cycle frequency, f_c, is 0.05 GHz, while the edge server's CPU cycle frequency, f_e, is 0.2 GHz. Additionally, the FLOPs per clock cycle of the client, denoted as ϕ_c, and for the edge server, denoted as ϕ_e, are 16 and 32, respectively. Notably, we conduct a statistical analysis to obtain the smashed datasize, q_l, and FLOPs, s_l, for each layer of the entire model, as demonstrated in Fig. 4. Also, the uploaded client-side model size, D_c, and the entire model size, D, can be obtained. In this scenario, the latencies for client-side model upload, denoted as T_1, entire model upload, denoted as T_2, and a single training iteration, denoted as T_3, can be calculated as follows:

$$T_1 = \frac{B_a D_c}{B_1 \log_2\left(1 + \frac{h_1 p_c}{N_0}\right)}, \tag{22}$$

$$T_2 = \frac{B_a D}{B_2 \log_2\left(1 + \frac{h_2 p_{e2}}{N_0}\right)}, \tag{23}$$

$$T_3 = B_a\left(\frac{\sum_{l=1}^{L_c} 3s_l}{\phi_c f_c} + \frac{\sum_{l=L_c+1}^{L^T} 3s_l}{\phi_e f_e}\right). \tag{24}$$

The result shown in Fig. 5 is based on the aforementioned wireless environment. It can be observed that HierSFL exhibits significant advantages over HierFL, achieving higher accuracy in a shorter time. Simultaneously, the shallower the partitioning point, the higher the accuracy. Additionally, the partitioning point also affects the total training time. Thus, we tend to prefer selecting partitioning points that are shallower and have smaller smashed datasize.

V. Conclusion

In this paper, we investigated a client-edge-cloud HierSFL architecture that enables different aggregation frequencies between client-side and server-side sub-models. We conducted a convergence analysis of the HierSFL algorithm, offering insights into the optimal selection of model partitioning points. Our experimental results demonstrated that the adoption of different aggregation frequencies can improve the test accuracy, and the HierSFL algorithm surpasses traditional hierarchical FL algorithm in training performance.

VI. Acknowledgment

This work of Xiangwang Hou was supported by the National Natural Science Foundation of China under grant No. 623B2060. This work of Dusit Niyato was supported by the National Research Foundation, Singapore, and Infocomm Media Development Authority under its Future Communications Research & Development Programme, Defence Science Organisation (DSO) National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-RP-2020-019 and FCP-ASTAR-TG-2022-003), Singapore Ministry of Education (MOE) Tier 1 (RG87/22), and the NTU Centre for Computational Technologies in Finance (NTU-CCTF).

References

[1] L. U. Khan, M. Guizani, A. Al-Fuqaha, C. S. Hong, D. Niyato, and Z. Han, "A joint communication and learning framework for hierarchical split federated learning," *IEEE Internet Things J.*, vol. 11, no. 1, pp. 268–282, Jan. 2024.

[2] C. Xu, J. Li, Y. Liu, Y. Ling, and M. Wen, "Accelerating split federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, Early Access 2023.

[3] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," *arXiv preprint arXiv:1812.00564*, 2018.

[4] X. Feng, C. Luo, J. Chen, Y. Huang, J. Zhang, W. Xu, J. Li, and V. C. Leung, "Iotsl: Towards efficient distributed learning for resource-constrained internet of things," *IEEE Internet Things J.*, vol. 10, no. 11, pp. 9892–9905, Jun. 2023.

[5] X. Liu, Y. Deng, and T. Mahmoodi, "Wireless distributed learning: A new hybrid split and federated learning approach," *IEEE Trans. Wireless Commun.*, vol. 22, no. 4, pp. 2650–2665, Apr. 2023.

[6] W. Wu, M. Li, K. Qu, C. Zhou, X. Shen, W. Zhuang, X. Li, and W. Shi, "Split learning over wireless networks: Parallel design and resource management," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 1051–1066, Apr. 2023.

[7] C. Thapa, P. C. M. Arachchige, S. Camtepe, and L. Sun, "Splitfed: When federated learning meets split learning," in *Proc. AAAI Conf. on Artif. Intell.*, vol. 36, no. 8, 2022, pp. 8485–8493.

[8] L. Zhao, S. Ni, D. Wu, and L. Zhou, "Cloud-edge-client collaborative learning in digital twin empowered mobile networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 11, pp. 3491–3503, Nov. 2023.

[9] R. Deng, X. Du, Z. Lu, Q. Duan, S.-C. Huang, and J. Wu, "Hsfl: Efficient and privacy-preserving offloading for split and federated learning in iot services," in *IEEE Intl. Conf. on Web Ser. (ICWS'23)*, Chicago, IL, USA, 2-8 Jul. 2023.

[10] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Hierarchical federated learning with quantization: Convergence analysis and system design," *IEEE Trans. Wireless Commun.*, vol. 22, no. 1, pp. 2–18, Jan. 2022.

[11] Y. Liao, Y. Xu, H. Xu, Z. Yao, L. Wang, and C. Qiao, "Accelerating federated learning with data and model parallelism in edge computing," *IEEE/ACM Trans. Netw.*, vol. 32, no. 1, pp. 904–918, Feb. 2024.