Tackling Class Imbalance and Client Heterogeneity for Split Federated Learning in Wireless Networks

Chunfeng Xie, Zhixiong Chen, *Member, IEEE*, Wenqiang Yi, *Member, IEEE*, Hyundong Shin, *Fellow, IEEE*, and Arumugam Nallanathan, *Fellow, IEEE*

***Abstract*—As the complexity of deep neural networks escalates, traditional federated learning (FL) frameworks increasingly struggle since the training overhead of the full model is costly for resource-limited clients. In addition, the class imbalance among local datasets and client heterogeneity may lead to significant deterioration in learning performance. To address these challenges, we first propose a novel wireless split federated learning (SFL) framework to enhance learning efficiency and performance in resource-constrained networks, which adaptively splits the global model between the clients and server to alleviate the computation burden for clients. Then, we theoretically analyze how the client sampling and wireless network parameters impact on the convergence bound. Based on the analysis, we identify the extent of class imbalance that significantly impacts learning performance. Inspired by this, we formulate an optimization problem to strike a balance between latency and performance by jointly optimizing the client selection, model splitting, and bandwidth allocation policies. To solve this problem, we introduce a latency and class imbalance-aware double greedy algorithm to obtain client scheduling policy. Additionally, bisection-enabled optimal bandwidth allocation and model splitting algorithms are developed to adaptively determine bandwidth allocation and model splitting policies, respectively. Extensive experimental results demonstrate that our approach significantly reduces latency and enhances learning performance.**

***Index Terms*—Split federated learning, resource allocation, client sampling, model splitting.**

I. Introduction

With the exponential growth of data at network edges, a variety of advanced machine learning (ML) techniques are increasingly being implemented for facilitating versatile applications, such as autonomous driving, the metaverse [2], and the Internet of Things (IoT) [3]. However, prevailing ML frameworks predominantly focus on centralized learning architectures, which often result in significant delays [4] and resource consumption. In response, federated learning (FL) [5], a distributed learning framework, is developed as a solution to mitigate aforementioned issues. FL permits edge clients to collaboratively train a global model while maintaining data privacy [6]. Despite its advantages, the application of FL in wireless networks meet challenges. Notably, FL is hindered by substantial client communication and computation burdens, attributed to the limited computing and communication capacities of the clients. This complication arises because large-sized models are transmitted, and the computation-intensive training process is confined solely to the clients. To address the aforementioned challenges in FL, another distributed learning framework called split learning (SL) [7] partitions models into sub-models, while the server-side sub-models are trained independently on server and the client-side sub-models are trained on distributed clients using local data. This framework leverages the servers superior computational and communication capacities to offload processing tasks, thereby alleviating the burden on clients. However, the standard sequential training method employed by SL can result in underutilization of client resources. Additionally, when multiple clients are involved, the training time overhead can escalate significantly.

Split federated learning (SFL) [8] is a promising approach that aims to mitigate the challenges inherent in both FL and SL. This approach has been gaining attention among researchers who are increasingly exploring its applications. For instance, SFL is applied in healthcare analytics [9] to minimize reliance on local hardware, facilitate semi-distributed learning, and enhance privacy protection. It is worth noting that this paper primarily focuses image classification tasks under supervised learning. Other approaches such as self-supervised learning [10], transfer learning [11], [12] , and adversarial learning [13], [14] can also be applied to SFL and will be explored in future work. Despite its advantages, the deployment of SFL in real-world wireless networks faces three primary challenges: 1) *Limited Wireless Resources*: Bandwidth is a critical resource in wireless communication networks, especially in scenarios involving intensive data exchanges, such as during the training of machine learning models. Inefficient use of bandwidth not only extends the training duration but also increases the energy consumption and operational costs, particularly affecting scalability and sustainability of machine learning deployments in large-scale networks. 2) *Data Distribution Heterogeneity*: In practical scenarios, the distribution of data across clients is not uniform, where clients may have abundant examples of certain classes while lacking in others. This imbalance can lead to biased models that perform well on data-rich classes but poorly on underrepresented [15] ones. When data is imbalanced, the model's ability to generalize decreases, potentially compromising the accuracy and fairness of the outcomes. As a result, it may exacerbate performance degradation due to class imbalance among the selected clients [16]. 3) *Clients*

Part of this work was submitted in IEEE Globecom, 2024 [1].

Chunfeng Xie, Zhixiong Chen and Arumugam Nallanathan are with the School of Electronic Engineering and Computer Science, Queen Mary University of London, E1 4NS London, U.K. (emails: {c.xie, zhixiong.chen, a.nallanathan}@qmul.ac.uk).

W. Yi is with the School of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, U.K. (email: wy23627@essex.ac.uk).

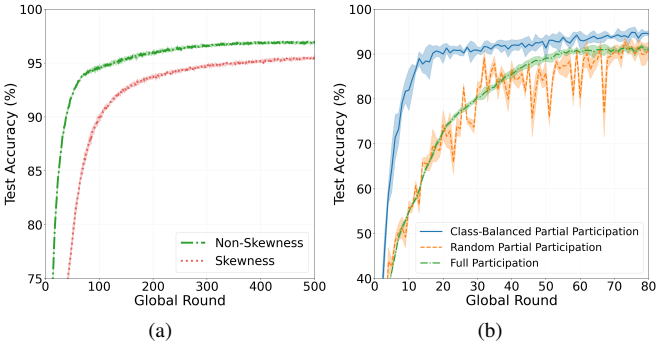
Hyundong Shin is with the Department of Electronics and Information Convergence Engineering, Kyung Hee University, Yongin-si, Gyeonggido 17104, Republic of Korea (e-mail: hshin@khu.ac.kr).

Heterogeneity: Clients exhibit significant disparities in computational and communication capabilities [17]. The variance in capabilities can lead to bottlenecks in the distributed learning process. Previous wireless SFL studies often focused on a fixed model cut layer configuration, requiring all clients to train identical models in each round. Consequently, clients with limited capabilities can delay overall aggregation and slow the learning process. To overcome these issues, the latest innovative strategies primarily focus on enhancing dynamic resource allocation and class-balanced client sampling.

Extensive research has focused on addressing the challenge of limited resources, incorporating strategies such as client sampling [18], [19] and model compression [20]. In [18], the client sampling approach determine a subset of clients to participate in the learning of each round, which helps alleviate the burden of communication. Research in [19] significantly mitigated the energy trade-off by exploring integrated device scheduling and resource management strategies. While client sampling efficiently alleviates the communication load in wireless FL, clients with suboptimal channel conditions face challenges in uploading complete models for aggregation. A model compression technique described in [20] introduced a gradient sparsification framework that reduces gradient dimensions to conserve bandwidth. However, despite its contribution to conserve communication resources, model compression introduces additional noise during model aggregation, potentially impacting learning performance, while the issue of high model complexity persists in these approaches, resulting in significant computation and time consumption for the clients. Additionally, the majority of these studies focus on FL, research on resource-limited wirless networks for SFL remains insufficient.

A limited number of studies addressed the problem of datasets class imbalance. Recent works [21]–[23] has underscored the problem of class imbalance resulting from random client sampling in non-IID conditions. Specifically, [21] describes a probabilistic client sampling algorithm designed to assemble groups of datasets with the minimal degree of class imbalance, while [22] develops a method to estimate the label distribution of client's dataset by analyzing gradients of model parameters. In contrast, [23] explores a novel model aggregation framework that assigns greater weight to clients with lower class imbalance, aiming to formulate a more balanced global dataset. However, these estimations of clients label distributions tend to be inaccurate, while these studies have not sufficiently considered the associated communication resource costs. Hence, in SFL networks, no studies address the issue of high skewness, which refers to class-imbalance. A high skewness value typically indicates a substantial class imbalance, which can significantly impair the learning performance of ML models.

