Straggler-Resilient Hierarchical Split Federated Learning via Smashed Data Generation

Chunfeng Xie, Zhixiong Chen, *Member, IEEE*, Wenqiang Yi, *Member, IEEE*, Hong Xing, *Member, IEEE*, Chan-Tong Lam, *Senior Member, IEEE*, and Arumugam Nallanathan, *Fellow, IEEE*

***Abstract*—Large-scale edge intelligence requires collaborative training across resource-constrained and heterogeneous connected devices without uploading raw data. Split federated learning reduces the computation burden on devices by partitioning a model between clients and edge servers, while hierarchical deployment improves the scalability of distributed training. However, existing hierarchical split federated learning (HSFL) designs are still sensitive to stragglers and non-IID data, since delayed clients reduce training efficiency and their missing or stale smashed data may bias server-side updates. This paper proposes SAGE-HSFL, a smashed data generation-enabled semi-asynchronous hierarchical split federated learning framework. In SAGE-HSFL, each edge server maintains a labeled smashed-data pool, learns online distributions from the received cut layer representations, and generates supplementary smashed data to compensate for underrepresented or delayed client updates. This enables server-side training to proceed with reduced gradient dissimilarity while preserving the raw-data locality of split learning. We establish a convergence bound for general non-convex objectives and reveal how smashed data generation mitigates the impact of stragglers on update bias. Based on the staleness-dependent term in the convergence bound, we formulate an online joint optimization of client scheduling, cut layer selection, bandwidth allocation, and edge association, and solve it through Lyapunov-based decomposition and adaptive association. Experiments on FMNIST and CIFAR-10 show that SAGE-HSFL improves test accuracy and reduces gradient dissimilarity. SAGE yields 0.79%–2.06% accuracy gains over its disabled variant, while the complete framework achieves 2.42%–4.53% higher accuracy than representative external baselines. The complete framework further achieves up to 10.21% and 14.17% higher accuracy than external baselines under the same target time on FMNIST and CIFAR-10, respectively, and reaches the target accuracy 6.20 and 7.02 times faster than a basic HSFL ablation.**

***Index Terms*—Hierarchical split federated learning, semi-asynchronous, smashed data generating, resource allocation**

I. Introduction

Global data generation is increasingly moving to the network edge, where billions of mobile and IoT devices produce high-rate data streams that are costly or impractical to upload in raw form [2]. According to an International Data Corporation report, global data creation reached 159.2 zettabytes (ZB) in 2024 and is projected to grow to 527.5 ZB by 2029 [3]. To support on-device learning under this trend, federated learning (FL) [4] has become a widely used paradigm: clients train models locally in parallel and periodically send updates to a parameter server for aggregation. However, model sizes continue to increase. For example, Gemma 2-2B [5], a large language model designed for on-device use, has 2 billion parameters, which is about 2 GB under 8-bit precision, while device resources remain limited. As a result, the computation and communication costs of conventional FL become prohibitive [6]. Split federated learning (SFL) [7] addresses this challenge by shifting part of the training workload from clients to an edge server. By combining ideas from FL and split learning (SL) [8], SFL partitions the model into client-side and server-side components, trains them collaboratively, and periodically aggregates sub-models following the FL principle.

Despite its advantages, deploying SFL in wireless networks faces three main challenges: (i) Straggler effect: In practical wireless systems, clients differ in computing speed and transmit power. Under synchronous training, fast clients must wait for slow ones, which wastes time and reduces efficiency [9]; (ii) Data heterogeneity: Client data are often non-independent and identically distributed (non-IID). This statistical mismatch can cause biased local updates and slow or unstable convergence [10], [11]; (iii) Limited wireless resources: Bandwidth and transmission opportunities are constrained, which limits the number of clients that can participate in each round and may degrade learning performance [12]. These challenges motivate the development of a new SFL framework tailored to wireless edge networks.

Most existing studies focus on two-tier wireless SFL, where a single edge server coordinates multiple clients. However, because one edge server has limited wireless and computing resources, this architecture scales poorly as the number of clients increases. To improve scalability, several recent works [13]–[15] extend SFL to a multi-tier setting, known as hierarchical split federated learning (HSFL). Specifically, [13] proposes an HSFL framework that combines client assignment with adaptive cut layer selection to coordinate clients, edge servers, and the cloud, aiming to reduce aggregation latency and communication overhead. In [14], the authors develop a hierarchical split FL scheme with decentralized aggregation to avoid a single point of failure, and jointly optimize association, spectrum allocation, computation offloading, and transmit power under a unified objective. Moreover, [15] provides tier-aware convergence guarantees under practical communication and computation constraints, models the latency-accuracy trade-off, and jointly optimizes cut layer placement and aggregation timing via block coordinate descent to minimize training time for a target accuracy. While these studies improve scalability, they still do not effectively address the straggler effect or data heterogeneity.

To reduce the straggler effect in synchronous training, many studies adopt asynchronous FL [16]–[20]. Asynchronous FL allows clients to train and upload updates without waiting for

Part of this work was presented in IEEE PIMRC, 2026 [1].

global rounds, which avoids straggler delays and improves resource utilization. For example, [16] is an early work that adaptively weights local updates to reduce the impact of stragglers. To further increase concurrency, [17] introduces a buffering mechanism that temporarily stores updates from fast clients. In [18], the authors define an effectivity score that captures the learning gain per wireless cost and use it to schedule transmissions under uncertain channels, prioritizing more valuable updates. The work in [19] studies asynchronous FL over wireless networks and proposes a freshness- and data-aware aggregation rule to cope with unreliable links and heterogeneous staleness. Moreover, [20] proposes channel-aware scheduling and freshness-weighted aggregation to prioritize useful updates and reduce model drift under time-varying connectivity. While these approaches effectively alleviate stragglers, asynchronous FL can still suffer from stronger negative effects of stale updates, especially in non-IID settings, where delayed gradients may amplify bias and slow convergence.

To reduce the negative impact of stale updates in asynchronous FL, semi-asynchronous FL has been proposed to balance training latency and model accuracy. For example, [21] presents a semi-asynchronous protocol that controls stragglers and staleness through selective participation and timely aggregation, enabling faster learning with low communication overhead. The method in [22] schedules clients based on their capabilities, uses soft deadlines to limit staleness, and aggregates updates promptly to accelerate training while maintaining convergence. In [23], the authors develop an adaptive asynchronous FL framework for resource-limited edge computing, where client participation, staleness weighting, and communication budgets are dynamically adjusted to handle heterogeneity and intermittent connectivity. Similarly, [24] proposes a semi-asynchronous scheme that adapts client scheduling and freshness-weighted aggregation to wireless conditions and update staleness. Although these semi-asynchronous approaches [21]–[24] alleviate stragglers and stale gradients, they still struggle with data heterogeneity, which can bias global updates and degrade convergence. Moreover, their two-tier architectures are difficult to scale when the number of users and the model size increase.

The challenges of stragglers and resource scarcity motivate a semi-asynchronous HSFL framework tailored to wireless edge networks. To this end, we propose a smashed data generating-enabled (SAGE) mechanism. SAGE stores the smashed data received at the edge server in a smashed-data pool and generates new smashed data according to the observed data distribution. This design helps mitigate both the straggler effect and data heterogeneity. Different from existing HSFL studies [13]–[15], which still suffer from stragglers, and prior asynchronous or semi-asynchronous FL works [16]–[24], which remain vulnerable to data heterogeneity and wireless resource limits, we develop a new framework called SAGE-enabled semi-asynchronous hierarchical split federated learning (SAGE-HSFL) to address these issues jointly. We further provide a convergence analysis to characterize the effect of SAGE on server-side gradient dissimilarity and the staleness-dependent error term. Based on the staleness-dependent term in the derived upper bound, we formulate a joint optimization problem over edge association, client scheduling, cut layer selection, and bandwidth allocation, and design corresponding solution methods to improve learning efficiency in wireless SAGE-HSFL. The key contributions are summarized as follows:

- We propose SAGE-HSFL, a three-tier wireless SFL framework that enables semi-asynchronous transmission of smashed data and client-side model updates. We further design SAGE, where each edge server estimates the label-wise distribution from received smashed data, maintains a smashed data pool, and generates smashed data from this distribution to support submodel updates, thereby reducing update bias caused by stragglers. To the best of our knowledge, this is the first multi-tier SFL study that jointly investigates the straggler effect and biased updates to improve learning performance.
- We provide a theoretical analysis under general non-convex objectives to study how SAGE affects server-side gradient dissimilarity. The analysis characterizes a sufficient condition under which SAGE reduces the server-side dissimilarity term. Based on the staleness-dependent term in the convergence bound, we use the average staleness of local gradients as a tractable optimization target and reformulate the original global loss minimization into an explicit joint design of client scheduling, cut layer decision, and bandwidth allocation for a single server.
- To solve the reformulated problem and support online client scheduling, we convert it into a deterministic form using the Lyapunov optimization framework. We then develop algorithms for bandwidth allocation, cut layer decision, and dynamic client scheduling to derive effective bandwidth and cut layer policies. Finally, we propose an adaptive edge association algorithm that assigns clients to edge servers based on utility, further improving overall system performance.
- We conduct extensive simulations on FMNIST and CIFAR-10 under two non-IID data settings to evaluate the proposed approach. The results show that SAGE improves test accuracy and reduces gradient dissimilarity compared with the baselines. In addition, SAGE-HSFL achieves higher time efficiency and better accuracy than representative baselines and ablation variants.

The remainder of this paper is organized as follows. Section II introduces the SAGE-HSFL framework and the wireless system model, and formulates the optimization problem. Section III presents the convergence analysis. Section IV reformulates the problem into a tractable form and develops algorithms to solve it. Section V reports experimental results that demonstrate the effectiveness of the proposed approach. Finally, Section VI concludes the paper and summarizes the main findings.

II. System Model and Learning Mechanism

In this section, we introduce the SAGE-HSFL framework for a resource-limited wireless network. To reduce the gradient dissimilarity that degrades HSFL performance, we propose SAGE. We then quantify the learning costs of the proposed

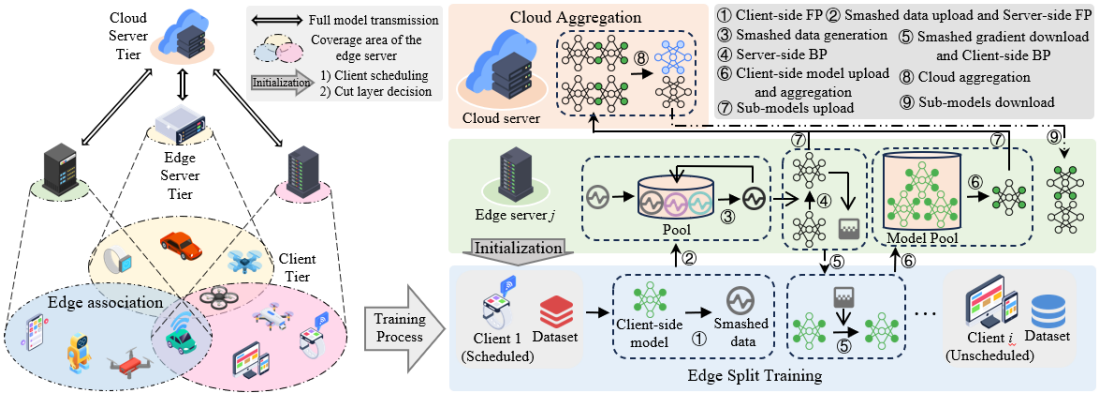


Fig. 1. Illustration of the proposed wireless SAGE-HSFL system.

framework and formulate an optimization problem that minimizes the global loss function.

A. Hierarchical Split Federated Learning System

We consider a three-tier HSFL wireless system with I clients in the bottom tier, indexed by $\mathcal{I} = \{1, \cdots, i, \cdots, I\}$, J edge servers in the middle tier, indexed by $\mathcal{J} = \{1, \cdots, j, \cdots, J\}$, and one cloud server in the top tier. Each client i ($i \in \mathcal{I}$) has a local dataset $\mathcal{D}_i$ containing $D_i = |\mathcal{D}_i|$ samples. We assume that local datasets are disjoint across clients. The overall dataset is $\mathcal{D} = \bigcup\{\mathcal{D}_i\}_{i=1}^{I}$ with total size $D = \sum_{i=1}^{I} D_i$. For a sample $(\boldsymbol{x}, y) \in \mathcal{D}$, $\boldsymbol{x} \in \mathbb{R}^O$ denotes the O-dimensional input vector and $y \in \mathbb{R}$ is the corresponding label. In HSFL, the global model $\boldsymbol{w}$ is partitioned into two parts. The client-side sub-model of client i associated with edge server j is denoted by $\boldsymbol{u}_i^j$, and the corresponding server-side sub-model on edge server j is denoted by $\boldsymbol{v}_i^j$. The resulting model for client i at server j is $\boldsymbol{w}_i^j = [\boldsymbol{u}_i^j; \boldsymbol{v}_i^j]$. The empirical loss of client i measures the error of $\boldsymbol{w}$ on a sampled mini-batch $\mathcal{B}_i$ and is given by

$$f(\boldsymbol{w}; \mathcal{B}_i) = h(\boldsymbol{v}; k(\boldsymbol{u}; \mathcal{B}_i)), \tag{1}$$

where $k(\boldsymbol{u}; \mathcal{B}_i)$ denotes the client-side mapping from the mini-batch $\mathcal{B}_i$ to the cut layer activations. The function $h(\boldsymbol{v}; k(\boldsymbol{u}; \mathcal{B}_i))$ denotes the server-side mapping that takes the smashed data as input and outputs the corresponding loss value. Therefore, the expected local loss of client i, which evaluates the model error over its local dataset, is defined as

$$F_i(\boldsymbol{w}) = \frac{1}{D_i} \sum\nolimits_{\mathcal{B}_i \subseteq \mathcal{D}_i} f(\boldsymbol{w}; \mathcal{B}_i). \tag{2}$$

The goal of the HSFL system is to learn a global model $\boldsymbol{w}$ by minimizing the global loss $F(\boldsymbol{w})$ over the entire dataset. The resulting finite-sum non-convex global loss over all scheduled clients is given by

$$\min_{\boldsymbol{w}} F(\boldsymbol{w}) = \min_{\boldsymbol{w}} \sum\nolimits_{i=1}^{I} p_i F_i(\boldsymbol{w}), \tag{3}$$

where p_i is the weight of client i, with $p_i \geq 0$ and $\sum_{i=1}^{I} p_i = 1$. We assume balanced dataset sizes and set $p_i = \frac{1}{I}$ [24].

B. The SAGE-HSFL Framework

This subsection describes the workflow of the SAGE-HSFL framework. As shown in Fig. 1, the training process includes three main stages: initialization, edge split training, and cloud aggregation. For each round t ($t \in \{0, 1, \cdots, T-1\}$), the steps in these three stages are described as follows.

(i) Initialization: Before training starts, two steps are performed.

1) ***Client Scheduling***: Let $\mathcal{U}_t^j \subseteq \boldsymbol{\mathcal{U}}_t$ denote the set of available clients associated with edge server j, where $\boldsymbol{\mathcal{U}}_t = \{\mathcal{U}_t^1, \mathcal{U}_t^2, \cdots, \mathcal{U}_t^J\}$. Let $\alpha_{i,t}^j \in \{0, 1\}$ denote the scheduling indicator of client i at edge server j. Here, $\alpha_{i,t}^j = 1$ means client i is selected by edge server j in round t, and $\alpha_{i,t}^j = 0$ otherwise. Let $\boldsymbol{\mathcal{M}}_t = \{\mathcal{M}_t^1, \mathcal{M}_t^2, \cdots, \mathcal{M}_t^J\}$ denote the scheduling decisions for all edge servers in round t, where $\mathcal{M}_t^j = \{i \mid \alpha_{i,t}^j = 1, \forall i \in \mathcal{I}\}$ is the set of scheduled clients for edge server j. We have $\mathcal{M}_t^j \subseteq \mathcal{U}_t^j$.

2) ***Cut Layer Decision***: Let $s_t^j \in \mathcal{S} = \{1, 2, \cdots, S\}$ denote the cut layer for clients associated with edge server j in round t, where S is the total number of model layers. Let $\boldsymbol{s}_t = \{s_t^1, s_t^2, \cdots, s_t^J\}$ denote the cut layer decisions for all edge servers in round t.

(ii) Edge Split Training: During this stage, forward propagation (FP) and backward propagation (BP) are performed at both the clients and the edge servers. For clarity, we divide this stage into eight steps and describe each step below.