To demonstrate the impact of skewness, Fig. 1 provides a comparative analysis of learning performance on the MNIST dataset. Specifically, Fig. 1(a) illustrates that the learning performance is substantially better on the dataset without skewness. Additionally, Fig. 1(b) examines the differences in learning outcomes between partial participation (10 clients) and full participation (100 clients) within an imbalanced



(a) (b)

Fig. 1. Learning performance comparison on the MNIST dataset for demonstrating the impact of skewness: (a) Comparison of datasets with and without Skewness; (b) Comparison of partial and full clients participation.

dataset. The results indicate that the scheduled non-skewness client subset achieves higher test accuracy compared to both the randomly scheduled subset and full participation.

To accommodate the training of client-side models tailored to the diverse communication and computation capabilities of various clients, model splitting approaches are developed. These approaches in [24]–[26] are designed to minimize resource demands on clients. The approach in [24] introduces a parallel computing scheme that reduces computational idleness by offloading workloads from the clients to the server, which effectively addresses the issue of resource scarcity and reduces latency compared to traditional SL. [25] explores the impact of global aggregation frequency on learning performance and introduces a method for cut layer selection. Additionally, [26] presents a probabilistic approach to optimal cut layer selection. Nonetheless, existing works typically have not investigated dynamic cut layer selection for every round, which accommodates changes in the time-varying channel or the dynamic availability of communication resources.

Previous studies [18]–[20] effectively addressed communication bottlenecks in wireless networks, yet they often incur high communication overhead for clients with limited resources. These methods might also preclude clients with poor channel conditions or low transmit power from participating in training, thereby reducing learning performance. Moreover, current client scheduling methods [21]–[23] seldom account for conditions with limited wireless resources. Additionally, prevalent model splitting techniques [24]–[26] often overlook the heterogeneity of clients' capabilities and channel variations, opting instead to maintain a uniform cut layer across all rounds, which may detrimentally affect learning performance. Furthermore, these studies primarily focus on FL or SL, frequently neglecting SFL. To enhance SFL learning performance and conserve wireless resources, we develop a novel framework, which includes a dynamic model splitting scheme that adjusts the cut layer based on each client's computing and communication capabilities, thereby optimizing learning latency. To tackle heterogeneous data distribution, we introduce a skewness-aware client sampling scheme that selects a subset of clients with approximately balanced datasets for training, alongside a latency-aware sampling scheme that prioritizes clients with low latency. Furthermore, we introduce

an algorithm dynamically allocates bandwidth to accommodate clients with limited transmit power and time-varying channel conditions during each global training round. By integrating these strategies, a latency and skewness-aware (L&S-aware) algorithm is proposed to enhance learning efficiency and reduce latency. By addressing three major challenges, the proposed framework not only enhances learning efficiency but also promotes a more equitable and balanced participation among clients, irrespective of their individual resource capabilities. Furthermore, while our study initially focuses on SFL, the principles underlying our optimized algorithms have broader applicability across various forms of distributed learning systems within wireless communication networks. The principal contributions of this paper are outlined as follows:

- To optimize SFL in resource-constrained wireless networks, we analyze the convergence bounds for the L&S-aware client sampling framework with a non-convex loss function and introduce a novel metric. This metric elucidates how the degree of class imbalance in clients' datasets negatively impacts learning performance, and demonstrates that minimizing this imbalance enhances SFL learning.
- To address the challenges of limited resources and heterogeneous client capabilities, we propose adaptive bandwidth allocation and a dynamic cut layer selection scheme. These allow clients under poor channel conditions to adaptively gain additional bandwidth, reducing idle time and accelerating the learning process. Furthermore, we dynamically adjust model split points in each global round to better accommodate the time-varying environment and the computational and communication capacities of clients.
- To leverage the benefits of skewness-aware and latency-aware strategies, we develop an L&S-aware algorithm that boosts learning performance and reduces overhead. This algorithm involves jointly optimizing client sampling, dynamic model splitting, and adaptive bandwidth allocation for minimizing the weighted sum of skewness degree and latency. To the best of our knowledge, this study is the first in SFL to characterize the effects of datasets' skewness and dynamic model split points.
- To validate the efficacy and robustness of our algorithms, we conducted extensive simulations to assess their performance across four distinct datasets: MNIST, Fashion-MNIST, CIFAR-10, and STL-10. Our results clearly demonstrate that the proposed algorithms significantly outperform the baselines in terms of both convergence speed and test accuracy.

The rest of this paper is organized as follows: In section II, we introduce the SFL system model and learning latency model. In section III, we present the convergence analysis of the SFL framework and problem formulation. Section IV introduces an optimal bandwidth allocation algorithm and an optimal model splitting algorithm for adaptive bandwidth allocation and dynamic model split, along with a L&S-aware algorithm. Simulations in Section V validate the proposed algorithms. In Section VI, we conclude the paper. The main notations used in this paper are summarized in Table I.

TABLE I
NOTATION SUMMARY

Notation	Definition
$\mathcal{U}; U$	Set of clients; size of $\mathcal{U}$
$\mathcal{M}; M$	Subset of clients; size of $\mathcal{M}$
$\mathcal{D}_u; D_u$	Local dataset of client u; size of $\mathcal{D}_u$
$\mathcal{D}; D$	Overall dataset in the system; size of $\mathcal{D}$
$Q; \boldsymbol{w}$	Number of classes; model parameter
$\boldsymbol{w}_{u,i}^{\mathrm{c}}; \boldsymbol{w}_{u,i}^{\mathrm{s}}$	Client/server-side model of client u in round i
$L(\boldsymbol{w}); l_u(\boldsymbol{x}, y)$	Average loss function with model $\boldsymbol{w}$; Loss function for data sample $(\boldsymbol{x}, y)$
$\eta_c; \eta_s$	Learning rate for client/server-side models
$\mathcal{I}; \mathcal{J}$	Set of global round; Set of local iteration
$\boldsymbol{\alpha}_i; \alpha_{i,j}$	Client sampling decision in round i; scheduling indicator of client u in round i
$s_{u,i}; \mathcal{S}$	Cut layer of client u in round i; set of cut layer
$f_u; f_s$	CPU frequency of client u; CPU frequency of the server
$p_u; p_s$	Transmit power of client u; transmit power of the server
$\beta; \boldsymbol{n}_i$	Wireless bandwidth; the proportion of β allocated to clients in round i
$\vartheta_u^{\mathrm{s}}, \vartheta_u^{\mathrm{c}}$	Smashed data size for one sample; data size of client-side model
$\phi_u^{\mathrm{F,c}}, \phi_u^{\mathrm{B,c}}$	Computing workload of client u in FP/BP
$\phi_u^{\mathrm{F,s}}, \phi_u^{\mathrm{B,s}}$	Computing workload of the server in FP/BP
$\boldsymbol{a}_u, \mathcal{G}(\mathcal{M})$	Local label distribution of client u; Skewness of client subset $\mathcal{M}$'s data
$\nabla l(\boldsymbol{w})$	Loss function's gradient

II. System Model and Learning Mechanism

In this paper, we adopt the SFL in wireless network, as shown in Fig 2, where one server and U clients collaboratively learn a global model $\boldsymbol{w}$. Clients are indexed by $\mathcal{U} = \{1, 2, \cdots, U\}$. Every client u ($u \in \mathcal{U}$) possesses a local dataset with D_u samples, i.e., $\mathcal{D}_u = \{\boldsymbol{x}_{u,i}, y_{u,i}\}_{i=1}^{D_u}$, where $\boldsymbol{x}_{u,i} \in \mathbb{R}^O$ represents the O-dimensional vector of input data, and $y_{u,i} \in \mathbb{R}$ represents the associated label. The overall dataset is represented by $\mathcal{D} = \sum_{u\in\mathcal{U}} \mathcal{D}_u$. Let $l(\boldsymbol{x}_{u,i}, y_{u,i}; \boldsymbol{w})$ represent the sample-wise loss function, it measures the fitting performance of model $\boldsymbol{w}$ on the data pair $(\boldsymbol{x}_{u,i}, y_{u,i})$. Thus, the local loss function of client u is given by $L_u(\boldsymbol{w}) = \frac{1}{|\mathcal{D}_u|} \sum_{\{\boldsymbol{x}_{u,i}, y_{u,i}\}\in\mathcal{D}_u} l_u(\boldsymbol{x}_{u,i}, y_{u,i}; \boldsymbol{w})$.

$L(\boldsymbol{w})$ represents the global loss function, which is defined as the weighted average of clients' local loss functions, the function of $L(\boldsymbol{w})$ is given by

$$L(\boldsymbol{w}) = \frac{\sum_{u\in\mathcal{U}} |\mathcal{D}_u| L_u(\boldsymbol{w})}{\sum_{u\in\mathcal{U}} |\mathcal{D}_u|}. \tag{1}$$

The goal of SFL system is to find a global model $\boldsymbol{w}$ with the aim of minimizing the global loss function $L(\boldsymbol{w})$ on the entire dataset $\mathcal{D}$, i.e., $\min_{\boldsymbol{w}} L(\boldsymbol{w})$. Note that the server and clients collaboratively train the global model without sharing the local raw data at each client.

A. SFL with Optimal Sampling and Model Split

Traditional SFL encounters significant challenges, including performance degradation stemming from the class imbalance

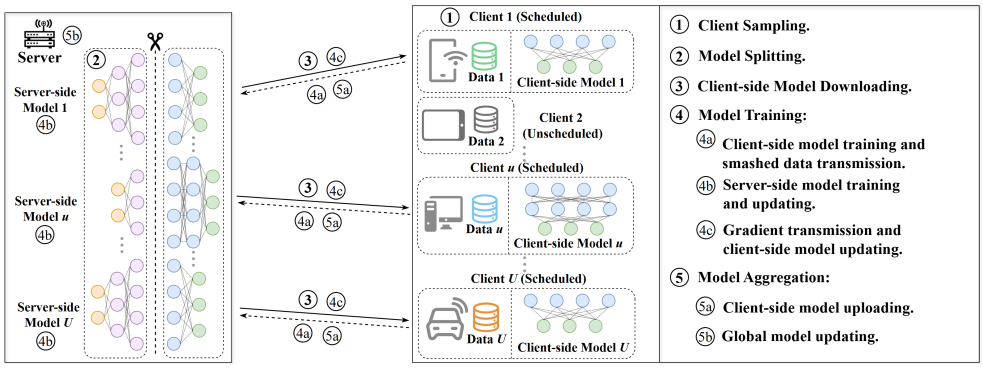


Fig. 2. Illustration of the designed wireless SFL system.

among clients' datasets and the heterogeneous communication and computation capabilities among clients. To tackle the previously mentioned challenges, we propose a novel SFL framework for efficiently sampling clients to participate in training, thereby mitigating the issue of skewness. The skewness is calculated locally by each client based solely on the distribution of labels within their own dataset, while it also provides the server with necessary information to assess the degree of imbalance without compromising the underlying data privacy, since it restricts the server or any other entities to reconstruct or infer the original label distribution of the client's data. In addition, the server dynamically splits the full model to formulate the sub-models to adapt the different computing capabilities of clients and dynamic channel characteristics. The learning process contains I global rounds and executes the subsequent steps in each round i $(i \in 0, 1, \cdots, I)$:

1) **Client Sampling**: A subset of clients $\mathcal{M}_i \subseteq \mathcal{U}$ is determined for taking part in the present round of training. $\alpha_{u,i} \in \{0, 1\}$ represent the scheduling parameter of client u in the i-th global training round, where $\alpha_{u,i} = 1$ indicates client u is scheduled, $\alpha_{u,i} = 0$ otherwise. For simplicity of presentation, let $\boldsymbol{\alpha}_i = \{\alpha_{1,i}, \alpha_{2,i}, \cdots, \alpha_{U,i}\}$ denote the client sampling decision in global round i.

2) **Model Split**: For adapting clients' heterogenous capabilities and channel condition of current round, the dynamic cut layer selection scheme determines the personalized cut layer for each client's model in every global round dynamically. Let $s_{u,i} \in \mathcal{S} = \{2, 3, \cdots, S\}$ denote cut layer decision for client u in i-th round, where S is the model's total layers. For ease of presentation, let $\boldsymbol{s}_i = \{s_{1,i}, s_{2,i}, \cdots, s_{u,i}\}$ denote the collection of cut layer decision for all clients in round i. Notably, due to privacy preservation consideration for client's raw data, the split point cannot be selected at the input layer and when the cut layer $s_{u,i} = S$, SFL system turns into FL.

3) **Client-side Model Downloading**: The server transmits the initial client-side model, denoted by $\boldsymbol{w}^{\mathrm{c}}_{u,i}$. Each sub-model is trained for J local iterations, indexed by $j \in \mathcal{J} = \{1, \cdots, j, \cdots, J\}$. Let $\boldsymbol{w}^{\mathrm{c}}_{u,i,j}$ denote the client-side model parameters for client u at iteration j in global round i. Hence, the initial client-side model is given by $\boldsymbol{w}^{\mathrm{c}}_{u,i,0} \leftarrow \boldsymbol{w}^{\mathrm{c}}_{u,i}$, while the corresponding server-side model is given by $\boldsymbol{w}^{\mathrm{s}}_{u,i,0} \leftarrow \boldsymbol{w}^{\mathrm{s}}_{u,i}$.