1) ***Client-side FP***: In each local iteration l, where $l \in \{0, 1, \cdots, L-1\}$, each scheduled client i randomly samples a mini-batch $\mathcal{B}_{i,t,l} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_b, y_b)\}$ of size b from its local dataset $\mathcal{D}_i$ and performs FP. The input set is $\boldsymbol{X}_i = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_b\}$ and the corresponding labels are $\boldsymbol{Y}_i = \{y_1, y_2, \cdots, y_b\}$. After client-side FP, the smashed data at the cut layer is

$$\boldsymbol{A}_{i,t,l} = k(\boldsymbol{u}_{i,t,l}^j; \mathcal{B}_{i,t,l}), \forall i \in \mathcal{I}. \tag{4}$$

2) ***Smashed Data Upload and Server-side FP***: After computation, client i uploads the smashed data $\boldsymbol{A}_{i,t,l}$ and the label set $\boldsymbol{Y}_{i,t,l}$ to its edge server over a wireless channel. Each edge server maintains a smashed-data pool Θ^j to store received smashed data with labels, and a client-side model pool $\mathcal{C}^j$ to store received client-side models together with the corresponding local data sizes. The capacities of Θ^j and $\mathcal{C}^j$ are set to $b\lambda_a^j$ and λ_c^j, respectively. After receiving data from client i, the edge server stores it in Θ^j as

$$\Theta^j \leftarrow \Theta^j \cup (\boldsymbol{A}_{i,t,l}, \boldsymbol{Y}_{i,t,l}). \tag{5}$$

The edge server then collects the smashed data from its associated clients and uses them to perform server-side FP.

3) ***Smashed Data Generating***: Each edge server maintains a smashed-data distribution for each label and updates it using

the received smashed data. When the number of smashed data samples in the pool exceeds its capacity, the edge server generates smashed data $\tilde{\boldsymbol{A}}$ from the learned distributions. The generated smashed data are then concatenated with the pooled smashed data to form $\mathcal{A}^j = \mathcal{A}^j \oplus \tilde{\boldsymbol{A}}$. The corresponding labels $\tilde{\boldsymbol{Y}}$ are concatenated in the same way $\mathcal{Y}^j = \mathcal{Y}^j \oplus \tilde{\boldsymbol{Y}}$.

4) ***Server-side BP***: Using stochastic gradient descent (SGD), the server-side sub-model is updated with the smashed data array as

$$\boldsymbol{v}_{t,l+1}^j = \boldsymbol{v}_{t,l}^j - \eta_v \nabla h(\boldsymbol{v}_{t,l}^j; (\mathcal{A}^j, \mathcal{Y}^j)), \tag{6}$$

where η_v is the learning rate for the server-side update. The parameters are updated layer by layer from the final layer to the cut layer, following the chain rule. The edge server also computes the smashed-data gradients for client i using the received smashed data, given by

$$\boldsymbol{g}_{i,t,l} = \nabla h_i(\boldsymbol{v}_{i,t,l}^j; (\boldsymbol{A}_{i,t,l}, \boldsymbol{Y}_{i,t,l})), \tag{7}$$

where h_i is the loss function for client i.

5) ***Smashed Gradient Download and Client-side BP***: After server-side BP, each edge server sends the smashed gradients to the corresponding clients. The scheduled clients then use these gradients to perform client-side BP. By the chain rule, clients update its client-side model using SGD for L steps as

$$\boldsymbol{u}_{i,t,l+1}^j = \boldsymbol{u}_{i,t,l}^j - \eta_u \boldsymbol{g}_{i,t,l} \nabla k_i(\boldsymbol{u}_{i,t,l}^j; \boldsymbol{X}_{i,t,l}), \tag{8}$$

where η_u is the learning rate for the client-side update.

6) ***Client-side Model Upload and Aggregation***: After client i finishes L local iterations, it uploads its updated client-side model to the edge server. The edge server stores the received model in the client-side model pool as $\mathcal{C}_t^j \leftarrow \mathcal{C}_t^j \cup (\boldsymbol{u}_{i,t}^j, D_i)$. When the number of stored client-side models exceeds λ_c^j, the edge server aggregates them to obtain the current client-side model $\boldsymbol{u}_{t+1}^j = \frac{\sum_{i \in \mathcal{M}_t^j} |\mathcal{D}_i| \boldsymbol{u}_{i,t}^j}{\sum_{i \in \mathcal{M}_t^j} |\mathcal{D}_i|}$. Since the update frequency of the server-side model and the aggregation frequency of the client-side model depend on the pool sizes, we typically set $\lambda_a^j = \lambda_c^j = \lambda^j$.

(iii) Cloud Aggregation: In the global aggregation stage, the cloud server aggregates the sub-models.

1) ***Sub-model Uploading***: After R rounds of client-side aggregation, edge servers upload sub-models to the cloud.

2) ***Cloud Aggregation***: Let $\mathcal{D}_j = \{\mathcal{D}_i \mid i \in \mathcal{U}_t^j\}$ denote the union of local datasets for clients associated with edge server j. The aggregated global model is

$$\boldsymbol{w}_{t+1} = \frac{\sum_{j \in \mathcal{J}} |\mathcal{D}_j| \boldsymbol{w}_t^j}{\sum_{j \in \mathcal{J}} |\mathcal{D}_j|}, \tag{9}$$

where $\boldsymbol{w}_{t+1}$ is the updated global model.

3) ***Sub-model Downloading***: After aggregation, the cloud server broadcasts the updated global model to all edge servers. This completes one cloud aggregation round.

C. Latency Model

In this subsection, we present a detailed latency analysis that accounts for both communication and computation.

1) **Communication Latency**: We consider a wireless network that uses frequency division multiple access (FDMA). For each edge server j, the scheduled clients share the total bandwidth B^j to transmit smashed data and sub-models. Let $\boldsymbol{\omega}_t = \{\boldsymbol{\omega}_t^1, \boldsymbol{\omega}_t^2, \cdots, \boldsymbol{\omega}_t^J\}$ denote the bandwidth allocation policy in round t, where $\boldsymbol{\omega}_t^j = \{\omega_{i,t}^j \mid i \in \mathcal{M}_t^j\}$ and $\omega_{i,t}^j$ is the fraction of B^j assigned to client i by edge server j in round t. Let $p_{i,t}$ be the transmit power of client i in round t, and let $h_{i,t}$ be the channel gain from client i to the edge server. The achievable uplink rate of client i in round t is

$$\mathbf{r}_{i,t}(\omega_{i,t}^j) = \omega_{i,t}^j B^j \log_2(1 + \frac{p_{i,t} h_{i,t}}{\sigma^2}), \tag{10}$$

where σ^2 is the noise power. Let γ_s denote the smashed-data size per sample at cut layer s. The latency for uploading smashed data is $t_{i,j,t}^{\mathrm{AT}} = \frac{b \sum_{s=1}^{s_t^j} \gamma_s L}{\mathbf{r}_{i,t}(\omega_{i,t}^j)}$. The latency for downloading smashed gradients is $t_{i,j,t}^{\mathrm{AD}} = \frac{|\mathcal{M}_t^j| b \sum_{s=1}^{s_t^j} \gamma_s L}{\mathbf{r}_{j,t}}$, where $\mathbf{r}_{j,t} = B^j \log_2(1 + \frac{p_j h_{i,t}}{\sigma^2})$ is the downlink rate of edge server j. Let ψ_s denote the size of the client-side sub-model up to cut layer s. The latency for uploading the client-side model is $t_{i,j,t}^{\mathrm{MU}} = \frac{\sum_{s=1}^{s_t^j} \psi_s}{\mathbf{r}_{i,t}(\omega_{i,t}^j)}$, and the latency for downloading the client-side model is $t_{i,j,t}^{\mathrm{MD}} = \frac{|\mathcal{M}_t^j| \sum_{s=1}^{s_t^j} \psi_s}{\mathbf{r}_{j,t}}$. Therefore, the total communication latency of client i in round t is

$$T_{i,j,t}^{\mathrm{CM}} = t_{i,j,t}^{\mathrm{AT}} + t_{i,j,t}^{\mathrm{AD}} + t_{i,j,t}^{\mathrm{MU}} + t_{i,j,t}^{\mathrm{MD}}, \forall i \in \mathcal{I}. \tag{11}$$

Note that the communication latency between the cloud server and the edge servers is ignored since the cloud has sufficient bandwidth.

2) **Computing Latency**: The latency of client-side FP is $t_{i,j,t}^{\mathrm{CF}} = \frac{b \sum_{s=1}^{s_t^j} \zeta_s L}{f_i}$, where ζ_s is the computation cost in FLOPs for forward propagation of layer s for one sample, and f_i is the computing capability of client i. The latency of server-side FP is $t_{i,j,t}^{\mathrm{SF}} = \frac{b \sum_{i \in \mathcal{M}_t^j} \sum_{s=1}^{s_t^j} \zeta_s L}{f_j}$, where f_j is the computing capability of edge server j. For BP, let ξ_s denote the computation cost in FLOPs for backward propagation of layer s for one sample. The latency of client-side BP is $t_{i,j,t}^{\mathrm{CB}} = \frac{b \sum_{s=s_t^j+1}^{S} \xi_s L}{f_i}$, and the latency of server-side BP is $t_{i,j,t}^{\mathrm{SB}} = \frac{b \sum_{i \in \mathcal{M}_t^j} \sum_{s=s_t^j+1}^{S} \xi_s L}{f_j}$. Therefore, the total computing latency of client i in round t is

$$T_{i,j,t}^{\mathrm{CP}} = t_{i,j,t}^{\mathrm{CF}} + t_{i,j,t}^{\mathrm{SF}} + t_{i,j,t}^{\mathrm{CB}} + t_{i,j,t}^{\mathrm{SB}}, \forall i \in \mathcal{I}. \tag{12}$$

Equations (11) and (12) retain the dominant communication and neural-network computation terms for tractable system-level optimization. The recursive moment update and diagonal-Gaussian sampling require only lightweight coordinate-wise operations at the edge server and introduce no additional communication. At each SAGE update, the generated samples increase the server-side sample workload by $b'/(b\lambda^j)$. Under the default settings $b = 32$, $\lambda^j = 10$, and $b' = 32$, this increase is 10%. The complete implementation overhead, including the additional FP and BP on generated samples, is quantified in Table IX.

Based on the latency components above, we characterize the latency of one global round at edge server j. Consider round t, where edge server j waits for the scheduled clients in $\mathcal{M}_t^j$ to finish their remaining local training and upload their client-side models. Due to the asynchronous mechanism, a client may

have already completed part of its local training before round t starts. Let $\widehat{T}^{\mathrm{CP}}_{i,j,t}$ denote the remaining computation time for client i in round t. Hence, the round latency is determined by the slowest scheduled client and is given by

$$\mathbf{T}^j_t = \max\nolimits_{i\in\mathcal{M}^j_t}\{T^{\mathrm{CM}}_{i,j,t} + \widehat{T}^{\mathrm{CP}}_{i,j,t}\}, \tag{13}$$

where $\mathbf{T}^j_t$ denotes the time cost of one global round at edge server j. The overall latency in round t is then defined as $\mathbf{T}_t = \max_{j\in\mathcal{J}}\{\mathbf{T}^j_t\}$.

D. Problem Formulation

In this work, we formulate a joint optimization problem over client scheduling, cut layer decision, and bandwidth allocation as follows:

$$\mathcal{P}: \min_{\{\boldsymbol{\mathcal{M}}_t,\boldsymbol{s}_t,\boldsymbol{\omega}_t\}_{t=0}^{T-1}} \mathbb{E}[F(\boldsymbol{w}_T)] \tag{14}$$

$$\text{s. t.} \quad \sum\nolimits_{t=0}^{T-1} \mathbf{T}^j_t \le T_{\max}, \forall j\in\mathcal{J}, \forall t, \tag{14a}$$

$$\sum\nolimits_{i\in\mathcal{M}^j_t} \omega^j_{i,t} \le 1, \forall i\in\mathcal{I}, \forall j\in\mathcal{J}, \forall t, \tag{14b}$$

$$0 \le \omega^j_{i,t} \le 1, \forall i\in\mathcal{I}, \forall j\in\mathcal{J}, \forall t, \tag{14c}$$

$$s^j_t \in \mathcal{S}, \forall j\in\mathcal{J}, \forall t, \tag{14d}$$

$$\alpha^j_{i,t} \in \{0,1\}, \forall i\in\mathcal{I}, \forall j\in\mathcal{J}, \forall t, \tag{14e}$$

where (14a) enforces the total latency constraint $T_{\max}$. (14b) and (14c) capture the bandwidth limits. (14d) specifies the cut layer decision. (14e) defines the client scheduling decision in each round. To solve problem $\mathcal{P}$, we would ideally express the global loss explicitly as a function of the client scheduling, cut layer decision, and bandwidth allocation policies. However, because training dynamics are highly complex, it is difficult to derive a closed-form expression for the expected loss $\mathbb{E}[F(\boldsymbol{w}_T)]$ in terms of $\boldsymbol{\mathcal{M}}_t$, $\boldsymbol{s}_t$, and $\boldsymbol{\omega}_t$. Therefore, in Section III, we derive an upper bound on $\mathbb{E}[F(\boldsymbol{w}_T)]$ and minimize this bound as a tractable surrogate objective. Additionally, we follow a divide-and-conquer approach similar to [25], and decompose the HSFL optimization into two subproblems. The first is edge association across multiple edge servers. The second is resource optimization within a single edge server. The edge association algorithm and the single-server solutions are presented in Section IV.

III. Smashed Data Generating Mechanism and Convergence Analysis

This section first presents the smashed data generating mechanism. It then analyzes the convergence bound of SAGE-HSFL under general non-convex objectives.

A. Smashed Data Generating Mechanism

To address the biased updating issue, we propose a SAGE method. Each edge server j maintains a label-wise smashed-data distribution, modeled as a diagonal Gaussian distribution $\mathcal{G}^j_y(\boldsymbol{\mu}_y, \mathrm{diag}(\boldsymbol{\sigma}_y))$, where $\boldsymbol{\mu}_y \in \mathbb{R}^{d_s}$ and $\boldsymbol{\sigma}_y \in \mathbb{R}^{d_s}_+$ are the coordinate-wise mean and variance of smashed data with label y at cut layer s. The diagonal form is adopted to support online updating and sampling for high-dimensional smashed data. It reduces the storage cost of distribution statistics from $O(|\mathcal{Y}|d_s^2)$ to $O(|\mathcal{Y}|d_s)$, where $|\mathcal{Y}|$ is the number of labels. We do not assume that this lightweight model exactly matches the true smashed-data distribution. Its effect on gradient dissimilarity is characterized by Lemma 2 and evaluated empirically in Section V.

The edge server updates the mean $\boldsymbol{\mu}$ and coordinate-wise variance $\boldsymbol{\sigma}$ using the asynchronously uploaded smashed data in a weighted manner. We assign larger weights to newer smashed data. Specifically, we use an increasing weight function $\mathcal{W}(e) = ke^a$, where e denotes the training progress defined by the total number of iterations $e = tL + l$. As training proceeds, older smashed data become less influential, which reduces the impact of stragglers on distribution estimation. Since smashed data arrive asynchronously, each sample has a different training progress. Let $e(\boldsymbol{A}_i)$ denote the progress index of smashed data $\boldsymbol{A}_i$. For a given progress level E, the weighted mean is $\boldsymbol{\mu}_E = \frac{1}{W_E}\sum_{\boldsymbol{A}_i\in\Theta^j_E}\mathcal{W}(e(\boldsymbol{A}_i))\boldsymbol{A}_i$ and the coordinate-wise weighted variance is $\boldsymbol{\sigma}_E = \frac{1}{W_E}\sum_{\boldsymbol{A}_i\in\Theta^j_E}\mathcal{W}(e(\boldsymbol{A}_i))(\boldsymbol{A}_i-\boldsymbol{\mu}_E)\odot(\boldsymbol{A}_i-\boldsymbol{\mu}_E)$, where $\odot$ denotes the Hadamard product, $W_E = \sum_{\boldsymbol{A}_i\in\Theta^j_E}\mathcal{W}(e(\boldsymbol{A}_i))$ is the total weight and Θ^j_E is the set of smashed data uploaded to edge server j up to progress E. The resulting update rules are given in the following lemma:

Lemma 1. *Since smashed data arrive dynamically, the mean and coordinate-wise variance are updated online. Given a newly received smashed data sample $\boldsymbol{A}_i$, let $w_i = \mathcal{W}(e(\boldsymbol{A}_i))$ and $W_E = W_{E-1} + w_i$. The mean is updated as $\boldsymbol{\mu}_E = \frac{W_{E-1}}{W_E}\boldsymbol{\mu}_{E-1} + \frac{w_i}{W_E}\boldsymbol{A}_i$. The coordinate-wise variance is updated as $\boldsymbol{\sigma}_E = \frac{W_{E-1}}{W_E}(\boldsymbol{\sigma}_{E-1} + (\boldsymbol{\mu}_E-\boldsymbol{\mu}_{E-1})\odot(\boldsymbol{\mu}_E-\boldsymbol{\mu}_{E-1})) + \frac{w_i}{W_E}(\boldsymbol{A}_i-\boldsymbol{\mu}_E)\odot(\boldsymbol{A}_i-\boldsymbol{\mu}_E)$.*

Proof: Consider a newly received smashed data sample $\boldsymbol{A}$. The weighted mean at training progress E is $\boldsymbol{\mu}_E = \frac{1}{W_E}\sum_{\boldsymbol{A}'\in\Theta^j_E}\mathcal{W}(e(\boldsymbol{A}'))\boldsymbol{A}' = \frac{W_{E-1}}{W_E}\boldsymbol{\mu}_{E-1} + \frac{\mathcal{W}(e(\boldsymbol{A}))}{W_E}\boldsymbol{A}$. The coordinate-wise weighted variance at training progress E is

$$\begin{aligned}\boldsymbol{\sigma}_E = \frac{1}{W_E}\Bigg(&\sum\nolimits_{\boldsymbol{A}'\in\Theta^j_{E-1}}\mathcal{W}(e(\boldsymbol{A}'))(\boldsymbol{A}'-\boldsymbol{\mu}_E)\odot(\boldsymbol{A}'-\boldsymbol{\mu}_E)\\ &+\mathcal{W}(e(\boldsymbol{A}))(\boldsymbol{A}-\boldsymbol{\mu}_E)\odot(\boldsymbol{A}-\boldsymbol{\mu}_E)\Bigg).\end{aligned} \tag{15}$$

For the first summation term, we have

$$\begin{aligned}&\sum\nolimits_{\boldsymbol{A}'\in\Theta^j_{E-1}}\mathcal{W}(e(\boldsymbol{A}'))(\boldsymbol{A}'-\boldsymbol{\mu}_E)\odot(\boldsymbol{A}'-\boldsymbol{\mu}_E)\\ &= W_{E-1}\left(\boldsymbol{\sigma}_{E-1} + (\boldsymbol{\mu}_E-\boldsymbol{\mu}_{E-1})\odot(\boldsymbol{\mu}_E-\boldsymbol{\mu}_{E-1})\right).\end{aligned} \tag{16}$$

Substituting (16) into (15) yields the online update of $\boldsymbol{\sigma}_E$, which completes the proof.