4) **Model Training**: The training process includes forward propagation (FP) stage and backward propagation (BP) stage. The FP stage includes client-side FP, smashed data transmission, and server-side FP. For client-side FP, each scheduled client randomly drawing a mini-batch of data samples $\mathcal{B}_{u,i,j} \subseteq \mathcal{D}_u$ from its dataset, where $B = |\mathcal{B}_{u,i,j}|$ is the mini-batch size. Let $\boldsymbol{X}_{u,i,j} \in \mathbb{R}^{B \times O}$ denote the mini-batch data samples' input vectors in client u. Subsequently, each client runs its individual client-side model using the selected data samples and generates smashed data $\boldsymbol{A}_{u,i,j}$, namely,

$$\boldsymbol{A}_{u,i,j} = f(\boldsymbol{X}_{u,i,j}; \boldsymbol{w}^{\mathrm{c}}_{u,i,j}), \forall u \in \mathcal{U}, \forall j \in \mathcal{J}, \tag{2}$$

where $f(\boldsymbol{x}; \boldsymbol{w})$ represents the mapping function from input data $\boldsymbol{x}$ to its predicted value, determined by model $\boldsymbol{w}$. Then each client sends its smashed data to the server for processing on the server side. After proceeding the smashed data to server-side model, we have the predicted result is given by

$$\hat{\boldsymbol{y}}_{u,i,j} = f(\boldsymbol{A}_{u,i,j}; \boldsymbol{w}^{\mathrm{s}}_{u,i,j}), \forall u \in \mathcal{U}, \forall j \in \mathcal{J}, \tag{3}$$

In BP stage, firstly, the average gradient of the loss function $\nabla l(\boldsymbol{w})$ is calculated with the predicted results and the corresponding true labels. Subsequently, the server-side model is updated through the application of the stochastic gradient descent (SGD) method:

$$\boldsymbol{w}^{\mathrm{s}}_{u,i,j+1} \leftarrow \boldsymbol{w}^{\mathrm{s}}_{u,i,j} - \eta_s \nabla l(\boldsymbol{w}^{\mathrm{s}}_{u,i,j}), \forall u \in \mathcal{U}, \forall j \in \mathcal{J}, \tag{4}$$

where η_s denotes the learning rate for the server-side model. The model parameters are updated layer-wise, starting from the final layer and progressing to the cut layer, following the chain rule for gradient calculation. Subsequently, once the gradient computation reaches the cut layer, the gradient of the smashed data for a mini-batch of data samples is transmitted back to the respective client. Finally, each sub-model of client is updated after receiving the corresponding smashed data's gradient:

$$\boldsymbol{w}^{\mathrm{c}}_{u,i,j+1} \leftarrow \boldsymbol{w}^{\mathrm{c}}_{u,i,j} - \eta_c \nabla l(\boldsymbol{w}^{\mathrm{c}}_{u,i,j}), \forall u \in \mathcal{U}, \forall j \in \mathcal{J}, \tag{5}$$

where η_c represents the client-side learning rate.

5) **Model Aggregation**: After completing J local iterations, the sub-models of clients are uploaded to the server. Along with the server-side models, these sub-models are aggregated to form a new global model. This aggregation process involves aggregating models from the first to the last layer, as expressed

by the equation:

$$\boldsymbol{w}_{i+1,s} = \frac{\sum_{u\in\mathcal{U}} |\mathcal{D}_u| \boldsymbol{w}_{u,i,s}}{\sum_{u\in\mathcal{U}} |\mathcal{D}_u|}, \tag{6}$$

where $\boldsymbol{w}_{u,i,s}$ denotes the weights of the s-th layer from the model of client u in the i-th round.

B. Latency Model

In this subsection, we present the communication and computation latency by taking the i-th round as an example.

1) **Communication Latency**: This work employs frequency division multiple access (FDMA) technology with a wireless bandwidth of β Hz for clients to transmit smashed datas and sub-models. Define $n_{u,i} \in [0, 1]$ as the bandwidth allocated to client u in round i, and $\boldsymbol{n}_i = (n_{1,i}, \cdots, n_{u,i}, \cdots, n_{U,i})$. For uplink transmission, let p_u represent the transmit power of client u. We postulate that the channel gain $h_{u,i}$ between client u and the server remains steady within a single round but changes independently across rounds, incorporating small-scale fading and path loss. Thus, client u's transmit rate is

$$r_{u,i}(n_{u,i}) = n_{u,i}\beta \log_2(1 + \frac{p_u h_{u,i}}{\sigma^2}), \tag{7}$$

where σ^2 denote the variance of Gaussian additive noise. Note that downlink communication latency is negligible because of the large transmit power of server. Let $\vartheta_u^{\mathrm{s}}(s_u)$ and $\vartheta_u^{\mathrm{c}}(s_u)$ represent the smashed data size for one sample and data size of client-side model, respectively. Hence, the latency of smashed data transmission is $t_{u,j}^{\mathrm{SDT}} = \frac{B\vartheta_u^{\mathrm{s}}(s_u)}{r_u(n_u)}$. Similarly, the latency of model uploading is $t_u^{\mathrm{MT}} = \frac{\vartheta_u^{\mathrm{c}}(s_u)}{r_u(n_u)}$. Thus, the overall communication latency for one round is

$$T_u^{\mathrm{CM}} = \sum\nolimits_{j=1}^{J} t_{u,j}^{\mathrm{SDT}} + t_u^{\mathrm{MT}}, \forall u \in \mathcal{U}. \tag{8}$$

Note that the latency of model distribution and gradients transmission are negligible since the server possesses high transmit power.

2) **Computing Latency**: The clients conduct client-side FP and sub-model update, while the server conducts server-side FP and sub-model update. In FP stage, the latency of client-side model training stage is $t_{u,j}^{\mathrm{CMT}} = \frac{B\phi_u^{\mathrm{F,c}}(s_u)}{f_u}$, where $\phi_u^{\mathrm{F,c}}(s_u)$ represents the computational workload (in FLOPs) of the client-side model's FP stage for processing a single data sample, and f_u represents the computing capability of client u, which is quantified as the number of floating-point operations per second (flops/s). Similarly, the latency of server-side model training is $t_{u,j}^{\mathrm{SMT}} = \frac{B\sum_{u\in\mathcal{M}} \phi_u^{\mathrm{F,s}}(s_u)}{f_s}$, where $\sum_{u\in\mathcal{M}} \phi_u^{\mathrm{F,s}}(s_u)$ denote the computation workload of the server, and f_s denote the computing capability of the server (flops/s). In BP stage, let $\phi_u^{\mathrm{B,c}}(s_u)$ and $\sum_{u\in\mathcal{M}} \phi_u^{\mathrm{B,s}}(s_u)$ denote the workload of client-side model updates and server-side model updates, respectively. Hence, we have latency of client-side and server-side model update $t_{u,j}^{\mathrm{CMU}} = \frac{B\phi_u^{\mathrm{B,c}}(s_u)}{f_c}$ and $t_{u,j}^{\mathrm{SMU}} = \frac{B\sum_{u\in\mathcal{M}} \phi^{\mathrm{B,s}}(s_u)}{f_s}$, respectively.
Counting these components, the computing latency of i-th round is

$$T_u^{\mathrm{CP}} = \sum_{j=1}^{J} (t_{u,j}^{\mathrm{CMT}} + t_{u,j}^{\mathrm{SMT}} + t_{u,j}^{\mathrm{SMU}} + t_{u,j}^{\mathrm{CMU}}), \forall u \in \mathcal{U}. \tag{9}$$

With the latency components we analyzed, the latency of one global round is given by

$$\mathcal{T}(\mathcal{M}) = \max_{u\in\mathcal{M}} \{T_u^{\mathrm{CM}} + T_u^{\mathrm{CP}}\}, \tag{10}$$

where $T(\mathcal{M})$ denote the highest latency among scheduled clients in i-th round. Note that the latency of model aggregation is negligible because of the low computing complexity. Note that packet loss issues are not considered in this work. In future research, we will explore methods to enhance the robustness of the framework, such as the application of Hybrid Automatic Repeat reQuest (HARQ).

III. Convergence Analysis and Problem Formation

In this section, we conceptually describe the convergence characteristics of the studied SFL system under the broad non-convex loss function framework. Subsequently, we identify a metric, specifically the degree of class imbalance, to help the design of the client sampling policy. Furthermore, we construct an optimization problem for client sampling that balances the trade-off between dataset skewness and latency in each round.

A. Convergence Analysis

We examine the convergence of the introduced framework and adopt several assumptions in this subsection.

Our global objective function $L(\boldsymbol{w}) > 0$ can be split as $L(\boldsymbol{w}) = \frac{1}{Q}\sum_{q=1}^{Q} L_q(\boldsymbol{w})$, where $L_q(\boldsymbol{w})$ represents the average loss function corresponding to all data from the q-th class in the global dataset, and Q is the number of classes in an image classification task. Similarly, the local objective function for the u-th client $L_u(\boldsymbol{w})$ can be expressed as $L_u(\boldsymbol{w}) = \sum_{q=1}^{Q} a_{(u,q)} L_{(u,q)}(\boldsymbol{w})$, where $L_{(u,q)}(\boldsymbol{w})$ denotes the average loss function for all data of the q-th class in the u-th client's local dataset. Let $\boldsymbol{w}_i$ and $\boldsymbol{w}_0$ denote the global model parameter at the i-th round and the initial global model parameter, respectively. To facilitate the analysis, we establish the following standard assumptions:

Assumption 1. (L-smooth) The global loss function $L(\boldsymbol{w}^c, \boldsymbol{w}^s)$ and the averaged local loss functions of clients $L_{u,q}(\boldsymbol{w}_u^{\mathrm{c}}, \boldsymbol{w}_u^{\mathrm{s}})$ are continuously differentiable with respect to $\boldsymbol{w}_u^{\mathrm{c}}$ and $\boldsymbol{w}_u^{\mathrm{s}}$. $\nabla_{\boldsymbol{w}^c} L_u(\boldsymbol{w}_u^{\mathrm{c}}, \boldsymbol{w}_u^{\mathrm{s}})$ is L_c-Lipschitz continuous with $\boldsymbol{w}_u^{\mathrm{c}}$ and L_{cs}-Lipschitz continuous with $\boldsymbol{w}_u^{\mathrm{s}}$, that is, $\|\nabla_{\boldsymbol{w}^c} L_u(\boldsymbol{w}_u^{\mathrm{c}}, \boldsymbol{w}_u^{\mathrm{s}}) - \nabla_{\boldsymbol{w}^c} L_u(\boldsymbol{w}_u^{\mathrm{c}'}, \boldsymbol{w}_u^{\mathrm{s}})\| \leq L_{\mathrm{c}}\|\boldsymbol{w}_u^{\mathrm{c}} - \boldsymbol{w}_u^{\mathrm{c}'}\|$, and $\|\nabla_{\boldsymbol{w}^c} L_u(\boldsymbol{w}_u^{c}, \boldsymbol{w}_u^{\mathrm{s}}) - \nabla_{\boldsymbol{w}^c} L_u(\boldsymbol{w}_u^{c}, \boldsymbol{w}_u^{\mathrm{s}'})\| \leq L_{cs}\|\boldsymbol{w}_u^{\mathrm{s}} - \boldsymbol{w}_u^{\mathrm{s}'}\|$. Similarly, $\nabla_{\boldsymbol{w}^s} L_u(\boldsymbol{w}_u^{\mathrm{c}}, \boldsymbol{w}_u^{\mathrm{s}})$ is L_s-Lipschitz continuous with $\boldsymbol{w}_u^{\mathrm{s}}$ and L_{sc}-Lipschitz continuous with $\boldsymbol{w}_u^{\mathrm{c}}$.

Assumption 2. (Unbiased Gradient and Bounded Variance) The stochastic gradient $\tilde{\nabla} L_u$ at each client serves as an unbiased estimator of the local gradient: $\mathbb{E}[\tilde{\nabla} L_u(\boldsymbol{w}|\mathcal{B}_{u,i})] = \nabla L_u(\boldsymbol{w})$, with the bounded variance $\mathbb{E}[\|\tilde{\nabla} L_u(\boldsymbol{w}|\mathcal{B}_{u,i}) - \nabla L_u(\boldsymbol{w})\|^2] \leq \kappa^2$, and $\kappa^2 \geq 0$.

Assumption 3. (Bounded Dissimilarity) Two non-negative constants exist $\delta \geq 1, \psi^2 \geq 0$, such that $\sum_{q=1}^{Q} \frac{1}{Q} \|\nabla_{\boldsymbol{w}} L_q(\boldsymbol{w})\|^2 \leq \delta \|\sum_{q=1}^{Q} \frac{1}{Q} \nabla_{\boldsymbol{w}} L_q(\boldsymbol{w})\| + \psi^2, \forall \boldsymbol{w}$.

Assumption 4. (Class-wise Similarity) For every u and q, the divergence between the gradient of the global average loss function and the local equivalent is confined within a specified constant in the l^2 norm, specifically, $\|\nabla_{\boldsymbol{w}^c} L_q(\boldsymbol{w}) - \nabla_{\boldsymbol{w}^c} L_{u,q}(\boldsymbol{w})\|^2 \leq \zeta_{u,q}^2$ and $\|\nabla_{\boldsymbol{w}^s} L_q(\boldsymbol{w}) - \nabla_{\boldsymbol{w}^s} L_{u,q}(\boldsymbol{w})\|^2 \leq \varsigma_{u,q}^2$.

Most deep neural networks (NNs) have multiple layers, which satisfy the L-smooth condition. According to [27], a deep NN is defined by a composition of functions, which qualifies as a Lipschitz NN if each function within every layer is Lipschitz continuous. It has been established in [27], [28] that convolutional layers, linear layers, and certain nonlinear activation functions (e.g., Sigmoid and tanh) are Lipschitz functions. Consequently, most deep NNs exhibit Lipschitz continuous gradients. In a Lipschitz NN where all layers are composed of Lipschitz functions, both the client-side and server-side models, comprised of Lipschitz layers, are also Lipschitz functions. Therefore, Assumption 1 posits that the entire NN is Lipschitz continuous is validated. Assumptions 2 and 3 are commonly employed in existing convergence analysis literature, as seen in references [28]–[31]. Assumption 4 relies on the premise that data from the same class are similar. As in the conventional setting, we formulate a pivotal lemma to support our investigation:

Lemma 1. *Let Assumption* 1 *holds, clients' averaged local loss function relationship:*

$$
\begin{aligned}
&L_u\left(\boldsymbol{w}_u^{c'}, \boldsymbol{w}_u^{s'}\right) - L_u\left(\boldsymbol{w}_u^c, \boldsymbol{w}_u^s\right) \leq \frac{1+\chi}{2} L_c \left\|\boldsymbol{w}_u^{c'} - \boldsymbol{w}_u^c\right\|^2 \\
&+ \left\langle \nabla_{\boldsymbol{w}^c} L_u\left(\boldsymbol{w}_u^c, \boldsymbol{w}_u^s\right), \boldsymbol{w}_u^{c'} - \boldsymbol{w}_u^c \right\rangle + \frac{1+\chi}{2} L_s \left\|\boldsymbol{w}_u^{s'} - \boldsymbol{w}_u^s\right\|^2 \\
&+ \left\langle \nabla_{\boldsymbol{w}^s} L_u\left(\boldsymbol{w}_u^c, \boldsymbol{w}_u^s\right), \boldsymbol{w}_u^{s'} - \boldsymbol{w}_u^s \right\rangle, \qquad (11)
\end{aligned}
$$

where $\chi = \max\{L_{cs}, L_{sc}\}/\sqrt{L_c L_s}$*, which quantifies the relative cross-sensitivity of* $\nabla_{\boldsymbol{w}^c} L_u(\boldsymbol{w}_u^c, \boldsymbol{w}_u^s)$ *with respect to* $\boldsymbol{w}_u^s$ *and* $\nabla_{\boldsymbol{w}^s} L_u(\boldsymbol{w}_u^c, \boldsymbol{w}_u^s)$ *with respect to* $\boldsymbol{w}_u^c$*.*

The proof is based on Lemma 1 in [31]. Lemma 1 illustrates the gradient interactions between a neural network's client-side and server-side models.