SAGE-HSFL inherits the raw-data locality of split learning because raw samples remain on clients, while edge servers receive only cut-layer smashed data and the labels required for server-side training. The smashed-data pool is bounded by $b\lambda^j_a$, and supplementary smashed data are generated locally from the online label-wise statistics without additional raw-data transmission. The present work focuses on straggler resilience, non-IID update mitigation, and wireless resource optimization. Privacy protection for intermediate representations is a complementary research direction in split learning [26], [27]. Integrating secure storage, feature perturbation, and privacy-aware cut-layer selection with SAGE-HSFL is an important direction for future work.

B. Convergence Analysis

In this subsection, we present a convergence analysis to better characterize the error bound and the performance gains of SAGE-HSFL. For the analysis, we make the following assumptions on the local loss function $F_i(\cdot)$.

Assumption 1. (Smoothness) The loss function $F_i(\boldsymbol{u}, \boldsymbol{v})$ is continuously differentiable with respect to $\boldsymbol{u}$ and $\boldsymbol{v}$. There exist constants L_u, L_v, L_{uv}, and L_{vu} such that $\nabla_{\boldsymbol{u}} F_i(\boldsymbol{u}, \boldsymbol{v})$ is L_u-Lipschitz continuous in $\boldsymbol{u}$ and L_{uv}-Lipschitz continuous in $\boldsymbol{v}$, that is $\|\nabla_{\boldsymbol{u}} F_i(\boldsymbol{u}, \boldsymbol{v}) - \nabla_{\boldsymbol{u}} F_i(\boldsymbol{u}', \boldsymbol{v})\| \leq L_u \|\boldsymbol{u} - \boldsymbol{u}'\|$ and $\|\nabla_{\boldsymbol{u}} F_i(\boldsymbol{u}, \boldsymbol{v}) - \nabla_{\boldsymbol{u}} F_i(\boldsymbol{u}, \boldsymbol{v}')\| \leq L_{uv} \|\boldsymbol{v} - \boldsymbol{v}'\|$. $\nabla_{\boldsymbol{v}} F_i(\boldsymbol{u}, \boldsymbol{v})$ is L_v-Lipschitz and L_{vu}-Lipschitz continuous in $\boldsymbol{v}$ and $\boldsymbol{u}$.

Assumption 2. (Bounded stochastic gradient) The squared norms of the gradients are uniformly bounded, that is $\|\nabla_{\boldsymbol{u}} F_i(\boldsymbol{u}, \boldsymbol{v})\|^2 \leq G_1^2$ and $\|\nabla_{\boldsymbol{v}} F_i(\boldsymbol{u}, \boldsymbol{v})\|^2 \leq G_2^2$.

Assumption 3. (Bounded gradient variance) The stochastic gradients for the client-side and server-side models have bounded variance, that is $\mathbb{E}\big[\|\nabla_{\boldsymbol{u}} f_i(\boldsymbol{u}, \boldsymbol{v}) - \nabla_{\boldsymbol{u}} F_i(\boldsymbol{u}, \boldsymbol{v})\|^2\big] \leq \sigma_1^2$ and $\mathbb{E}\big[\|\nabla_{\boldsymbol{v}} f_i(\boldsymbol{u}, \boldsymbol{v}) - \nabla_{\boldsymbol{v}} F_i(\boldsymbol{u}, \boldsymbol{v})\|^2\big] \leq \sigma_2^2$.

Assumption 4. (Bounded dissimilarity) Data heterogeneity across clients induces gradient bias. We assume the gradient dissimilarity is bounded, that is $\mathbb{E}\big[\|\nabla_{\boldsymbol{u}} F_i(\boldsymbol{u}, \boldsymbol{v}) - \nabla_{\boldsymbol{u}} F(\boldsymbol{u}, \boldsymbol{v})\|^2\big] \leq \delta_1^2$ and $\mathbb{E}\big[\|\nabla_{\boldsymbol{v}} F_i(\boldsymbol{u}, \boldsymbol{v}) - \nabla_{\boldsymbol{v}} F(\boldsymbol{u}, \boldsymbol{v})\|^2\big] \leq \delta_2^2$.

In SAGE-HSFL, the smashed data distribution is learned online from the received smashed data pool. Since the received pool may be affected by non-IID participation and stragglers, we do not assume that the generated distribution is identical to the global smashed data distribution. Instead, we characterize the effect of generated smashed data through the following lemma.

Lemma 2. *Let g_1 denote the server-side stochastic gradient computed from the received smashed data, g_2 denote the server-side stochastic gradient computed from the generated smashed data, and $g_0 = \nabla_{\boldsymbol{v}} F(\boldsymbol{u}, \boldsymbol{v})$ denote the global server-side gradient. For notational simplicity, we omit the edge-server index and write $\lambda = \lambda^j$. Let $\rho = \frac{b\lambda}{b\lambda + b'}$, where $b\lambda$ is the number of received smashed-data samples and b' is the number of generated smashed-data samples. The SAGE server-side gradient $g_{\mathrm{SAGE}} = \rho g_1 + (1-\rho) g_2$ satisfies*

$$\mathbb{E}[\|g_{\mathrm{SAGE}} - g_0\|^2] = \rho^2 D_1 + (1-\rho)^2 D_2 + 2\rho(1-\rho) C_{12}, \tag{17}$$

where $D_1 = \mathbb{E}[\|g_1 - g_0\|^2]$, $D_2 = \mathbb{E}[\|g_2 - g_0\|^2]$, and $C_{12} = \mathbb{E}[\langle g_1 - g_0, g_2 - g_0 \rangle]$. Therefore, SAGE tightens the server-side gradient dissimilarity compared with the received-data-only update if

$$(1-\rho) D_2 + 2\rho C_{12} \leq (1+\rho) D_1. \tag{18}$$

Proof: Since the server-side gradient computed using both received and generated smashed data is the weighted average of the two mini-batch gradients, we have

$$g_{\mathrm{SAGE}} - g_0 = \rho(g_1 - g_0) + (1-\rho)(g_2 - g_0). \tag{19}$$

Expanding the squared norm and taking expectation yields

$$\begin{aligned}
&\mathbb{E}[\|g_{\mathrm{SAGE}} - g_0\|^2] = \rho^2 \mathbb{E}[\|g_1 - g_0\|^2] \\
&+ (1-\rho)^2 \mathbb{E}[\|g_2 - g_0\|^2] + 2\rho(1-\rho)\mathbb{E}[\langle g_1 - g_0, g_2 - g_0 \rangle] \\
&= \rho^2 D_1 + (1-\rho)^2 D_2 + 2\rho(1-\rho) C_{12},
\end{aligned} \tag{20}$$

which proves (17). Compared with the received-data-only dissimilarity D_1, SAGE achieves a tighter dissimilarity bound when $\rho^2 D_1 + (1-\rho)^2 D_2 + 2\rho(1-\rho) C_{12} \leq D_1$. Since $0 < \rho < 1$, this inequality is equivalent to $(1-\rho) D_2 + 2\rho C_{12} \leq (1+\rho) D_1$, which proves (18). This completes the proof. Before presenting the convergence results of SAGE-HSFL, we introduce the following lemma, which will be used in the analysis.

Lemma 3. *If Assumptions 1 and 3 hold, then the drift between the global models at two rounds t_1 and t_2 with $t_1 > t_2$ is bounded by $\mathbb{E}[\|\boldsymbol{u}_{t_1} - \boldsymbol{u}_{t_2}\|^2] \leq 3 q_1 \eta_u^2 (t_1 - t_2)^2$ and $\mathbb{E}[\|\boldsymbol{v}_{t_1} - \boldsymbol{v}_{t_2}\|^2] \leq 3 q_1' \eta_v^2 (t_1 - t_2)^2$, where $q_1 = (L^2 + L - 1)\sigma_1^2 + (L^2 + 2L(L-1)) G_1^2$ and $q_1' = (L^2 + L - 1)\sigma_2^2 + (L^2 + 2L(L-1)) G_2^2$.*

Proof: Please see Appendix A. ∎

Based on Lemma 3, we derive the one-round convergence bound of SAGE-HSFL in Theorem 1.

Theorem 1. *Suppose Assumptions 1, 3, and 4 hold. If the learning rates satisfy $\eta_u \leq \frac{1}{2LL_u(\kappa+1)}$ and $\eta_v \leq \frac{1}{2L_v(\kappa+1)}$, then the one-round convergence bound is*

$$\begin{aligned}
&F(\boldsymbol{u}_{t+1}, \boldsymbol{v}_{t+1}) - F(\boldsymbol{u}_t, \boldsymbol{v}_t) \leq -\frac{\eta_v}{2} \|\nabla_{\boldsymbol{v}} F(\boldsymbol{u}_t, \boldsymbol{v}_t)\|^2 \\
&+ (L^2 L_u \eta_u^2 (\kappa+1) - \frac{1}{2}\eta_u L) \|\nabla_{\boldsymbol{u}} F(\boldsymbol{u}_t, \boldsymbol{v}_t)\|^2 \\
&+ \frac{3}{I} q_1 L L_u \eta_u^3 (L_u + 3\eta_u(\kappa+1)) \sum\nolimits_{i=1}^{I} \tau_{i,t}^2 + q_2,
\end{aligned} \tag{21}$$

where $q_2 = 5L^2 L_u^2 \eta_u^3 \sigma_1^2 (2G_1^2 + \frac{1}{2}\sigma_1^2) + 15 L^3 L_u \eta_u^6 (\sigma_1^2 + 4G_1^2)(\kappa+1) + L^2 L_u \eta_u^2 \delta_1^2 (\kappa+1)(3\delta_1^2 + \frac{1}{2}\sigma_1^2) - \frac{\eta_v G_2^2}{2} + \frac{\eta_v \delta_2^2}{2} + L_v \eta_v^2 (\kappa+1)(\delta_2^2 + \frac{\sigma_2^2}{2})$.

Proof: Please see Appendix B. ∎

Based on Theorem 1, we obtain the convergence bound of SAGE-HSFL after T training rounds as follows.

Corollary 1. *Suppose the conditions in Theorem 1 hold and the learning rates satisfy $\eta_u \leq \frac{1}{2LL_u(\kappa+1)}$ and $\eta_v \leq \frac{1}{2L_v(\kappa+1)}$. the expected gap between the global loss after T training rounds and the optimal loss is bounded by*

$$\begin{aligned}
&\mathbb{E}[F(\boldsymbol{w}_T) - F(\boldsymbol{w}^*)] \leq q_3^T \mathbb{E}[F(\boldsymbol{w}_0) - F(\boldsymbol{w}^*)] \\
&+ q_2 \sum\nolimits_{t=0}^{T-1} q_3^t + \frac{3}{I} q_1 L L_u \eta_u^3 (L_u + 3\eta_u(\kappa+1)) \\
&\times \sum\nolimits_{t=0}^{T-1} q_3^{T-t-1} \sum\nolimits_{i=1}^{I} (1 - \alpha_{i,t}^j)(\tau_{i,t-1} + 1)^2,
\end{aligned} \tag{22}$$

where $q_3 = 1 - LL_u\eta_u + 2L^2 L_u^2 \eta_u^2 (\kappa+1) - L_v \eta_v + 2L_v^2 \eta_v^2 (\kappa+1)$.

Proof: Please see Appendix C. ∎

According to Corollary 1, the convergence bound of SAGE-HSFL consists of three terms. The first term is the initial gap between the global loss and the optimal loss. The second term depends on system hyperparameters. The third term captures the cumulative staleness of clients' local gradients. Note that $q_3 \leq 1$ when $\eta_u \leq \frac{1}{2LL_u(\kappa+1)}$ and $\eta_v \leq \frac{1}{2L_v(\kappa+1)}$. As T increases, q_3^T approaches 0, the first term vanishes. The second term is determined by system hyperparameters. In contrast, the third term is an explicit function of the client scheduling policy. Under a latency constraint, each edge server needs to

balance latency and accuracy by controlling the number of scheduled clients per round. For the third term, we provide the following remark:

Remark 1. As in many existing works, e.g., [10], [28], [29], we assume that the available wireless resources and the set of available clients are independent across rounds in problem $\mathcal{P}$. Based on Theorem 1 and Corollary 1, minimizing the convergence bound of $\mathbb{E}[F(\boldsymbol{w}_T) - F(\boldsymbol{w}^*)]$ can be approximated by minimizing the weighted sum of gradient staleness across rounds, namely $\sum_{t=0}^{T-1} q_3^{T-t-1} \sum_{i=1}^{I} (1-\alpha_{i,t}^j)(\tau_{i,t-1}+1)^2$. This term serves as a surrogate objective for minimizing the global loss.

IV. Optimal client scheduling, cut layer decision, bandwidth allocation, and edge association

In this section, we reformulate problem $\mathcal{P}$ based on the staleness-dependent term in the convergence analysis. We then develop online algorithms for cut layer decision, bandwidth allocation, and dynamic client scheduling within a single edge server. Subsequently, we design an adaptive edge association algorithm across all edge servers to improve overall system performance.

A. Problem Transformation

According to Corollary 1 and Remark 1, we reformulate problem $\mathcal{P}$ by focusing on the last term on the right-hand side of (22). In this respect, we introduce weights θ_t^j for each round t to capture the impact of gradient staleness at edge server j. We then define a weighted staleness objective over the training process, $\sum_{t=0}^{T-1} \theta_t^j \sum_{i\in\mathcal{U}_t^j} \alpha_{i,t}^j (\tau_{i,t-1}+1)^2$, and minimize it as a surrogate for minimizing the global loss. This leads to the following reformulated problem $\widetilde{\mathcal{P}}$:

$$\widetilde{\mathcal{P}}: \min_{\{\mathcal{M}_t^j, s_t^j, \boldsymbol{\omega}_t^j\}_{t=0}^{T-1}} \sum_{t=0}^{T-1} \theta_t^j \sum\nolimits_{i\in\mathcal{U}_t^j} \alpha_{i,t}^j (\tau_{i,t-1}+1)^2 \quad (23)$$
$$\text{s. t.} \quad (14a), (14b), (14c), (14d), (14e).$$

In $\widetilde{\mathcal{P}}$, the weight θ_t^j affects learning performance. Based on Corollary 1 and Remark 1, θ_t^j should increase with t because staleness in later rounds has a larger impact than in earlier rounds. At the same time, θ_t^j in early rounds should not be too small. Otherwise, too few clients may be scheduled early on, which can slow initial progress and hurt final performance. The term q_3^{T-t-1} with $0 < q_3 < 1$ increases with t and has a convex shape. This motivates choosing an increasing and convex weighting rule. We therefore set $\theta_t^j = \frac{1}{T-t}$. The effectiveness of this choice is evaluated in Section V.