Theorem 1. *Let Assumption* 1*,* 2*,* 3 *and* 4 *hold,* $\eta_c \leq \frac{1}{2(1+\chi)L_c}$ *and* $\eta_s \leq \frac{1}{2(1+\chi)L_s}$*, after* I *rounds, the disparity between the global loss function and the optimal loss is bounded by:*

$$
\begin{aligned}
&L(\boldsymbol{w}_I) - L(\boldsymbol{w}^*) \leq {H_1}^I \left(L(\boldsymbol{w}_0) - L(\boldsymbol{w}^*)\right) \\
&+ \frac{1 - {H_1}^I}{1 - H_1} \left\{H_2 + \left({\psi_1}^2 \eta_c + {\psi_2}^2 \eta_s\right) \mathbb{E}\left[\mathcal{G}(\mathcal{M})\right]\right\}, \qquad (12)
\end{aligned}
$$

where $H_1 = 1 - L_c\eta_c(1 - \frac{8K_1}{1-K_1}) - L_s\eta_s(1 - \frac{8K_2}{1-K_2}) + (L_c\eta_c\delta_1 + L_s\eta_s\delta_2)2B\mathbb{E}[\mathcal{G}(\mathcal{M})]$*, and* $H_2 = 4\eta_c\zeta^2 + 4\eta_s\xi^2 + (\frac{\eta_c^3 L_c^2 \kappa_1^2}{1-K_1} + \frac{\eta_s^3 L_s^2 \kappa_2^2}{1-K_2})4(J-1) + \frac{1}{B}\frac{2K_1}{1-K_1}\eta_c\zeta^2 + \frac{1}{B}\frac{2K_2}{1-K_2}\eta_s\xi^2 + {\eta_c}^2 L_c \kappa_1^2(1+\chi) + {\eta_s}^2 L_s \kappa_2^2(1+\chi)$.

Proof: Please see Appendix A. ∎

As the expected value $\mathbb{E}[\mathcal{G}(\mathcal{M})]$ decreases, both H_1 and $\frac{1-{H_1}^I}{1-H_1}$ simultaneously decrease, resulting in a tighter convergence bound. Theorem 1 substantiates that reducing class imbalance is advantageous for the SFL system and additionally provides a convergence guarantee for our algorithm.

B. Skewness Measurement

We use a vector of size $Q \geq 2$ to indicate the local label distribution of a client's training samples $\mathcal{D}_u$, denoted by $\boldsymbol{a}_u = [a_{(u,1)}, \cdots, a_{(u,q)}, \cdots, a_{(u,Q)}]$, where each $a_{(u,q)} \geq 0$ and $\sum_{q=1}^{Q} a_{(u,q)} = 1$. We aim to find a subset $\mathcal{M}$ of size M, such that the grouped dataset $\mathcal{D}_{\mathcal{M}} = \sum_{u\in\mathcal{M}} \mathcal{D}_u$ is class-balanced, where the following vector $\boldsymbol{a}_{\mathcal{M}}$ can represent the label distribution of $\mathcal{D}_{\mathcal{M}}$,

$$
\begin{aligned}
\boldsymbol{a}_{\mathcal{M}} = \frac{\sum_{n\in\mathcal{M}} D_u \boldsymbol{a}_u}{\sum_{n\in\mathcal{M}} D_u} = [&\frac{\sum_{n\in\mathcal{M}} D_u a_{(u,1)}}{\sum_{n\in\mathcal{M}} D_u}, \cdots, \\
&\frac{\sum_{n\in\mathcal{M}} D_u a_{(u,q)}}{\sum_{n\in\mathcal{M}} D_u}, \cdots, \frac{\sum_{n\in\mathcal{M}} D_u a_{(u,Q)}}{\sum_{n\in\mathcal{M}} D_u}]. \qquad (13)
\end{aligned}
$$

The term $\mathcal{G}(\mathcal{M})$, related to the client sampling policy, measures the skewness of scheduled client subset $\mathcal{M}$'s datasets:

$$
\mathcal{G}(\mathcal{M}) = \sum_{q=1}^{Q} \left(\frac{\sum_{u\in\mathcal{M}} D_u a_{(u,q)}}{\sum_{u\in\mathcal{M}} D_u} - \frac{1}{Q}\right)^2. \qquad (14)
$$

For clarity, we define the skewness of a grouped dataset as $\mathcal{G}$. Specifically, $\mathcal{G}(\mathcal{M}) = \sum_{q=1}^{Q} (\frac{\sum_{u\in\mathcal{M}} D_u a_{(u,q)}}{\sum_{u\in\mathcal{M}} D_u} - \frac{\sum_{u\in\mathcal{U}} D_u a_{(u,q)}}{\sum_{u\in\mathcal{U}} D_u})^2$ generally quantifies the skewness. Essentially, $\mathcal{G}(\mathcal{M})$ measures the L_2 discrepancy between the label distribution of $\mathcal{D}_{\mathcal{M}}$ and that of an ideal class-balanced dataset with a uniform label distribution. While other probabilistic distances are more commonly used than L_2, $\mathcal{G}(\mathcal{M})$ is selected for its analytical tractability and computational efficiency.

Hence, we aim to further reduce $\mathcal{G}(\mathcal{M})$ and expedite learning convergence by selecting a subset of clients and adjusting pre-client stepsizes $\lambda_u (u \in \mathcal{M}_i)$ in each global round i. This approach aims to create a partial participants' grouped dataset that closely resembles the full dataset. To achieve this, we introduce a mapping function $\mathcal{Z}: \mathcal{U} \to \mathcal{M}_i$, which assigns each client $u \in \mathcal{U}$ to a scheduled client $\mathcal{Z}(u) \in \mathcal{M}_i$, ensuring that the data distribution of client u is approximated by $\mathcal{Z}(u)$. For each client $m \in \mathcal{M}_i$, let $\Omega_m = u : u \in \mathcal{U}, \mathcal{Z}(u) = m$ represent the set of clients whose data distribution is approximated by client m, with $\lambda_u = |\Omega_m|$. For each type of data sample, we have

$$
\begin{aligned}
\frac{1}{U}\sum_{u=1}^{U} a_{(u,b)} &= \frac{1}{U}\sum_{u=1}^{U} \left(a_{(u,b)} - a_{(\partial(u),b)} + a_{(\partial(u),b)}\right) \\
&= \frac{1}{U}\sum_{u=1}^{U} \left(a_{(u,b)} - a_{(\partial(u),b)}\right) + \frac{1}{U}\sum_{u\in\mathcal{M}_i} \lambda_u a_{(\partial(u),b)}. \qquad (15)
\end{aligned}
$$

By rearranging this function and taking norm, we have

$$
\frac{1}{U}\|\sum\nolimits_{u=1}^{U} a_{(u,b)} - \sum\nolimits_{u\in\mathcal{M}_i} \lambda_u a_{(\mathcal{Z}(u),b)}\|
$$

$$= \frac{1}{U}\|\sum_{u=1}^{U}(a_{(u,b)} - a_{(\mathcal{Z}(u),b)})\| \overset{(a)}{\leq} \frac{1}{U}\sum_{u=1}^{U}\|a_{(u,b)} - a_{(\mathcal{Z}(u),b)}\| \tag{16}$$

where (a) follows triangular inequality. A function's upper-bound is reduced when $\mathcal{Z}$ maps $u \in \mathcal{U}$ to a client in $\mathcal{M}_i$ with minimal Euclidean distance between local label distributions. That is, $\mathcal{Z}(u) = \arg\min_{m\in\mathcal{M}_i}\|a_{(u,b)} - a_{(\mathcal{Z}(u),b)}\|$. Consequently, the approximation error in formula (16) is

$$\begin{aligned}&\|\sum\nolimits_{u=1}^{U} a_{(u,b)} - \sum\nolimits_{u\in\mathcal{M}_i}\lambda_u a_{(\mathcal{Z}(u),b)}\| \\ &\leq \sum\nolimits_{u=1}^{U}\min\nolimits_{m\in\mathcal{M}_i}\|a_{(u,b)} - a_{(m,b)}\|.\end{aligned} \tag{17}$$

Hence, the approximation error can be reduced by minimizing the second term. By minimizing the upper bound of the approximation error, the performance of training is boosted. We define our target $G(\mathcal{M}_i) = \sum_{u=1}^{U}\min_{m\in\mathcal{M}_i}\|a_{(u,b)} - a_{(m,b)}\|$ to quantify the approximate error of a client sampling strategy.

C. Problem Formulation

This study aims to boost the effectiveness of the proposed framework through jointly optimizing the client sampling, model splitting, and bandwidth allocation policies under latency and wireless resource constraints. According to our convergence analysis, reducing skewness is shown to directly improve learning performance. Additionally, minimizing overall latency is crucial for accelerating the training process. To expedite learning convergence, we strategically schedule clients that exhibit the lowest latency, which is enabled by favorable channel characteristics and robust computational capabilities, as well as minimal dataset skewness. To this end, we attempt to enhance SFL learning performance by balancing the trade-offs between skewness and latency, as demonstrated in prior research (e.g., [32]). We introduce two weight factors, $\rho_1 \geq 0$ and $\rho_2 \geq 0$, to capture the Pareto-optimal trade-offs between skewness and latency, with their values adjusted to fit particular scenarios. A larger ρ_1 combined with a smaller ρ_2 emphasizes class imbalance, whereas the reverse setup prioritizes minimizing client latency. We optimize SFL performance per round, considering the independence of bandwidth and clients, rather than seeking a multi-round optimization under long-term resource constraints [33]. The problem is formulated as follows:

$$\mathcal{P}: \quad \min_{\boldsymbol{s}_i,\mathcal{M}_i,\boldsymbol{n}_i} \ \rho_1\mathcal{G}(\mathcal{M}_i) + \rho_2\mathcal{T}(\mathcal{M}_i) \tag{18}$$

$$\text{s. t.} \quad \sum\nolimits_{u\in\mathcal{M}_i} n_{u,i} \leq 1, \forall u \in \mathcal{U}, \forall i, \tag{18a}$$

$$0 \leq n_{u,i} \leq 1, \forall u \in \mathcal{U}, \forall i, \tag{18b}$$

$$s_{u,i} \in \mathcal{S}, \forall u \in \mathcal{U}, \forall i, \tag{18c}$$

$$\alpha_{u,i} \in \{0,1\}, \forall u \in \mathcal{U}, \forall i, \tag{18d}$$

where (18a) stipulates that the total bandwidth allocated to the clients is not allowed to surpass the accessible bandwidth. (18b) enforces limits on the bandwidth assigned to each individual client. (18c) mandates model split point. (18d) determines the participants for each round. Notably, we employ 'virtual clients' to manage strict latency constraints effectively. Problem $\mathcal{P}$ presents significant challenges due to its combinatorial nature, involving optimization over a multi-dimensional space that includes both discrete and continuous variables. In the subsequent section, we explore two specific instances of $\mathcal{P}$: skewness-aware and latency-aware client sampling. These are submodular maximization problems and are NP-hard, thus establishing the NP-hardness of problem $\mathcal{P}$.

IV. Optimal Client Sampling, Model Splitting, and Bandwidth allocation

In this section, we develop efficient algorithms for solving $\mathcal{P}$ in polynomial time. We first design an optimal bandwidth allocation algorithm and an optimal model splitting algorithm. Then based on two exceptional examples, we demonstrate that $\mathcal{P}$ is a non-monotonic submodular optimization issue. Additionally, we propose a L&S-aware client sampling algorithm to solve $\mathcal{P}$ and yield optimal client sampling, model splitting, and bandwidth allocation policies.

A. Optimal Bandwidth Allocation

To determine the optimal bandwidth allocation policy for any client sampling subset $\mathcal{M}_i$ and model splitting strategy $\boldsymbol{s}_i$, we decompose the subproblem of optimal bandwidth allocation from $\mathcal{P}$ as follows:

$$\begin{aligned}\mathcal{P}_1: \quad &\min_{\boldsymbol{n}_i} \ \max_{u\in\mathcal{M}_i}\{T_{u,i}^{\mathrm{CM}} + T_{u,i}^{\mathrm{CP}}\} \\ &\text{s. t.} \ \ (18\text{a}), (18\text{b}).\end{aligned} \tag{19}$$

Both the target function and the constraints meet the criteria for convexity. Hence, $\mathcal{P}_1$ is a standard convex optimization problem, and we utilize Lemma 2 to find its optimal solution:

Lemma 2. *The optimal bandwidth allocation strategy adheres to:*

$$n_{u,i} = \frac{\nu_{u,i}}{(\mathcal{T}_i^*(\mathcal{M}_i) - T_{u,i}^{\mathrm{CP}})\beta\log_2(1 + \frac{p_u h_{u,i}}{\sigma^2})}, \tag{20}$$

where $\mathcal{T}_i^*(\mathcal{M}_i)$ represents the ideal latency for the sampled client subset $\mathcal{M}_i$ in round i, and its value is defined by $\sum_{u\in\mathcal{M}_i} n_{u,i} = 1$. In addition, $\nu_{u,i}$=$JB\vartheta_{u,i}^{\mathrm{s}}(s_u) + \vartheta_{u,i}^{\mathrm{c}}(s_u)$ is the total datasize in one round communication.