Problem $\widetilde{\mathcal{P}}$ involves both discrete and continuous decision variables, and is therefore a mixed-integer optimization problem that is generally NP-hard. Moreover, solving it directly would require allocating the latency budget across rounds in advance. This is impractical because it needs accurate knowledge of channel conditions and client computation schedules over the entire training process. To enable online dynamic client scheduling, we adopt the Lyapunov optimization framework to handle the coupling across rounds. We introduce a virtual queue φ_t^j that evolves as

$$\varphi_{t+1}^j = \max(\varphi_t^j + \mathbf{T}_t^j - \frac{T_{\max}}{T}, 0), \quad (24)$$

Algorithm 1 Latency-aware Cut Layer Decision Algorithm

1: **Initialization:** Scheduled client subset $\mathcal{M}_t^j$, optimal bandwidth allocation policy $\boldsymbol{\omega}_t^j$ from the proposed binary search algorithm.
2: **for** each split point $s_t^j \in \mathcal{S}$ **do**
3: Calculate the expected latency $\mathbf{T}_t^j(\boldsymbol{\omega}_t^j, s_t^j) = \max_{i\in\mathcal{M}_t^j} \{T_{i,j,t}^{\mathrm{CM}} + \widehat{T}_{i,j,t}^{\mathrm{CP}}\}$.
4: **end for**
5: $s_{j,t}^* = \arg\min_{s_t^j\in\mathcal{S}} \mathbf{T}_t^j(\boldsymbol{\omega}_t^j, s_t^j)$.
6: Set $\boldsymbol{\omega}_{j,t}^* = \boldsymbol{\omega}_t^j$ as the optimal bandwidth allocation policy.
7: **return** Cut layer decision $s_{j,t}^*$ and bandwidth allocation $\boldsymbol{\omega}_{j,t}^*$.

with initial value $\varphi_0^j = 0$. Following the drift-plus-penalty method [30], we combine the long-term latency constraint in (14a) with the objective in (23) into a Lyapunov drift-plus-penalty function. This leads to the following per-round problem, which supports online scheduling:

$$\widehat{\mathcal{P}}: \min_{\mathcal{M}_t^j, s_t^j, \boldsymbol{\omega}_t^j} -\beta\theta_t^j \sum_{i\in\mathcal{U}_t^j} \alpha_{i,t}^j (\tau_{i,t-1}+1)^2 + \varphi_t^j \mathbf{T}_t^j \quad (25)$$
$$\text{s. t.} \quad (14b), (14c), (14d), (14e).$$

In $\widehat{\mathcal{P}}$, β controls the tradeoff between latency and accuracy. A larger β places more emphasis on reducing staleness and improving accuracy, while a smaller β prioritizes lower latency.

B. Optimal Bandwidth Allocation and Cut Layer Decision

For a given scheduled client set $\mathcal{M}_t^j$ and a cut layer decision s_t^j, we focus on the bandwidth allocation subproblem

$$\widehat{\mathcal{P}}_1: \min_{\boldsymbol{\omega}_t^j} \mathbf{T}_t^j \quad (26)$$
$$\text{s. t.} \quad (14b), (14c).$$

Problem $\widehat{\mathcal{P}}_1$ is convex [31], and its optimal solution is given in the following lemma.

Lemma 4. *The optimal bandwidth allocation of problem $\widehat{\mathcal{P}}_1$ satisfies the following condition:*

$$\omega_{i,t}^j = \frac{Q_{i,t}^j}{(\mathbf{T}_{j,t}^* - T_{i,j,t}^{\mathrm{rest}}) B^j \log_2(1+\frac{p_{i,t}h_{i,t}}{\sigma^2})}, \quad (27)$$

where $\mathbf{T}_{j,t}^*$ is the optimal latency for the scheduled client set in round t and is chosen to satisfy $\sum_{i\in\mathcal{M}_i^j} \omega_{i,t}^j = 1$. Here, $T_{i,j,t}^{\mathrm{rest}} = \widehat{T}_{i,j,t}^{\mathrm{CP}} + t_{i,j,t}^{\mathrm{AD}} + t_{i,j,t}^{\mathrm{MD}}$, and $Q_{i,t}^j = b\sum_{s=1}^{s_t^j}\gamma_s L + \sum_{s=1}^{s_t^j}\psi_s$ is the total client-side data size transmitted in one round.

Proof: From (11), the communication latency $T_{i,j,t}^{\mathrm{CM}}$ decreases monotonically with the allocated bandwidth ratio $\omega_{i,t}^j$. If a client finishes communication earlier than others, its unused bandwidth can be reassigned to slower clients. We can repeat this reassignment until all scheduled clients finish. Hence, the optimal bandwidth allocation satisfies

$$\begin{cases} T_{i,j,t}^{\mathrm{CM}} + \widehat{T}_{i,j,t}^{\mathrm{CP}} = \mathbf{T}_{j,t}^*, & \forall i \in M_t^j, \\ \sum_{i\in\mathcal{M}_i^j} \omega_{i,t}^j = 1, \end{cases} \quad (28)$$

where $\mathbf{T}_{j,t}^*$ denotes the optimal round latency at edge server j in round t. We complete the proof.

In Lemma 4, the optimal bandwidth allocation depends on an unknown variable $\mathbf{T}_{j,t}^*$. From (27), $\omega_{i,t}^j$ decreases as $\mathbf{T}_{j,t}^*$ increases. Therefore, we can use binary search to find $\mathbf{T}_{j,t}^*$ and then compute the optimal bandwidth allocation. To apply binary search, we first derive lower and upper bounds for

Algorithm 2 Dynamic Client Scheduling Algorithm

Initialization: Weight factor β, virtual queue φ_t^j, and θ_t^j.
Sort clients according to $\mathcal{R}_t^j$ ascendingly to acquire the sorted client set $\boldsymbol{M}^j$.
for $i \in \boldsymbol{M}^j$ **do**
  Update $\mathcal{M}_t^j = \mathcal{M}_t^j \cup \{i\}$.
  Solve the optimal bandwidth allocation and cut layer decision policy by Algorithm 1, i.e., $\boldsymbol{\omega}_{j,t}^*(\mathcal{M}_t^j)$ and $s_{j,t}^*(\mathcal{M}_t^j)$.
  Compute the drift-plus-penalty of $\mathcal{M}_t^j$, i.e., $\mathcal{R}(\mathcal{M}_t^j) = -\beta\theta_t^j \sum_{i\in\mathcal{M}_t^j} \alpha_{i,t}^j(\tau_{i,t-1}+1)^2 + \varphi_t^j \mathbf{T}_t^j$.
  if $-\beta(\tau_{i,t-1}+1)^2 + \varphi_t^j \mathbf{T}_t^j > 0$ **then**
    Break the loop.
  else
    $\widehat{\mathcal{M}} = \widehat{\mathcal{M}} \cup \mathcal{M}_t^j$
  end if
end for
Find the optimal policy $\mathcal{M}_{j,t}^* = \arg\min_{\mathcal{M}_t^j \in \widehat{\mathcal{M}}} \mathcal{R}(\mathcal{M}_t^j)$.
return Optimal policies $\mathcal{M}_{j,t}^*$, $\boldsymbol{\omega}_{j,t}^*(\mathcal{M}_{j,t}^*)$, and $s_{j,t}^*(\mathcal{M}_{j,t}^*)$.

Algorithm 3 Adaptive Edge Association Algorithm

Initialization: Sets of clients $\boldsymbol{\mathcal{U}}$, history set Φ^j.
for $j = 1$ to J **do**
  Edge server j randomly forms initial clients subset $\mathcal{U}^j$.
  Edge server j solves optimization problem by Algorithm 2 and derives an initial $\vartheta(\mathcal{U}^j)$.
end for
An initial edge association policy $\boldsymbol{\mathcal{U}}$ is obtained.
repeat
  for $i = 1$ to I **do**
    Each pair of edge server j and k ($j \neq k$) perform **transferring** client i ($i \in \mathcal{U}^j$) from $\mathcal{U}^j$ to $\mathcal{U}^k$ if permitted. Then Φ^j and Φ^k are accordingly updated.
  end for
  Randomly pick client $i \in \mathcal{U}^j$ and $n \in \mathcal{U}^k$ ($j \neq k$) to perform **exchanging** if permitted.
  Then Φ^j and Φ^k and $\boldsymbol{\mathcal{U}}$ are accordingly updated.
until no edge association adjustment is permitted by any client $i \in \mathcal{I}$.
return $\boldsymbol{\mathcal{U}}^*$ with each $\mathcal{U}^j \in \boldsymbol{\mathcal{U}}^*$ achieving $\mathcal{M}_j^*$, s_j^*, and $\boldsymbol{\omega}_j^*$.

$\mathbf{T}_{j,t}^*$. For the lower bound, note that the smallest allocated bandwidth fraction satisfies $\min_{i\in\mathcal{M}_t^j} \omega_{i,t}^{j*} \leq \frac{1}{|\mathcal{M}_t^j|}$. Thus, $\frac{\min_{i\in\mathcal{M}_t^j} \frac{Q_{i,t}^j}{B^j \log_2(1+\frac{p_{i,t}h_{i,t}}{\sigma^2})}}{\max_{i\in\mathcal{M}_t^j}(\mathbf{T}_{j,t}^* - T_{i,j,t}^{\mathrm{rest}})} \leq \frac{1}{|\mathcal{M}_t^j|}$. This yields the lower bound

$$\mathbf{T}_{j,t}^{\mathrm{lb}} = \min_{i\in\mathcal{M}_t^j} \frac{|\mathcal{M}_t^j| Q_{i,t}^j}{B^j \log_2(1+\frac{p_{i,t}h_{i,t}}{\sigma^2})} + \min_{i\in\mathcal{M}_t^j} T_{i,j,t}^{\mathrm{rest}}. \tag{29}$$

For the upper bound, we use $\max_{i\in\mathcal{M}_t^j} \omega_{i,t}^{j*} \geq \frac{1}{|\mathcal{M}_t^j|}$. Using a similar derivation, we obtain

$$\mathbf{T}_{j,t}^{\mathrm{ub}} = \max_{i\in\mathcal{M}_t^j} \frac{|\mathcal{M}_t^j| Q_{i,t}^j}{B^j \log_2(1+\frac{p_{i,t}h_{i,t}}{\sigma^2})} + \max_{i\in\mathcal{M}_t^j} T_{i,j,t}^{\mathrm{rest}}. \tag{30}$$

With $\mathbf{T}_{j,t}^{\mathrm{lb}}$ and $\mathbf{T}_{j,t}^{\mathrm{ub}}$, we design a binary search algorithm to find $\mathbf{T}_{j,t}^*$ and then compute the optimal bandwidth allocation. The algorithm repeatedly halves the search interval and stops when the tolerance ϵ is reached. The time complexity of it is $\mathcal{O}(\log_2 \frac{\mathbf{T}_{j,t}^{\mathrm{ub}} - \mathbf{T}_{j,t}^{\mathrm{lb}}}{\epsilon})$. Given a scheduled client set $\mathcal{M}_t^j$ and a bandwidth allocation policy $\boldsymbol{\omega}_t^j$, the cut layer decisions are decoupled and affect the objective additively. Therefore, each edge server can optimize its cut layer independently. We define the cut layer selection subproblem as

$$\widehat{\mathcal{P}}_2: \min_{s_t^j} \ \mathbf{T}_t^j \quad \text{s. t. (14d)}, \tag{31}$$

Problem $\widehat{\mathcal{P}}_2$ is nonconvex because (14d) restricts s_t^j to a finite discrete set. In addition, the sizes of smashed data, smashed gradients, and client-side models, as well as the computation costs of client-side FP and BP, all depend on the cut layer. Using a binary search algorithm, we adopt a sample average approximation (SAA) based method [32] to jointly determine the cut layer and bandwidth allocation. The procedure is summarized in Algorithm 1, with time complexity $\mathcal{O}(S \log_2 \frac{\mathbf{T}_{j,t}^{\mathrm{ub}} - \mathbf{T}_{j,t}^{\mathrm{lb}}}{\epsilon})$.

C. Dynamic Client Scheduling

Based on the above analysis, for any scheduled client set $\mathcal{M}_t^j$, the optimal bandwidth allocation and cut layer decision can be obtained using Algorithm 1. For client scheduling, a direct approach is to evaluate the objective value for every possible subset of $\mathcal{U}_t^j$ and choose the one with the minimum value. However, this is not practical because there are $2^{|\mathcal{U}_t^j|}$ possible scheduling subsets. This results in an exponential time complexity $\mathcal{O}(2^{|\mathcal{U}_t^j|} S \log_2 \frac{\mathbf{T}_{j,t}^{\mathrm{ub}} - \mathbf{T}_{j,t}^{\mathrm{lb}}}{\epsilon})$. Next, we develop an efficient algorithm to compute the client scheduling policy.

From the objective in (25), the goal is to choose a client subset that minimizes the drift-plus-penalty value $\mathcal{R}(\mathcal{U}_t^j) = -\beta\theta_t^j \sum_{i\in\mathcal{U}_t^j} \alpha_{i,t}^j(\tau_{i,t-1}+1)^2 + \varphi_t^j \mathbf{T}_t^j$. To identify suitable clients, we first assign equal bandwidth to all clients associated with edge server j and fix the cut layer. In each round t, we compute for each client i the score $\mathcal{R}_t^j = -\beta\theta_t^j(\tau_{i,t-1}+1)^2 + \varphi_t^j \mathbf{T}_t^j$, and sort clients in ascending order of $\mathcal{R}_{i,t}^j$. Let $\boldsymbol{M}^j$ denote the sorted client list. Standard sorting methods such as Heapsort or Mergesort have worst-case complexity $\mathcal{O}(|\mathcal{U}_t^j| \log |\mathcal{U}_t^j|)$. We then construct candidate scheduling sets by adding clients one by one from $\boldsymbol{M}^j$ into $\mathcal{M}_t^j$. For each candidate set $\mathcal{M}_t^j$, we run Algorithm 1 to obtain the corresponding optimal bandwidth allocation $\boldsymbol{\omega}_{j,t}^*(\mathcal{M}_t^j)$ and cut layer decision $s_{j,t}^*(\mathcal{M}_t^j)$. The resulting drift-plus-penalty value is denoted by $\mathcal{R}(\mathcal{M}_t^j)$. Let $\widehat{\mathcal{M}}$ be the collection of all candidate scheduling sets generated in this way. We select the final scheduling decision as $\mathcal{M}_{j,t}^* = \arg\min_{\mathcal{M}_t^j \in \widehat{\mathcal{M}}} \mathcal{R}(\mathcal{M}_t^j)$. The full procedure is summarized in Algorithm 2. The algorithm runs Algorithm 1 at most $|\mathcal{U}_t^j|$ times and has polynomial time complexity $\mathcal{O}(S|\mathcal{U}_t^j| \log_2 \frac{\mathbf{T}_{j,t}^{\mathrm{ub}} - \mathbf{T}_{j,t}^{\mathrm{lb}}}{\epsilon})$.

To evaluate the performance of Algorithm 2, we compare it with an optimal offline benchmark. The offline benchmark corresponds to problem $\widetilde{\mathcal{P}}$, where the computation and communication times over the whole learning process are assumed to be known in advance. Let $\alpha_{i,t}^{j*}$ denote the optimal client participation indicators obtained by solving the offline problem. The performance guarantee of the proposed dynamic client scheduling algorithm is given in Proposition 1.

Proposition 1. *Compared with the optimal offline solution, the cumulative objective value achieved by Algorithm* 2 satisfies

$$\sum_{t=0}^{T-1} \theta_t^j \sum_{i\in\mathcal{U}_t^j} \hat{\alpha}_{i,t}^j(\tau_{i,t-1}+1)^2 \geq -\frac{T^2\phi^2}{2\beta} + \sum_{t=0}^{T-1} \theta_t^j \sum_{i\in\mathcal{U}_t^j} \alpha_{i,t}^{j*}(\tau_{i,t-1}+1)^2, \tag{32}$$

and the latency of the proposed algorithm is bounded by

$$\sum_{t=0}^{T-1}\mathbf{T}_t^j \le T_{\max}+\sqrt{T\phi^2+2\beta\sum_{t=0}^{T-1}\theta_t^j\sum_{i\in\mathcal{U}_t^j}(\tau_{i,t-1}+1)^2}, \tag{33}$$

where $\phi=\max_t\{|T_t-\frac{T_{\max}}{T}|\}$.

Proof: Firstly, we define the Lyapunov function as $\mathcal{L}(t)=\frac{1}{2}(\varphi_t^j)^2$. According to the virtual queue defined in (24), we have $(\varphi_{t+1}^j)^2 \le (\varphi_t^j+\mathbf{T}_t^j-\frac{T_{\max}}{T})^2$. The Lyapunov drift in round t is $\Upsilon(t)=\mathcal{L}(t+1)-\mathcal{L}(t)$, i.e., $\Upsilon(t)=\frac{1}{2}((\varphi_{t+1}^j)^2-(\varphi_t^j)^2) \le \frac{1}{2}\phi^2+\varphi_t^j(\mathbf{T}_t^j-\frac{T_{\max}}{T})$, where $\phi=\max_t\{|\mathbf{T}_t^j-\frac{T_{\max}}{T}|\}$. By adding $-\beta\theta_t^j\sum_{i\in\mathcal{U}_t^j}\alpha_{i,t}^j(\tau_{i,t-1}+1)^2$ into both sides, the upper bound of the drift-plus-penalty in round t is

$$\Upsilon(t)-\beta\theta_t^j\sum\nolimits_{i\in\mathcal{U}_t^j}\alpha_{i,t}^j(\tau_{i,t-1}+1)^2 \le \frac{1}{2}\phi^2 + \varphi_t^j(\mathbf{T}_t^j-\frac{T_{\max}}{T})-\beta\theta_t^j\sum\nolimits_{i\in\mathcal{U}_t^j}\alpha_{i,t}^j(\tau_{i,t-1}+1)^2. \tag{34}$$

The classical drift-plus-penalty algorithm strives to minimize the upper bound of $\Upsilon(t)-\beta\theta_t^j\sum_{i\in\mathcal{U}_t^j}\alpha_{i,t}^j(\tau_{i,t-1}+1)^2$. Thus, its solution makes the right side of (34) not larger than that achieved by any feasible solution. Denoting the solution of the proposed algorithm by $\hat{\alpha}_{i,t}^j$, and considering a feasible solution with $\mathbf{T}_t^j=0$ and $\alpha_{i,t}^j=0$, we have

$$\Upsilon(t)-\beta\theta_t^j\sum\nolimits_{i\in\mathcal{U}_t^j}\hat{\alpha}_{i,t}^j(\tau_{i,t-1}+1)^2 \le \frac{1}{2}\phi^2. \tag{35}$$

Let $\Upsilon_T=\mathcal{L}(T-1)-\mathcal{L}(0)=\frac{1}{2}(\varphi_{T-1}^j)^2$ be the T round drift. Thus, we bound the T round drift-plus-penalty as $\Upsilon_T=\frac{1}{2}(\varphi_{T-1}^j)^2 \le \frac{1}{2}T\phi^2+\sum_{t=0}^{T-1}\beta\theta_t^j\sum_{i\in\mathcal{U}_t^j}(\tau_{i,t-1}+1)^2$. Thus, we have $(\varphi_{T-1}^j)^2 \le T\phi^2+2\sum_{t=0}^{T-1}\beta\theta_t^j\sum_{i\in\mathcal{U}_t^j}(\tau_{i,t-1}+1)^2$. According to the queue defined in (24), we have $\mathbf{T}_t^j-\frac{T_{\max}}{T} \le \varphi_{t+1}^j-\varphi_t^j$. Summing the above inequality along T rounds obtains $\sum_{t=0}^{T-1}(\mathbf{T}_t^j-\frac{T_{\max}}{T}) \le \sum_{t=0}^{T-1}(\varphi_{t+1}^j-\varphi_t^j) \le \sqrt{T\phi^2+2\sum_{t=0}^{T-1}\beta\theta_t^j\sum_{i\in\mathcal{U}_t^j}(\tau_{i,t-1}+1)^2}$. By rearranging the above inequality, we have (33). Below we analyze the optimality of the proposed algorithm, which minimizes the right side of (34). Since $\Upsilon(t)$ is positive, we have

$$-\sum_{t=0}^{T-1}\beta\theta_t^j\sum\nolimits_{i\in\mathcal{U}_t^j}\hat{\alpha}_{i,t}^j(\tau_{i,t-1}+1)^2 \le \sum_{t=0}^{T-1}\varphi_t^j(\mathbf{T}_{j,t}^*-\frac{T_{\max}}{T}) + \frac{1}{2}T\phi^2-\sum\nolimits_{t=0}^{T-1}\beta\theta_t^j\sum\nolimits_{i\in\mathcal{U}_t^j}\alpha_{i,t}^{j*}(\tau_{i,t-1}+1)^2, \tag{36}$$

where the second term of the above inequality is bounded as

$$\sum\nolimits_{t=0}^{T-1}\varphi_t^j(\mathbf{T}_{j,t}^*-\frac{T_{\max}}{T}) \le \frac{T(T-1)}{2}\phi^2, \tag{37}$$

where the inequality holds due to $\varphi_0^j=0$ and $\varphi_{t+1}^j-\varphi_t^j\le\phi$. Substituting (37) into (36), we have (32), the proof is completed. Based on Proposition 1, we provide the following remark.

Remark 2. For Algorithm 2, the latency constraint is satisfied up to an $\mathcal{O}(\sqrt{\beta})$ term. It also achieves an $\mathcal{O}(\frac{1}{\beta})$ optimality gap relative to the optimal offline solution. Therefore, the algorithm provides an $\mathcal{O}(\sqrt{\beta},\frac{1}{\beta})$ tradeoff between latency and learning performance. When β increases, the algorithm places more emphasis on reducing gradient staleness to improve accuracy, which can lead to higher latency, and vice versa.

D. Adaptive Edge Association

Initially, each edge server j is randomly assigned a client set $\mathcal{U}^j$. Given this association, each edge server solves the single-server optimization using Algorithm 2 and obtains the optimal client scheduling $\mathcal{M}_j^*$, cut layer decision s_j^*, and bandwidth allocation $\boldsymbol{\omega}_j^*$. Let $\vartheta(\mathcal{U}^j)=-\mathcal{R}(\mathcal{M}_j^*,s_j^*,\boldsymbol{\omega}_j^*)$ denote the utility of group $\mathcal{U}^j$. Let $\boldsymbol{\mathcal{U}}=\{\mathcal{U}^j,\forall j\in\mathcal{J}\}$ denote the collection of all groups, and define the total utility as $\vartheta(\boldsymbol{\mathcal{U}})=\sum_{j=1}^J\vartheta(\mathcal{U}^j)$. If $\vartheta(\boldsymbol{\mathcal{U}}_2)>\vartheta(\boldsymbol{\mathcal{U}}_1)$, then $\boldsymbol{\mathcal{U}}_2$ is preferred because maximizing $\vartheta(\boldsymbol{\mathcal{U}})$ is equivalent to minimizing the drift-plus-penalty. During the iterative association process, clients are allowed to perform the following operations:

- Transferring: move a client i from $\mathcal{U}^j$ to $\mathcal{U}^k$ only if the total utility increases, that is $\Delta=\vartheta(\boldsymbol{\mathcal{U}}_2)-\vartheta(\boldsymbol{\mathcal{U}}_1)>0$.
- Exchanging: swap client $i\in\mathcal{U}^j$ with client $n\in\mathcal{U}^k$ if $\Delta>0$.

After each transfer or exchange, the corresponding optimal policies and utilities are updated. To avoid repeatedly solving the same cases, each edge server maintains a history set Φ^j that stores past group compositions and their utilities. The procedure stops when no further transfer or exchange can increase the total utility, yielding a stable association policy $\boldsymbol{\mathcal{U}}^*$. The full procedure is summarized in Algorithm 3, with polynomial time complexity $\mathcal{O}(IJS|\mathcal{U}_t^j|\log_2\frac{\mathbf{T}_{j,t}^{\mathrm{ub}}-\mathbf{T}_{j,t}^{\mathrm{lb}}}{\epsilon})$. The framework supports an arbitrary number of edge servers. When $J=1$, the transfer and exchange operations in Algorithm 3 are inactive, and the system reduces to the single-edge two-tier case.

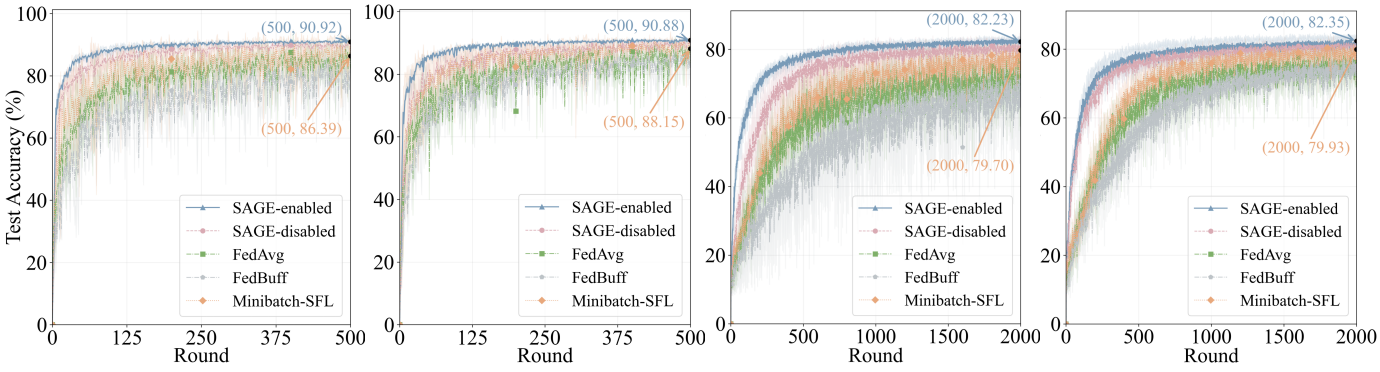
V. Experimental Results

In this section, we conduct extensive simulations to evaluate the SAGE-HSFL framework and the proposed algorithms.

TABLE II
Parameter Settings

Parameter	Value	Parameter	Value
I	100	J	5
B	10MHz	p_i	[10, 20]dBm
σ^2	10^{-12}W	h_0	-30dBm
v	2	f_i	[0.1, 0.8]GHz
f_j	[10, 20]GHz	b	32
p_j	[30, 40]dBm	R	{2, 4, 10}
η_u	0.01	η_v	0.01
L	{2, 10}	T	100
λ^j	10	b'	32

Experimental setup: We consider a three-tier HSFL system with one cloud server, J edge servers, and I clients. To model a practical wireless scenario, clients and edge servers are independently and uniformly placed in a circular area with radius 1000 m, and the cloud server is located at the center. Following [33], the channel gain is modeled as $h_{i,t}=h_0\rho_i(t)d_{i,j}^{-\nu}$, where h_0 is the path-loss constant, $\rho_i(t)\sim\exp(1)$ is the small-scale fading power gain of client i in round t and is independent across clients and rounds, $d_{i,j}$ is the distance between client i and edge server j, and ν is the path-loss exponent. We use a linearly increasing weighting function $\mathcal{W}(e)=e$. Unless otherwise stated, the default parameter settings follow [18], [22], [23], [28], [29] and are listed in Table II. Table II lists the main system and algorithmic parameters used in the experiments, including the numbers of clients and edge servers, wireless parameters, batch size, learning rates, local



(a) FMNIST, $\varpi = 2$ (b) FMNIST, $\iota = 0.1$ (c) CIFAR-10, $\varpi = 2$ (d) CIFAR-10, $\iota = 0.1$

Fig. 2. Test accuracy comparison between the SAGE-enabled framework and baselines on the FMNIST and CIFAR-10 datasets.

TABLE I
ACCURACY COMPARISON BETWEEN SAGE-ENABLED FRAMEWORK AND OTHER APPROACHES

Dataset	Target	FedAvg		FedBuff		Minibatch-SFL		SAGE-disabled		**SAGE-enabled**		Improvement	
		$\varpi = 2$	$\iota = 0.1$	$\varpi = 2$	$\iota = 0.1$	$\varpi = 2$	$\iota = 0.1$	$\varpi = 2$	$\iota = 0.1$	$\varpi = 2$	$\iota = 0.1$	$\varpi = 2$	$\iota = 0.1$
FMNIST	500	81.79% ± 1.75%	83.34% ± 0.90%	82.40% ± 0.97%	85.87% ± 0.42%	**86.39%** ± 2.12%	**88.15%** ± 2.45%	89.43% ± 1.01%	90.09% ± 0.26%	**90.92%** ± 0.25%	**90.88%** ± 0.20%	4.53%	2.73%
CIFAR-10	2000	76.65% ± 0.82%	76.05% ± 1.58%	70.82% ± 0.76%	75.18% ± 0.24%	**79.70%** ± 0.41%	**79.93%** ± 1.63%	80.23% ± 1.43%	80.29% ± 1.36%	**82.23%** ± 0.47%	**82.35%** ± 1.17%	2.53%	2.42%

iterations, and global rounds. The parameters θ_t^j and $\mathcal{W}(e)$ are algorithmic weighting rules. By default, we use $\theta_t^j = 1/(T-t)$ and $\mathcal{W}(e) = e$. The constants L_u, L_v, σ_1^2, σ_2^2, δ_1^2, δ_2^2, G_1, and G_2 are used for convergence analysis and are not tuned as experimental inputs. Each learning curve is averaged over five independent runs, and the shaded region denotes one standard deviation around the mean. Table I reports the corresponding final-round accuracy as mean ± standard deviation.

Dataset and model: We evaluate on FMNIST and CIFAR-10. To model data heterogeneity, we partition data across clients using a Dirichlet distribution Dir(ι), where ι controls the heterogeneity level. Smaller ι implies more severe heterogeneity. We also consider an extreme heterogeneity setting by splitting the dataset into equal partitions by class and assigning only ϖ class partitions to each client, where $\varpi = 3$ means each client has data from at most three classes. We use AlexNet as the model architecture.

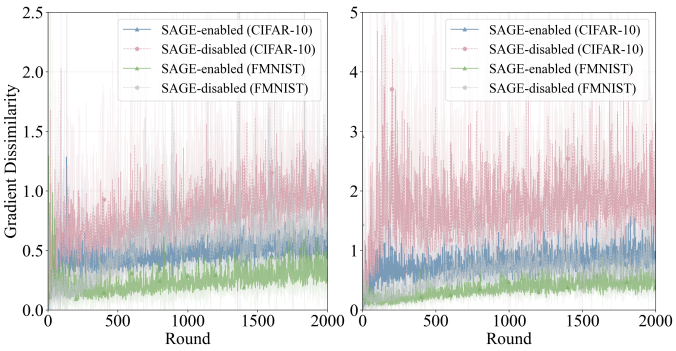
Baselines setting: We compare SAGE-HSFL with four representative baselines: 1) FedAvg [4] averages full local models from selected clients each round. 2) FedBuff [17] uses a buffer to support asynchronous FL. 3) Minibatch-SFL [34] is a synchronous split learning scheme that updates the server-side model using mini-batches to stabilize training. 4) S2FL [35] mitigates stragglers in synchronous SFL through adaptive model splitting and activation grouping. For the generation-strategy comparison, we further consider Replay-only and GMM. Replay-only directly resamples label-matched smashed data from the bounded pool, while GMM uses a label-wise two-component mixture with diagonal covariance. Replay-only, GMM, and SAGE-enabled use the same pool capacity and supplementary sample number.

A. *Effectiveness of Smashed Data Generating*

In this subsection, we evaluate the complete SAGE-HSFL framework against representative external baselines and isolate the contribution of SAGE through an otherwise identical SAGE-disabled variant. The external methods provide broader comparisons with existing learning protocols, while the SAGE-enabled versus SAGE-disabled comparison measures the effect of supplementary smashed data generation under the same system-control modules. In addition, we compare the aggregate server-side gradient dissimilarity of the SAGE-enabled and SAGE-disabled variants to complement the theoretical characterization in Lemma 2.

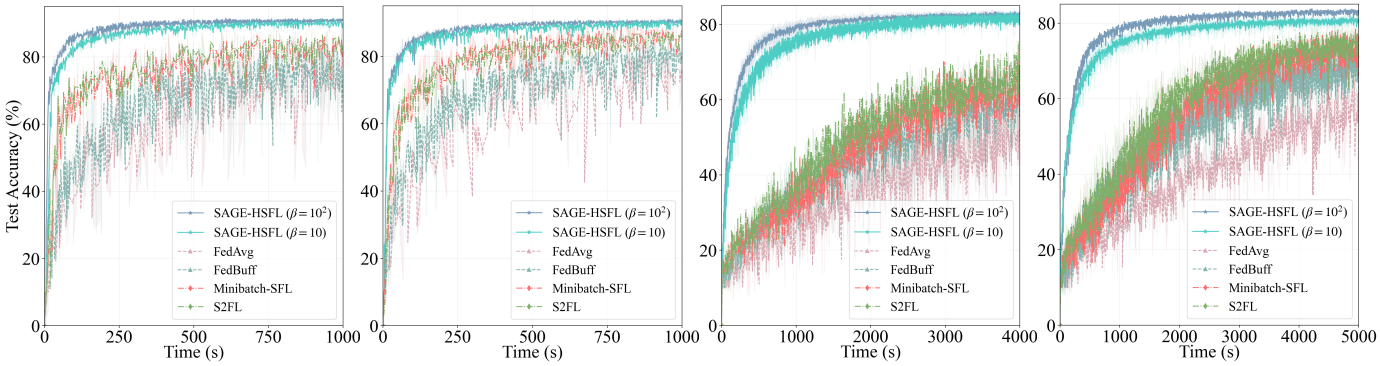
We conduct image classification on the FMNIST and CIFAR-10 datasets under both $\varpi = 2$ and $\iota = 0.1$. The learning curves are shown in Fig. 2, and the corresponding final-round results are summarized in Table I as mean ± standard deviation over five runs. Compared with the strongest representative external baseline, SAGE-HSFL improves the mean accuracy by 4.53% and 2.73% on FMNIST under $\varpi = 2$ and $\iota = 0.1$, respectively, and by 2.53% and 2.42% on CIFAR-10 under the same settings. These results evaluate the effectiveness of the complete framework relative to existing methods.

To isolate the contribution of smashed-data generation, we further compare SAGE-HSFL with the otherwise identical



(a) $\varpi = 2$ (b) $\iota = 0.1$

Fig. 4. Impact of the SAGE mechanism on gradient dissimilarity.



(a) FMNIST, $\varpi = 2$ (b) FMNIST, $\iota = 0.1$ (c) CIFAR-10, $\varpi = 2$ (d) CIFAR-10, $\iota = 0.1$

Fig. 3. Time-efficiency comparison of SAGE-HSFL and baselines on the FMNIST and CIFAR-10 datasets.

TABLE III
ACCURACY-OVER-TIME COMPARISON BETWEEN SAGE-HSFL AND THE BASELINES.

Dataset	Target	FedAvg		FedBuff		Minibatch-SFL		S2FL		SAGE ($\beta = 10$)		SAGE ($\beta = 10^2$)		Improvement	
		$\varpi=2$	$\iota=0.1$	$\varpi=2$	$\iota=0.1$	$\varpi=2$	$\iota=0.1$	$\varpi=2$	$\iota=0.1$	$\varpi=2$	$\iota=0.1$	$\varpi=2$	$\iota=0.1$	$\varpi=2$	$\iota=0.1$
FMNIST	1000s	72.68%	76.26%	74.30%	81.11%	77.71%	**87.66%**	**80.81%**	86.16%	89.92%	90.13%	**91.02%**	**90.52%**	10.21%	2.86%
CIFAR-10	4000s	51.75%	54.85%	55.83%	67.38%	66.60%	**75.06%**	**68.56%**	72.41%	81.11%	80.20%	**82.73%**	**82.51%**	14.17%	7.45%

SAGE-disabled variant. SAGE improves the mean accuracy by 1.49% and 0.79% on FMNIST under $\varpi = 2$ and $\iota = 0.1$, respectively, and by 2.00% and 2.06% on CIFAR-10. The gains are positive in all four evaluated settings. On FMNIST, the gain is larger under the more extreme class-partition setting, while on CIFAR-10 the gains are comparable across the two non-IID settings. Moreover, the gains on CIFAR-10 are larger than those on FMNIST under both corresponding settings, suggesting that SAGE offers greater benefits for the more complex learning task. Since FMNIST is comparatively simple and the SAGE-disabled framework already approaches 90% accuracy, the remaining improvement space is limited. Nevertheless, SAGE still provides a positive mean gain in every evaluated setting. We assess the SAGE-enabled versus SAGE-disabled accuracy gains using two-sided paired t-tests on the five matched final-round results, with Holm correction across the four comparisons. All adjusted p-values are below 0.05, supporting statistically significant positive mean gains on FMNIST and CIFAR-10 under both $\varpi = 2$ and $\iota = 0.1$.

Fig. 4 reports the aggregate server-side gradient dissimilarity on FMNIST and CIFAR-10 with and without SAGE. Across all data distribution settings, the SAGE-enabled approach consistently yields lower gradient dissimilarity than its SAGE-disabled counterpart. The consistent reduction provides empirical support for the dissimilarity-reduction behavior characterized by Lemma 2 under the evaluated settings. The results indicate that SAGE improves test accuracy and mitigates server-side gradient dissimilarity.

To further evaluate the generated smashed data quality, we compare generated samples with held-out real smashed data that are not used for fitting the label-wise distributions. We report the normalized mean discrepancy Δ_μ, normalized variance discrepancy Δ_σ, and maximum mean discrepancy. They are defined as $\Delta_\mu = \frac{1}{|\mathcal{Y}|}\sum_{y\in\mathcal{Y}} \frac{\|\boldsymbol{\mu}_y^{\text{real}} - \boldsymbol{\mu}_y^{\text{gen}}\|_2}{\|\boldsymbol{\mu}_y^{\text{real}}\|_2 + \epsilon}$ and $\Delta_\sigma = \frac{1}{|\mathcal{Y}|}\sum_{y\in\mathcal{Y}} \frac{\|\boldsymbol{\sigma}_y^{\text{real}} - \boldsymbol{\sigma}_y^{\text{gen}}\|_2}{\|\boldsymbol{\sigma}_y^{\text{real}}\|_2 + \epsilon}$, where ϵ is a small constant for numerical stability. We also evaluate different cut layers to test the effect of smashed data dimensionality.

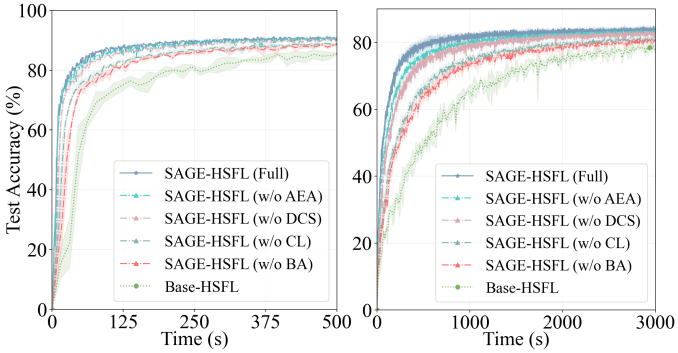
TABLE IV
GENERATION QUALITY AND CUT LAYER SENSITIVITY.

Dataset	Cut layer	d_s	Δ_μ	Δ_σ	MMD
FMNIST	s_1	6272	0.0734	0.1803	0.1102
FMNIST	s_2	3136	0.0744	0.2009	0.0979
CIFAR-10	s_1	8192	0.1321	0.2320	0.0926
CIFAR-10	s_2	4096	0.0907	0.2103	0.0945

Table IV shows that the generated smashed data preserve the main first-order and second-order statistics of held-out real smashed data, while the discrepancy varies with the cut-layer dimension. This observation is consistent with Lemma 2, which treats generation quality as a condition for reducing server-side gradient dissimilarity.

B. Superiority of SAGE-HSFL

In this subsection, we evaluate the time efficiency and accuracy gains of the proposed SAGE-HSFL framework against four baseline methods on the FMNIST and CIFAR-10 datasets. We consider $\beta \in \{10, 10^2\}$ as representative low and high operating points to illustrate its effect on the accuracy-over-time behavior. The analytical order of the latency-learning tradeoff is characterized in Remark 2. As shown in Fig. 3, SAGE-HSFL consistently achieves superior convergence speed and higher



(a) FMNIST, $\iota = 0.5$ (b) CIFAR-10, $\iota = 0.5$

Fig. 5. Convergence speed comparison between SAGE-HSFL and ablation baselines on the FMNIST and CIFAR-10 datasets.

test accuracy compared with the baselines. The corresponding quantitative results are summarized in Table III.

Specifically, Fig. 3(a) reports the results on FMNIST with $\varpi = 2$. With $\beta = 10^2$, SAGE-HSFL achieves the highest test accuracy of 91.02% under the target time of 1000s, whereas the strongest baseline, S2FL, attains only 80.81%, i.e., 10.21% lower. When the data distribution is switched to $\iota = 0.1$, Fig. 3(b) shows that SAGE-HSFL ($\beta = 10^2$) reaches 90.52% accuracy, outperforming the best baseline Minibatch-SFL (87.66%) by 2.86%.

On CIFAR-10, Fig. 3(c) demonstrates that, with $\varpi = 2$ and a target time of 4000s, SAGE-HSFL ($\beta = 10^2$) achieves 82.73% accuracy, exceeding S2FL (68.56%) by 14.17%. Under the Dirichlet setting $\iota = 0.1$ in Fig. 3(d), SAGE-HSFL ($\beta = 10^2$) still attains 82.51% test accuracy, which is 7.45% higher than the test accuracy of Minibatch-SFL (75.06%).

It can be observed that SAGE-HSFL with both $\beta = 10^2$ and $\beta = 10$ attains higher test accuracy than all baseline methods under the same target time. Notably, the performance gain becomes more pronounced under more heterogeneous (i.e., more extreme) data distributions. These results demonstrate the superiority of SAGE-HSFL in achieving both improved time efficiency and higher learning accuracy compared with the baselines.

To complement the same-time accuracy comparison, we further evaluate the wall-clock time required to reach the same target accuracy. For each dataset and non-IID setting, the target accuracy is selected within the range reached by the external baselines. We record the first time at which each method reaches this target. Among FedAvg, FedBuff, Minibatch-SFL, and S2FL, the method requiring the shortest time is reported as the best external baseline for that setting. Table V shows that SAGE-HSFL reaches the selected target accuracies 3.14–6.48 times faster than the best external baselines under the tested settings. This confirms that the time-efficiency improvement of SAGE-HSFL is also observed against external baselines, while the 6.20 and 7.02 times speedups in the ablation study quantify the gain over the basic HSFL ablation.

TABLE V
TIME-TO-TARGET COMPARISON WITH EXTERNAL BASELINES.

Dataset	Setting	Target	SAGE-HSFL	Best baseline	Speedup
FMNIST	$\varpi = 2$	80%	78.76s	331.82s (S2FL)	**4.21**×
FMNIST	$\iota = 0.1$	85%	137.15s	431.33s (Minibatch-SFL)	**3.14**×
CIFAR-10	$\varpi = 2$	65%	394.09s	2553.15s (FedBuff)	**6.48**×
CIFAR-10	$\iota = 0.1$	75%	795.78s	3350.94s (S2FL)	**4.21**×

C. Ablation Experiment of SAGE-HSFL

To evaluate the effectiveness of the proposed algorithms, we conduct an ablation study that includes the full SAGE-HSFL framework, four ablated variants, and a baseline HSFL scheme, as summarized below: (i) SAGE-HSFL (Full): the complete framework proposed in this paper. (ii) SAGE-HSFL (w/o AEA): SAGE-HSFL without adaptive edge association (AEA); clients are associated with the nearest edge server using a static rule. (iii) SAGE-HSFL (w/o DCS): SAGE-HSFL without dynamic client scheduling (DCS); clients are selected uniformly at random. (iv) SAGE-HSFL (w/o CL): SAGE-HSFL without latency-aware cut layer decision (CL); a fixed cut layer is used for all clients. (v) SAGE-HSFL (w/o BA): SAGE-HSFL without optimal bandwidth allocation (BA); bandwidth is allocated uniformly. (vi) Base-HSFL: a basic HSFL framework that disables SAGE, BA, CL, DCS, and AEA, i.e., it uses no smashed-data generation and adopts static nearest-edge association, random client selection, a fixed cut layer, and uniform bandwidth allocation. All experiments are conducted on the FMNIST and CIFAR-10 datasets under a Dirichlet data partition with $\iota = 0.5$. The detailed results are reported in Table VI.

TABLE VI
CONVERGENCE SPEED COMPARISON BETWEEN SAGE-HSFL AND ABLATION VARIANTS.

Method	FMNIST		CIFAR-10	
	Target	Time	Target	Time
Base-HSFL	85%	**433.21s**	75%	**1899.15s**
SAGE-HSFL (w/o BA)	85%	197.64s	75%	1342.19s
SAGE-HSFL (w/o CL)	85%	171.35s	75%	1118.34s
SAGE-HSFL (w/o DCS)	85%	97.70s	75%	487.95s
SAGE-HSFL (w/o AEA)	85%	98.77s	75%	489.39s
SAGE-HSFL (Full)	85%	**69.88s**	75%	**270.65s**
Speed Up	**6.20x**		**7.02x**	

As shown in Fig. 5, the proposed SAGE-HSFL (Full) converges faster than the ablation variants and Base-HSFL. Specifically, for FMNIST with a target accuracy of 85%, Fig. 5(a) indicates that SAGE-HSFL (Full) requires only 69.88s to reach the target, whereas the slowest method, Base-HSFL, needs 433.21s to reach this target accuracy, which is 6.20 times longer than the proposed framework. Similarly, on CIFAR-10 with a target accuracy of 75%, SAGE-HSFL (Full) only needs 270.65s to reach the target, while the Base-HSFL takes up to 1899.15s, corresponding to a 7.02 times longer convergence time.

These results demonstrate that the full resource-optimized SAGE-HSFL framework substantially accelerates convergence compared with the basic HSFL ablation. Moreover, the ablation study confirms that each proposed algorithm effectively contributes to improving the overall learning performance of SAGE-HSFL.

D. Generality Evaluation and Sensitivity Study

To further evaluate the generality of SAGE-HSFL, we conduct additional experiments on CIFAR-10 with ResNet-18. We test three cut layers, denoted by s_1, s_2, and s_3, corresponding to shallow, middle, and deep split points. We also consider a high-heterogeneity wireless setting, where the client computing frequency is sampled from $[0.05, 0.8]$ GHz and the client transmit power is sampled from $[5, 20]$ dBm. Compared with the default setting in Table II, this setting increases both computation and communication heterogeneity.

TABLE VII
GENERALITY EVALUATION OF TEST ACCURACY.

Scenario	Cut layer	Best baseline	SAGE-disabled	SAGE-HSFL
Larger model	s_1	87.98% (S2FL)	89.75%	**90.60%**
Deeper split	s_2	87.87% (S2FL)	88.62%	**89.14%**
Deeper split	s_3	87.41% (S2FL)	88.71%	**89.40%**
Wireless stress	s_2	88.10% (S2FL)	89.32%	**89.53%**

Table VII evaluates the generality of test accuracy under a larger model, deeper split points, and a more heterogeneous wireless setting. The results show that the accuracy benefit of SAGE-HSFL persists beyond the default AlexNet setting. At the deeper ResNet-18 splits s_2 and s_3, SAGE-HSFL improves test accuracy over SAGE-disabled by 0.52 and 0.69 percentage points, respectively, supporting the effectiveness of diagonal-Gaussian supplementation at these split points.

To test the sensitivity to the weighting rules, we further evaluate different choices of θ_t^j and $\mathcal{W}(e)$ under a Dirichlet data partition with $\iota = 0.5$. For θ_t^j, we compare a constant rule, a linear increasing rule, and the default reciprocal rule $1/(T-t)$ while fixing $\mathcal{W}(e) = e$. For $\mathcal{W}(e) = ke^a$, we fix $k = 1$ and the default $\theta_t^j = 1/(T-t)$, and vary a, where $a = 0$ gives uniform weighting and larger a gives stronger preference to newer smashed data. In Table VIII, Accuracy denotes the test accuracy at the final training round, and Time denotes the simulated wall-clock time required to reach the target accuracy, i.e., 85% for FMNIST and 75% for CIFAR-10.

TABLE VIII
SENSITIVITY TO WEIGHTING RULES.

Study	Setting	FMNIST		CIFAR-10	
		Accuracy	Time	Accuracy	Time
θ_t^j	Constant	89.32%	668.58s	73.72%	6916.35s
θ_t^j	Linear	89.67%	265.51s	77.45%	5290.84s
θ_t^j	$1/(T-t)$	91.16%	69.88s	84.31%	270.65s
$\mathcal{W}(e)$	$a=0$	91.35%	61.36s	83.33%	325.49s
$\mathcal{W}(e)$	$a=1$	91.16%	69.88s	84.31%	270.65s
$\mathcal{W}(e)$	$a=2$	91.08%	67.04s	84.15%	344.32s

Table VIII reports the sensitivity of SAGE-HSFL to the staleness-weighting and recency-weighting rules. For θ_t^j, the reciprocal rule $1/(T-t)$ achieves the best accuracy-time tradeoff on both datasets. Compared with the constant and linear rules, it improves the final accuracy from 89.32% and 89.67% to 91.16% on FMNIST, and reduces the time-to-target from 668.58s and 265.51s to 69.88s. The gain is more pronounced on CIFAR-10, where the reciprocal rule improves the final accuracy from 73.72% and 77.45% to 84.31%, and reduces the time-to-target from 6916.35s and 5290.84s to 270.65s. These results show that assigning larger weights to later-round staleness is beneficial, since stale gradients in later rounds can cause larger model drift and have stronger impact on the final model. For $\mathcal{W}(e)$, the results are relatively stable across $a = 0$, $a = 1$, and $a = 2$. On FMNIST, the final accuracies remain within 0.27 percentage points and the time-to-target values are close. On CIFAR-10, $a = 1$ achieves the best final accuracy and the shortest time-to-target among the tested recency-weighting choices. This indicates that SAGE-HSFL is not highly sensitive to the exact recency exponent, while the default setting $\mathcal{W}(e) = e$ provides a favorable and balanced choice across datasets.

We further evaluate the sensitivity to the generated smashed-data number b' on CIFAR-10 under $\varpi = 2$. We fix $\lambda^j = 10$ and set b' to 16, 32, and 64, yielding $\rho = 0.952$, 0.909, and 0.833, respectively. The corresponding final-round accuracies are 81.89%, 82.23%, and 82.26%, while the simulated times required to reach 75% accuracy are 987.43s, 802.54s, and 929.00s. Increasing b' from 16 to 32 improves the final accuracy by 0.34 percentage points and reduces the time-to-target by 184.89s. Increasing b' further to 64 provides only a 0.03 percentage-point accuracy gain but increases the time-to-target by 126.46s. These results show that the default setting $b' = 32$ provides the most favorable accuracy-time tradeoff among the evaluated settings.

TABLE IX
COMPARISON OF SMASHED-DATA SUPPLEMENTATION STRATEGIES ON CIFAR-10.

Method	Accuracy (%)		Extra time (ms/update)
	$\varpi = 2$	$\iota = 0.1$	
SAGE-disabled	80.23 ± 1.43	80.29 ± 1.36	–
Replay-only	80.81 ± 0.87	80.62 ± 0.42	38.10
GMM	82.42 ± 1.30	82.18 ± 1.64	224.88
SAGE-enabled	82.23 ± 0.47	82.35 ± 1.17	27.82

E. Comparison of Smashed-Data Supplementation Strategies

To separate historical-data reuse from distribution-based generation, we compare four smashed-data supplementation strategies on CIFAR-10 under $\varpi = 2$ and $\iota = 0.1$. SAGE-disabled uses no supplementary smashed data. Replay-only directly resamples label-matched smashed data from the same bounded pool. GMM samples from a label-wise two-component Gaussian mixture with diagonal covariance. SAGE-enabled uses the proposed label-wise diagonal Gaussian model. All four methods use the same data partitions, model initialization, wireless realizations, system-control modules, and random seeds. Replay-only, GMM, and SAGE-enabled further use the same pool capacity, supplementary sample number, recency weighting, and server-update schedule. Accuracy denotes the final-round test accuracy and is reported as mean $\pm$ standard deviation over five runs. Extra time denotes the average additional edge-server processing time per supplementation-triggered update, including historical-data replay or online distribution updates, distribution sampling, and the additional FP and BP on supplementary smashed data.

Table IX separates the contribution of historical smashed-data reuse from that of distribution-based generation. The comparison between SAGE-disabled and Replay-only measures the benefit of reusing stored samples, while the comparison between Replay-only and SAGE-enabled measures the additional benefit of generating new samples. The GMM results further examine whether increased distribution expressiveness provides sufficient accuracy improvement to justify its additional server-side processing cost.

Replay-only yields mean accuracy gains of 0.58 and 0.33 percentage points over SAGE-disabled under $\varpi = 2$ and $\iota = 0.1$, respectively. These modest positive gains indicate that historical smashed-data reuse provides a limited contribution. SAGE-enabled further improves upon Replay-only by 1.42 and 1.73 percentage points, showing that most of the mean improvement originates from distribution-based generation rather than historical replay alone. Replay-only and SAGE-enabled incur comparable additional processing times of 38.10 and 27.82 ms per update because both process the same number of supplementary samples and share the

additional FP and BP workload. Across the two non-IID settings, GMM and SAGE-enabled differ in mean accuracy by no more than 0.19 percentage points. Under $\varpi = 2$, GMM achieves $82.42 \pm 1.30\%$, which is 0.19 percentage points higher than SAGE-enabled at $82.23 \pm 0.47\%$. Under $\iota = 0.1$, SAGE-enabled achieves $82.35 \pm 1.17\%$, which is 0.17 percentage points higher than GMM at $82.18 \pm 1.64\%$. GMM requires 224.88 ms of additional processing per update, compared with 27.82 ms for SAGE-enabled, corresponding to an 87.63% reduction. These results demonstrate that the label-wise diagonal Gaussian provides a favorable balance between accuracy, stability, and online edge complexity.

For SAGE-enabled, the complete additional processing time is 27.82 ms per supplementation-triggered update. Together with the 10% increase in server-side sample workload and the absence of additional communication, this result provides direct implementation-level accounting of the SAGE overhead omitted from the dominant-term latency model.

VI. Conclusion

In this work, we propose SAGE-HSFL, a scalable learning framework for wireless networks. To address stragglers and data heterogeneity and to tighten the convergence bound, we design a SAGE mechanism that generates smashed data from a historical pool to reduce bias in server-side updates. We then analyze the convergence of SAGE-HSFL and derive a metric that motivates a new optimization objective. Based on this objective, we reformulate the original problem into a tractable form and develop algorithms for bandwidth allocation, latency-aware cut layer decision, and dynamic client scheduling. We also design an adaptive edge association algorithm that assigns clients to edge servers to further improve system performance. Simulations show that SAGE-HSFL improves test accuracy and speeds up convergence.

Appendix

A. Proof of Lemma 3

For two different rounds, i.e., t_1 and t_2 $(t_1 > t_2)$, we have

$$\mathbb{E}[\|\boldsymbol{u}_{t_1} - \boldsymbol{u}_{t_2}\|^2] = \mathbb{E}\left[\left\|\sum\nolimits_{t=t_2}^{t_1-1}(\boldsymbol{u}_{t+1} - \boldsymbol{u}_t)\right\|^2\right]$$
$$\overset{(a)}{\leq} \frac{3\eta_u^2 L}{I}(t_1 - t_2)\sum_{t=t_2}^{t_1-1}\sum_{i=1}^{I}\sum_{l=0}^{L-1}\mathbb{E}[\|\boldsymbol{u}_{i,t-\tau_{i,t},l} - \boldsymbol{u}_{t-\tau_{i,t}}\|^2]$$
$$+ 3\eta_u^2 L^2(t_1 - t_2)^2(G_1^2 + \sigma_1^2). \tag{38}$$

where (a) follows triangle inequality. Now we bound $\mathbb{E}[\|\boldsymbol{u}_{i,t-\tau_{i,t},l} - \boldsymbol{u}_{t-\tau_{i,t}}\|^2]$:

$$\mathbb{E}[\|\boldsymbol{u}_{i,t,l} - \boldsymbol{u}_t\|^2] = \mathbb{E}[\|\boldsymbol{u}_{i,t-1,l} - \eta_u\nabla_{\boldsymbol{u}} f_i(\boldsymbol{w}_{i,t-1,l}) - \boldsymbol{u}_t\|^2]$$
$$= \mathbb{E}[\|\boldsymbol{u}_{i,t-1,l} - \eta_u\nabla_{\boldsymbol{u}} F_i(\boldsymbol{w}_{i,t-1,l}) - \boldsymbol{u}_t\|^2]$$
$$+ \eta_u^2\mathbb{E}[\|\nabla_{\boldsymbol{u}} f_i(\boldsymbol{w}_{i,t-1,l}) - \nabla_{\boldsymbol{u}} F_i(\boldsymbol{w}_{i,t-1,l})\|^2] \overset{(a)}{\leq} 2\eta_u^2 L G_1^2$$
$$+ (1 + \frac{3}{2(L-1)})\mathbb{E}[\|\boldsymbol{u}_{i,t-1,l} - \boldsymbol{u}_t\|^2] + \eta_u^2\sigma_1^2, \tag{39}$$

where (a) follows Assumption 3, triangle inequality and Assumption 1. Telescoping the above inequality and we obtain $\mathbb{E}[\|\boldsymbol{u}_{i,t,l} - \boldsymbol{u}_t\|^2] \leq (2\eta_u^2 L G_1^2 + \eta_u^2\sigma_1^2)\frac{(1+\frac{3}{2(L-1)})^{L-1}-1}{\frac{3}{2(L-1)}} \overset{(a)}{\leq} 4\eta_u^2(L-1)(2LG_1^2 + \sigma_1^2)$, where (a) is due to $(1 + \frac{3}{2(L-1)})^{L-1} = (1 + \frac{3}{2(L-1)})^{\frac{3}{2}\frac{2(L-1)}{3}} \leq \exp(\frac{3}{2}) < 5$ and $\frac{2(L-1)}{3} < L - 1$, plug it back into (38), we have

$$\mathbb{E}[\|\boldsymbol{u}_{t_1} - \boldsymbol{u}_{t_2}\|^2] \leq 3q_1\eta_u^2(t_1 - t_2)^2, \tag{40}$$

where $q_1 = (L^2 + L - 1)\sigma_1^2 + (L^2 + 2L(L-1))G_1^2$. Similarly, for the server-side model drift, we have $\mathbb{E}[\|\boldsymbol{v}_{t_1} - \boldsymbol{v}_{t_2}\|^2] \leq 3q_1'\eta_v^2(t_1 - t_2)^2$, where $q_1' = (L^2 + L - 1)\sigma_2^2 + (L^2 + 2L(L-1))G_2^2$. This completes the proof.

B. Proof of Theorem 1

Using L_u-smooth of $F_i(\boldsymbol{u}_i, \cdot)$ and L_v-smooth of $F_i(\cdot, \boldsymbol{v}_i)$, we have $\mathbb{E}[F(\boldsymbol{u}_t', \boldsymbol{v}_t') - F(\boldsymbol{u}_t, \boldsymbol{v}_t')] \leq \langle\nabla_{\boldsymbol{u}} F(\boldsymbol{u}_t, \boldsymbol{v}_t'), \boldsymbol{u}_t' - \boldsymbol{u}_t\rangle + \frac{L_u}{2}\|\boldsymbol{u}_t' - \boldsymbol{u}_t\|^2$, and $\mathbb{E}[F(\boldsymbol{u}_t, \boldsymbol{v}_t') - F(\boldsymbol{u}_t, \boldsymbol{v}_t)] \leq \langle\nabla_{\boldsymbol{v}} F(\boldsymbol{u}_t, \boldsymbol{v}_t'), \boldsymbol{v}_t' - \boldsymbol{v}_t\rangle + \frac{L_v}{2}\|\boldsymbol{v}_t' - \boldsymbol{v}_t\|^2$. Summarizing aforementioned two inequalities, we have

$$\mathbb{E}[F(\boldsymbol{u}_t', \boldsymbol{v}_t') - F(\boldsymbol{u}_t, \boldsymbol{v}_t)] \leq \langle\nabla_{\boldsymbol{u}} F(\boldsymbol{u}_t, \boldsymbol{v}_t'), \boldsymbol{u}_t' - \boldsymbol{u}_t\rangle$$
$$+ \frac{L_u}{2}\|\boldsymbol{u}_t' - \boldsymbol{u}_t\|^2 + \langle\nabla_{\boldsymbol{v}} F(\boldsymbol{u}_t, \boldsymbol{v}_t), \boldsymbol{v}_t' - \boldsymbol{v}_t\rangle + \frac{L_v}{2}\|\boldsymbol{v}_t' - \boldsymbol{v}_t\|^2. \tag{41}$$

We now focus on bounding $\langle\nabla_{\boldsymbol{u}} F(\boldsymbol{u}_t, \boldsymbol{v}_t'), \boldsymbol{u}_t' - \boldsymbol{u}_t\rangle$ as follows

$$\langle\nabla_{\boldsymbol{u}} F(\boldsymbol{u}_t, \boldsymbol{v}_t'), \boldsymbol{u}_t' - \boldsymbol{u}_t\rangle \overset{(a)}{=} \langle\nabla_{\boldsymbol{u}} F(\boldsymbol{u}_t, \boldsymbol{v}_t), \boldsymbol{u}_t' - \boldsymbol{u}_t\rangle$$
$$+ \langle\nabla_{\boldsymbol{u}} F(\boldsymbol{u}_t, \boldsymbol{v}_t') - \nabla_{\boldsymbol{u}} F(\boldsymbol{u}_t, \boldsymbol{v}_t), \boldsymbol{u}_t' - \boldsymbol{u}_t\rangle$$
$$\overset{(b)}{\leq} \langle\nabla_{\boldsymbol{u}} F(\boldsymbol{u}_t, \boldsymbol{v}_t), \boldsymbol{u}_t' - \boldsymbol{u}_t\rangle + L_{uv}\|\boldsymbol{v}_t' - \boldsymbol{v}_t\|\|\boldsymbol{u}_t' - \boldsymbol{u}_t\|$$
$$\overset{(c)}{\leq} \langle\nabla_{\boldsymbol{u}} F(\boldsymbol{u}_t, \boldsymbol{v}_t), \boldsymbol{u}_t' - \boldsymbol{u}_t\rangle$$
$$+ \frac{1}{2}\kappa L_v\|\boldsymbol{v}_t' - \boldsymbol{v}_t\|^2 + \frac{1}{2}\kappa L_u\|\boldsymbol{u}_t' - \boldsymbol{u}_t\|^2, \tag{42}$$

where (a) is derived by adding and subtracting $\nabla_{\boldsymbol{u}} F(\boldsymbol{u}_t, \boldsymbol{v}_t)$, (b) follows the Cauchy-Schwarz inequality and Assumption 1, (c) is due to the definition of $\kappa = \max\{L_{uv}, L_{vu}\}/\sqrt{L_u L_v}$, which measures the relative cross-sensitivity of $\nabla_{\boldsymbol{u}} F(\boldsymbol{u}_t, \boldsymbol{v}_t)$ with respect to $\boldsymbol{v}_t$ and $\nabla_{\boldsymbol{v}} F(\boldsymbol{u}_t, \boldsymbol{v}_t)$ with respect to $\boldsymbol{u}_t$. Substituting (42) into (41), we have

$$\mathbb{E}[F(\boldsymbol{u}_t', \boldsymbol{v}_t') - F(\boldsymbol{u}_t, \boldsymbol{v}_t)]$$
$$\leq \langle\nabla_{\boldsymbol{u}} F(\boldsymbol{u}_t, \boldsymbol{v}_t), \boldsymbol{u}_t' - \boldsymbol{u}_t\rangle + \frac{\kappa + 1}{2}L_u\|\boldsymbol{u}_t' - \boldsymbol{u}_t\|^2$$
$$\langle\nabla_{\boldsymbol{v}} F(\boldsymbol{u}_t, \boldsymbol{v}_t), \boldsymbol{v}_t' - \boldsymbol{v}_t\rangle + \frac{\kappa + 1}{2}L_v\|\boldsymbol{v}_t' - \boldsymbol{v}_t\|^2. \tag{43}$$

For bounding $\mathbb{E}[\langle\nabla_{\boldsymbol{u}} F(\boldsymbol{u}_t, \boldsymbol{v}_t), \boldsymbol{u}_{t+1} - \boldsymbol{u}_t\rangle]$, we have

$$\mathbb{E}[\langle\nabla_{\boldsymbol{u}} F(\boldsymbol{u}_t, \boldsymbol{v}_t), \boldsymbol{u}_{t+1} - \boldsymbol{u}_t\rangle] \overset{(a)}{\leq} \frac{1}{2}\eta_u\sum_{l=0}^{L-1}\mathbb{E}[\|\nabla_{\boldsymbol{u}} F(\boldsymbol{u}_t, \boldsymbol{v}_t)$$
$$- \sum\nolimits_{i=1}^{I}\frac{1}{I}\nabla_{\boldsymbol{u}} F_i(\boldsymbol{w}_{i,t-\tau_{i,t},l})\|^2] - \frac{1}{2}\eta_u L\|\nabla_{\boldsymbol{u}} F(\boldsymbol{u}_t, \boldsymbol{v}_t)\|^2, \tag{44}$$

where (a) follows the triangle inequality. Now we turn to bound $\mathbb{E}[\|\nabla_{\boldsymbol{u}} F(\boldsymbol{u}_t, \boldsymbol{v}_t) - \sum_{i=1}^{I}\frac{1}{I}\nabla_{\boldsymbol{u}} F_i(\boldsymbol{w}_{i,t-\tau_{i,t},l})\|^2]$:

$$\mathbb{E}[\|\nabla_{\boldsymbol{u}} F(\boldsymbol{u}_t, \boldsymbol{v}_t) - \sum\nolimits_{i=1}^{I}\frac{1}{I}\nabla_{\boldsymbol{u}} F_i(\boldsymbol{w}_{i,t-\tau_{i,t},l})\|^2]$$
$$\overset{(a)}{\leq} L_u^2\sum\nolimits_{i=1}^{I}\mathbb{E}[\|\boldsymbol{u}_t - \boldsymbol{u}_{i,t-\tau_{i,t},l}\|^2]$$

$$\overset{(b)}{\leq} \frac{2L_u^2}{I}\sum_{i=1}^{I}\mathbb{E}[\|\boldsymbol{u}_t-\boldsymbol{u}_{t-\tau_{i,t}}\|^2] + 2L_u^2\sum_{i=1}^{I}\frac{1}{I}\mathbb{E}[\|\boldsymbol{u}_{t-\tau_{i,t}}-\boldsymbol{u}_{i,t-\tau_{i,t},l}\|^2], \quad (45)$$

where (a) follows Assumption 1 and Jensen's inequality, (b) is due to the inequality $\|a+b\|^2\leq 2\|a\|^2+2\|b\|^2$. Now bounding $\mathbb{E}[\|\boldsymbol{u}_{t-\tau_{i,t}}-\boldsymbol{u}_{i,t-\tau_{i,t},l}\|^2]$:

$$\begin{aligned}&\mathbb{E}[\|\boldsymbol{u}_t-\boldsymbol{u}_{i,t,l}\|^2]=\mathbb{E}[\|\boldsymbol{u}_{i,t,l-1}-\boldsymbol{u}_t-\eta_u\nabla_{\boldsymbol{u}}f_i(\boldsymbol{w}_{i,t,l-1})\|^2]\\ &\leq \mathbb{E}[\|\boldsymbol{u}_{i,t,l-1}-\boldsymbol{u}_t-\eta_u(\nabla_{\boldsymbol{u}}f_i(\boldsymbol{w}_{i,t,l-1})-\nabla_{\boldsymbol{u}}F_i(\boldsymbol{w}_{i,t,l-1})\\ &+\nabla_{\boldsymbol{u}}F_i(\boldsymbol{w}_{i,t,l-1})-\nabla_{\boldsymbol{u}}F_i(\boldsymbol{w}_t)+\nabla_{\boldsymbol{u}}F_i(\boldsymbol{w}_t))\|^2]\\ &\overset{(a)}{\leq}(1+\frac{1}{L-1})\mathbb{E}[\|\boldsymbol{u}_{i,t,l-1}-\boldsymbol{u}_t\|^2]+\eta_u^2\sigma_1^2\\ &+4L\eta_u^2\mathbb{E}[\|\nabla_{\boldsymbol{u}}F(\boldsymbol{w}_t)\|^2], \quad (46)\end{aligned}$$

where (a) follows Young's inequality, the inequality $\|a+b\|^2\leq 2\|a\|^2+2\|b\|^2$, Assumption 1-4, and $\eta_u\leq\frac{1}{8L_uL}$. Unrolling the recursion, we have

$$\begin{aligned}&\mathbb{E}[\|\boldsymbol{u}_t-\boldsymbol{u}_{i,t,l}\|^2]\leq\sum_{l'=0}^{l-1}(1+\frac{1}{L-1})^{l'}(\eta_u^2\sigma_1^2+4L\eta_u^2G_1^2)\\ &\leq(L-1)[(1+\frac{1}{L-1})^L-1](\eta_u^2\sigma_1^2+4L\eta_u^2G_1^2)\\ &\overset{(a)}{\leq}5L\eta_u^2\sigma_1^2+20L\eta_u^2G_1^2, \quad (47)\end{aligned}$$

where (a) is due to $(L-1)[(1+\frac{1}{L-1})^L-1]\leq 5L$. Hence, plug (40) and (47) into (45), then plug (45) into (44), we have

$$\begin{aligned}&\mathbb{E}[\langle\nabla_{\boldsymbol{u}}F(\boldsymbol{u}_t,\boldsymbol{v}_t),\boldsymbol{u}_{t+1}-\boldsymbol{u}_t\rangle]\leq\frac{3q_1LL_u^2\eta_u^3}{I}\sum_{i=1}^{I}\tau_{i,t}^2\\ &+10L^2\eta_u^3L_u^2G_1^2+\frac{5}{2}L^2\eta_u^3L_u^2\sigma_1^2-\frac{1}{2}\eta_uL\|\nabla_{\boldsymbol{u}}F(\boldsymbol{u}_t,\boldsymbol{v}_t)\|^2. \quad (48)\end{aligned}$$

For bounding $\frac{\kappa+1}{2}L_u\|\boldsymbol{u}_{t+1}-\boldsymbol{u}_t\|^2$, we have

$$\begin{aligned}&\frac{\kappa+1}{2}L_u\mathbb{E}[\|\boldsymbol{u}_{t+1}-\boldsymbol{u}_t\|^2]=\frac{L^2L_u\eta_u^2(\kappa+1)}{2}\\ &\times\mathbb{E}[\|\frac{1}{IL}\sum_{i=1}^{I}\sum_{l=0}^{L-1}\nabla_{\boldsymbol{u}}f_i(\boldsymbol{w}_{i,t-\tau_{i,t},l})\|^2], \quad (49)\end{aligned}$$

Now we bound $\mathbb{E}[\|\frac{1}{IL}\sum_{i=1}^{I}\sum_{l=0}^{L-1}\nabla_{\boldsymbol{u}}f_i(\boldsymbol{w}_{i,t-\tau_{i,t},l})\|^2]$:

$$\begin{aligned}&\mathbb{E}[\|\frac{1}{IL}\sum_{i=1}^{I}\sum_{l=0}^{L-1}\nabla_{\boldsymbol{u}}f_i(\boldsymbol{w}_{i,t-\tau_{i,t},l})\|^2]\\ &\overset{(a)}{\leq}\frac{2}{IL}\sum_{i=1}^{I}\sum_{l=0}^{L-1}\mathbb{E}[\|\nabla_{\boldsymbol{u}}F_i(\boldsymbol{w}_{i,t-\tau_{i,t},l})-\nabla_{\boldsymbol{u}}F(\boldsymbol{w}_t)\|^2]\\ &+2\mathbb{E}[\|\nabla_{\boldsymbol{u}}F(\boldsymbol{w}_t)\|^2]+\sigma_1^2, \quad (50)\end{aligned}$$

where (a) is due to Assumption 3 and $\|a+b\|^2\leq 2\|a\|^2+2\|b\|^2$, now we turn to bound the first term of (50)

$$\begin{aligned}&\frac{2}{IL}\sum_{i=1}^{I}\sum_{l=0}^{L-1}\mathbb{E}[\|\nabla_{\boldsymbol{u}}F_i(\boldsymbol{w}_{i,t-\tau_{i,t},l})-\nabla_{\boldsymbol{u}}F(\boldsymbol{w}_t)\|^2]\\ &\leq\frac{6}{IL}\sum_{i=1}^{I}\sum_{l=0}^{L-1}\eta_u^2\mathbb{E}[\|\boldsymbol{u}_{i,t-\tau_{i,t},l}-\boldsymbol{u}_{t-\tau_{i,t}}\|^2]\\ &+\frac{6}{I}\sum_{i=1}^{I}\eta_u^2\mathbb{E}[\|\boldsymbol{u}_{t-\tau_{i,t}}-\boldsymbol{u}_t\|^2]+6\delta_1^2. \quad (51)\end{aligned}$$

Substituting (51) into (50), then plug the result back to (49):

$$\begin{aligned}&\frac{\kappa+1}{2}L_u\mathbb{E}[\|\boldsymbol{u}_{t+1}-\boldsymbol{u}_t\|^2]\leq L^2L_u\eta_u^2(\kappa+1)\mathbb{E}[\|\nabla_{\boldsymbol{u}}F(\boldsymbol{w}_t)\|^2]\\ &+\frac{3LL_u\eta_u^2(\kappa+1)}{I}\sum_{i=1}^{I}\sum_{l=0}^{L-1}\eta_u^2\mathbb{E}[\|\boldsymbol{u}_{i,t-\tau_{i,t},l}-\boldsymbol{u}_{t-\tau_{i,t}}\|^2]\\ &+\frac{3LL_u\eta_u^2(\kappa+1)}{I}\sum_{i=1}^{I}\eta_u^2\mathbb{E}[\|\boldsymbol{u}_{t-\tau_{i,t}}-\boldsymbol{u}_t\|^2]\\ &+\frac{L^2L_u\eta_u^2(6\delta_1^2+\sigma_1^2)(\kappa+1)}{2}. \quad (52)\end{aligned}$$

Then turning to bound $\mathbb{E}[\langle\nabla_{\boldsymbol{v}}F(\boldsymbol{u}_t,\boldsymbol{v}_t),\boldsymbol{v}_{t+1}-\boldsymbol{v}_t\rangle]$:

$$\begin{aligned}&\mathbb{E}[\langle\nabla_{\boldsymbol{v}}F(\boldsymbol{u}_t,\boldsymbol{v}_t),\boldsymbol{v}_{t+1}-\boldsymbol{v}_t\rangle]\overset{(a)}{\leq}\frac{\eta_v\delta_2^2}{2}\\ &-\frac{\eta_v}{2}\mathbb{E}[\|\nabla_{\boldsymbol{v}}F(\boldsymbol{u}_t,\boldsymbol{v}_t)\|^2]-\frac{\eta_v}{2}\mathbb{E}[\|\nabla_{\boldsymbol{v}}F_i(\boldsymbol{u}_t,\boldsymbol{v}_t)\|^2], \quad (53)\end{aligned}$$

where (a) is due to Assumption 4, then we turn to bound $\frac{L_v(\kappa+1)}{2}\mathbb{E}[\|\boldsymbol{v}_{t+1}-\boldsymbol{v}_t\|^2]\leq L_v\eta_v^2(\kappa+1)\mathbb{E}[\|\nabla_{\boldsymbol{v}}F(\boldsymbol{u}_t,\boldsymbol{v}_t)\|^2]$

$$+\frac{L_v\eta_v^2\sigma_2^2(\kappa+1)}{2}+L_v\eta_v^2\delta_2^2(\kappa+1) \quad (54)$$

Hence we have

$$\begin{aligned}&\mathbb{E}[\langle\nabla_{\boldsymbol{v}}F(\boldsymbol{u}_t,\boldsymbol{v}_t),\boldsymbol{v}_{t+1}-\boldsymbol{v}_t\rangle]+\frac{L_v(\kappa+1)}{2}\mathbb{E}[\|\boldsymbol{v}_{t+1}-\boldsymbol{v}_t\|^2]\\ &\leq(L_v\eta_v^2(\kappa+1)-\frac{\eta_v}{2})\mathbb{E}[\|\nabla_{\boldsymbol{v}}F(\boldsymbol{u}_t,\boldsymbol{v}_t)\|^2]-\frac{\eta_vG_2^2}{2}\\ &+L_v\eta_v^2(\kappa+1)(\delta_2^2+\frac{\sigma_2^2}{2})+\frac{\eta_v\delta_2^2}{2}. \quad (55)\end{aligned}$$

Substituting (48), (52), and (55) back into (43), we have (21). The proof is completed.

C. Proof of Corollary 1

Subtracting $F(\boldsymbol{u}^*,\boldsymbol{v}^*)$ into both $F(\boldsymbol{u}_{t+1},\boldsymbol{v}_{t+1})$ and $F(\boldsymbol{u}_t,\boldsymbol{v}_t)$, then using L-smooth of loss function $\|\nabla_{\boldsymbol{u}}F(\boldsymbol{u}_t,\boldsymbol{v}_t)\|^2\leq 2L_u(F(\boldsymbol{u}_t,\boldsymbol{v}_t)-F(\boldsymbol{u}^*,\boldsymbol{v}^*))$ and $\|\nabla_{\boldsymbol{v}}F(\boldsymbol{u}_t,\boldsymbol{v}_t)\|^2\leq 2L_v(F(\boldsymbol{u}_t,\boldsymbol{v}_t)-F(\boldsymbol{u}^*,\boldsymbol{v}^*))$, we have

$$\begin{aligned}&\mathbb{E}[F(\boldsymbol{u}_{t+1},\boldsymbol{v}_{t+1})-F(\boldsymbol{u}^*,\boldsymbol{v}^*)]\leq\mathbb{E}[F(\boldsymbol{u}_t,\boldsymbol{v}_t)-F(\boldsymbol{u}^*,\boldsymbol{v}^*)]\\ &\times(1-\eta_uLL_u+2L^2L_u^2\eta_u^2(\kappa+1)-L_v\eta_v+2L_v^2\eta_v^2(\kappa+1))\\ &+\frac{3}{I}q_1LL_u\eta_u^3(L_u+3\eta_u(\kappa+1))\sum_{i=1}^{I}\tau_{i,t}^2+q_2. \quad (56)\end{aligned}$$

Since the staleness of each client i's local gradient evolves as

$$\tau_{i,t}=\begin{cases}\tau_{i,t-1}+1, & \text{if } \alpha_{i,t}^j=0,\\ 0, & \text{else,}\end{cases} \quad (57)$$

we have $\tau_{i,t}^2=(1-\alpha_{i,t}^j)^2(\tau_{i,t-1}+1)^2=(1-\alpha_{i,t}^j)(\tau_{i,t-1}+1)^2$. Consequently, we have

$$\begin{aligned}&\mathbb{E}[F(\boldsymbol{w}_{t+1})-F(\boldsymbol{w}^*)]\leq\mathbb{E}[F(\boldsymbol{w}_t)-F(\boldsymbol{w}^*)]\\ &\times(1-\eta_uLL_u+2L^2L_u^2\eta_u^2(\kappa+1)-L_v\eta_v+2L_v^2\eta_v^2(\kappa+1))\\ &+\frac{3}{I}q_1LL_u\eta_u^3(L_u+3\eta_u(\kappa+1))\\ &\times\sum_{i=1}^{I}(1-\alpha_{i,t}^j)(\tau_{i,t-1}+1)^2+q_2. \quad (58)\end{aligned}$$

By telescoping the above inequality, and let $\eta_u\leq\frac{1}{2LL_u(\kappa+1)}$ and $\eta_v\leq\frac{1}{2L_v(\kappa+1)}$, we have (22), the proof is complete.

REFERENCES

[1] C. Xie, Z. Chen, W. Yi, H. Shin, and A. Nallanathan, "Accelerating wireless hierarchical split federated learning via smashed data generation," in *Submitted in IEEE 37th Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Singapore, 2026.

[2] M. A. Ferrag, O. Friha, B. Kantarci, N. Tihanyi, L. Cordeiro, M. Debbah, D. Hamouda, M. Al-Hawawreh, and K.-K. R. Choo, "Edge learning for 6g-enabled internet of things: A comprehensive survey of vulnerabilities, datasets, and defenses," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 4, pp. 2654–2713, 2023.

[3] "Worldwide IDC Global DataSphere Forecast, 2025–2029," International Data Corporation, Tech. Rep., May, 2025. [Online]. Available: https://my.idc.com/getdoc.jsp?containerId=US53363625

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proc. AISTATS*, 20–22, Apr. 2017.

[5] G. Team, M. Riviere, S. Pathak, P. G. Sessa, C. Hardin, S. Bhupatiraju, L. Hussenot, T. Mesnard, B. Shahriari, A. Ramé *et al.*, "Gemma 2: Improving open language models at a practical size," *arXiv preprint arXiv:2408.00118*, 2024.

[6] G. Qu, Q. Chen, W. Wei, Z. Lin, X. Chen, and K. Huang, "Mobile edge intelligence for large language models: A contemporary survey," *IEEE Commun. Surveys Tuts.*, 2025.

[7] C. Thapa, M. A. P. Chamikara, S. Camtepe, and L. Sun, "Splitfed: When federated learning meets split learning," 2022. [Online]. Available: https://arxiv.org/abs/2004.12088

[8] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," 2018. [Online]. Available: https://arxiv.org/abs/1812.00564

[9] P. Li, G. Cheng, X. Huang, J. Kang, R. Yu, Y. Wu, M. Pan, and D. Niyato, "Snowball: Energy efficient and accurate federated learning with coarse-to-fine compression over heterogeneous wireless edge devices," *IEEE Trans. Wireless Commun.*, vol. 22, no. 10, pp. 6778–6792, 2023.

[10] Z. Chen, W. Yi, H. Shin, and A. Nallanathan, "Adaptive model pruning for communication and computation efficient wireless federated learning," *IEEE Trans. Wireless Commun.*, pp. 1–1, 2023.

[11] C. Xie, Z. Chen, W. Yi, H. Shin, and A. Nallanathan, "Tackling class imbalance and client heterogeneity for split federated learning in wireless networks," *IEEE Trans. Wireless Commun.*, 2025.

[12] N. Huang, M. Dai, Y. Wu, T. Q. Quek, and X. Shen, "Wireless federated learning with hybrid local and centralized training: A latency minimization design," *IEEE J. Sel. Topics Signal Process.*, vol. 17, no. 1, pp. 248–263, 2022.

[13] T. Xia, Y. Deng, S. Yue, J. He, J. Ren, and Y. Zhang, "Hsfl: An efficient split federated learning framework via hierarchical organization," in *2022 18th International Conference on Network and Service Management (CNSM)*. IEEE, 2022, pp. 1–9.

[14] 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, 2023.

[15] Z. Lin, W. Wei, Z. Chen, C.-T. Lam, X. Chen, Y. Gao, and J. Luo, "Hierarchical split federated learning: Convergence analysis and system optimization," *IEEE Trans. Mobile Comput.*, 2025.

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

[17] J. Nguyen, K. Malik, H. Zhan, A. Yousefpour, M. Rabbat, M. Malek, and D. Huba, "Federated learning with buffered asynchronous aggregation," in *Proc. Conf. Artif. Intell. Statist. (AISTATS)*, 2022, pp. 3581–3607.

[18] H.-S. Lee and J.-W. Lee, "Adaptive transmission scheduling in wireless networks for asynchronous federated learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3673–3687, 2021.

[19] Z. Wang, Z. Zhang, Y. Tian, Q. Yang, H. Shan, W. Wang, and T. Q. Quek, "Asynchronous federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, vol. 21, no. 9, pp. 6961–6978, 2022.

[20] C.-H. Hu, Z. Chen, and E. G. Larsson, "Scheduling and aggregation design for asynchronous federated learning over wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 874–886, 2023.

[21] W. Wu, L. He, W. Lin, R. Mao, C. Maple, and S. Jarvis, "Safa: A semi-asynchronous protocol for fast federated learning with low overhead," *IEEE Trans. Comput.*, vol. 70, no. 5, pp. 655–668, 2020.

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

[23] J. Liu, H. Xu, L. Wang, Y. Xu, C. Qian, J. Huang, and H. Huang, "Adaptive asynchronous federated learning in resource-constrained edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 2, pp. 674–690, 2021.

[24] Z. Chen, W. Yi, H. Shin, and A. Nallanathan, "Adaptive semi-asynchronous federated learning over wireless networks," *IEEE Trans. Commun.*, 2024.

[25] S. Luo, X. Chen, Q. Wu, Z. Zhou, and S. Yu, "Hfel: Joint edge association and resource allocation for cost-efficient hierarchical federated edge learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6535–6548, 2020.

[26] X. Xu, M. Yang, W. Yi, Z. Li, J. Wang, H. Hu, Y. Zhuang, and Y. Liu, "A stealthy wrongdoer: Feature-oriented reconstruction attack against split learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2024, pp. 12 130–12 139.

[27] J. Liu, X. Lyu, Q. Cui, and X. Tao, "Similarity-based label inference attack against training and inference of split learning," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 2881–2895, 2024.

[28] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 3, pp. 1935–1949, 2021.

[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, 2020.

[30] M. Neely, *Stochastic network optimization with application to communication and queueing systems*. Morgan & Claypool Publishers, 2010.

[31] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[32] 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, 2023.

[33] Z. Chen, W. Yi, Y. Liu, and A. Nallanathan, "Knowledge-aided federated learning for energy-limited wireless networks," *IEEE Trans. Commun.*, vol. 71, no. 6, pp. 3368–3386, 2023.

[34] C. Huang, G. Tian, and M. Tang, "When minibatch sgd meets splitfed learning: Convergence analysis and performance evaluation," *arXiv preprint arXiv:2308.11953*, 2023.

[35] D. Yan, M. Hu, Z. Xia, Y. Yang, J. Xia, X. Xie, and M. Chen, "Have your cake and eat it too: Toward efficient and accurate split federated learning," *arXiv preprint arXiv:2311.13163*, 2023.