Proof: Please see Appendix B. ∎

According to Lemma 2, the bandwidth allocation approach includes an unknown variable $\mathcal{T}_i^*(\mathcal{M}_i)$. The bisection method is utilized to seek the optimal bandwidth allocation strategy, as $n_{u,i}$ decreases monotonically with respect to $\mathcal{T}_i(\mathcal{M}_i)$. To achieve this, we establish the lower and upper bounds of $\mathcal{T}_i(\mathcal{M}_i)$. For the lower bound, the minimum bandwidth allocated to clients in $\mathcal{M}_i$ is less than $\frac{1}{|\mathcal{M}_i|}$, or $\min_{u\in\mathcal{M}_i} n_{u,i} \leq \frac{1}{|\mathcal{M}_i|}$. Thus,

$$\frac{\min\limits_{u\in\mathcal{M}_i}\frac{\nu_{u,i}}{\beta\log_2(1+\frac{p_u h_{u,i}}{\sigma^2})}}{\max(\mathcal{T}_i(\mathcal{M}_i) - T_{u,i}^{\mathrm{CP}})} \leq \frac{1}{|\mathcal{M}_i|}, \forall u \in \mathcal{M}_i, \tag{21}$$

we have the lower bound of $\mathcal{T}_i(\mathcal{M}_i)$:

$$\mathcal{T}_{i,lb}(\mathcal{M}_i) = \min_{u\in\mathcal{M}_i}\frac{|\mathcal{M}_i|\nu_{u,i}}{\beta\log_2(1 + \frac{p_u h_{u,i}}{\sigma^2})} + \min_{u\in\mathcal{M}_i} T_{u,i}^{\mathrm{CP}}. \tag{22}$$

Algorithm 1 Optimal Bandwidth Allocation Algorithm

Initialization: Scheduled client subset $\mathcal{M}_i$, model splitting decision value $\boldsymbol{s}_i$, bandwidth allocation policy $\boldsymbol{n}_i$, the lower bound ($\mathcal{T}_{i,lb}$) and the upper bound ($\mathcal{T}_{i,ub}$) of latency, precision requirements $\varepsilon > 0$.
repeat
  Set $\mathcal{T} = (\mathcal{T}_{i,lb} + \mathcal{T}_{i,ub})/2$.
  For each client $u \in \mathcal{M}_i$, compute the required bandwidth allocation ratio $n_{u,i}(\mathcal{T})$ derived from (20).
  Compute total required bandwidth allocation proportion $\sum_{u\in\mathcal{M}_i} n_{u,i}(\mathcal{T})$.
  if $\sum_{u\in\mathcal{M}_i} n_{u,i}(\mathcal{T}) > 1$ **then**
    Halve the searching region by setting $\mathcal{T}_{i,lb} = \mathcal{T}$ and $\mathcal{T}_{i,ub} = \mathcal{T}_{i,ub}$.
  else if $0 < \sum_{u\in\mathcal{M}_i} n_{u,i}(\mathcal{T}) < 1 - \varepsilon$ **then**
    Halve the searching region by setting $\mathcal{T}_{i,lb} = \mathcal{T}_{i,lb}$ and $\mathcal{T}_{i,ub} = \mathcal{T}$.
  else
    Break the loop.
until $|\mathcal{T}_{i,ub} - \mathcal{T}_{i,lb}| < \varepsilon$
return The optimal latency $\mathcal{T}_i^* = \mathcal{T}$, the optimal bandwidth allocation policy $\boldsymbol{n}_i^*$.

For the upper bound, the maximum bandwidth allocated to clients in $\mathcal{M}_i$ exceeds $\frac{1}{|\mathcal{M}_i|}$, with $\max_{u\in\mathcal{M}_i} n_{u,i} \geq \frac{1}{|\mathcal{M}_i|}$. The upper bound is derived similarly to the lower bound:

$$\mathcal{T}_{i,ub}(\mathcal{M}_i) = \max_{u\in\mathcal{M}_i} \frac{|\mathcal{M}_i|\nu_{u,i}}{\beta \log_2(1 + \frac{p_u h_{u,i}}{\sigma^2})} + \max_{u\in\mathcal{M}_i} T_{u,i}^{\mathrm{CP}}. \quad (23)$$

Leveraging the lower and upper bounds, a bisection algorithm is employed for identifying the ideal latency $\mathcal{T}_i^*(\mathcal{M}_i)$. To enhance comprehension, the procedure of allocating bandwidth optimally is introduced in Algorithm 1. This algorithm progressively reduces the search interval with each iteration, ceasing when the required precision (i.e., ε) is achieved. For time complexity, each iteration of this algorithm involves a half-interval search, which logarithmically reduces the search space $\mathcal{O}(\log_2 \frac{\mathcal{T}_{i,ub}(\mathcal{M}_i) - \mathcal{T}_{i,lb}(\mathcal{M}_i)}{\varepsilon})$. For each interval halving, it computes the bandwidth allocation for M clients in subset $\mathcal{M}_i$, thus the total complexity is $\mathcal{O}(M \log_2 \frac{\mathcal{T}_{i,ub}(\mathcal{M}_i) - \mathcal{T}_{i,lb}(\mathcal{M}_i)}{\varepsilon})$. From the preceding analysis, we derive the following remark.

Remark 1. According to (20), the fraction of wireless bandwidth, i.e., $n_{u,i}$, allotted to client $u (u \in \mathcal{U})$ decreases monotonically with CPU frequency f_u and channel gain $h_{u,i}$. Thus, clients with inadequate computing capabilities and weak channel conditions require additional bandwidth.

B. Optimal Model Splitting

In this subsection, we determine the optimal model splitting solution for a specified client sampling policy $\mathcal{M}_i$ and bandwidth allocation policy $\boldsymbol{n}_i$. The optimal model splitting subproblem is

$$\mathcal{P}_2: \quad \min_{\boldsymbol{s}_i} \ \{T_{u,i}^{\mathrm{CM}} + T_{u,i}^{\mathrm{CP}}\} \quad \text{s. t.} \ \ (18\mathrm{c}). \quad (24)$$

The target is constructed to be a summation of affine functions of the decision variables $s_{u,i}$. Since an affine function maintains the property of convexity, we have the target function is convex in terms of the splitting point variables $s_{u,i}$. The linear constraint $(18\mathrm{c})$ renders $\mathcal{P}_2$ a convex optimization problem.

Algorithm 2 Optimal Model Splitting Algorithm

Initialization: Scheduled client subset $\mathcal{M}_i$, optimal bandwidth allocation policy $\boldsymbol{n}_i^*$ from Algorithm 1, model splitting decision value $\boldsymbol{s}_i$.
for $u \in \mathcal{M}_i$ **do**
  for each split point $s_{u,i} \in \mathcal{S}$ **do**
    Calculate the expected learning latency $\widetilde{T}(s_{u,i}) = \{T_{u,i}^{\mathrm{CM}} + T_{u,i}^{\mathrm{CP}}\}$.
  end for
  $s_{u,i}^* = \arg\min_{s_{u,i}\in\mathcal{S}} \widetilde{T}(s_{u,i})$
end for
Set $\boldsymbol{s}_i^* = \boldsymbol{s}_i^{(\tau)}$ as the optimal model splitting strategy.
return The optimal model splitting policy $\boldsymbol{s}_i^*$.

We utilize the SAA method [34] in the Algorithm 2 for efficiently solving optimal model splitting problem.

As aforementioned, we decompose the optimal bandwidth allocation problem $\mathcal{P}_1$ and optimal model splitting problem $\mathcal{P}_2$ from $\mathcal{P}$, which are efficiently solved with proposed algorithms. Accordingly, Algorithm 2 utilize the result of optimal bandwidth allocation policy from Algorithm 1. Therefore, the time complexity of this algorithm is $\mathcal{O}(MS \log_2 \frac{\mathcal{T}_{i,ub}(\mathcal{M}_i) - \mathcal{T}_{i,lb}(\mathcal{M}_i)}{\varepsilon})$. To adapt the time-varying communication and training conditions, both of the two algorithms are designed to be executed at each round, which enables the optimal resource allocation strategy that best aligns with the current environment.

C. Latency and Skewness-Aware Client Sampling Strategy

In this subsection, we explore two specific instances of problem $\mathcal{P}$, i.e., the latency and skewness-aware client sampling issues. Based on them, we propose a L&S-aware client sampling algorithm.

To formulate the latency-aware client sampling problem, set $\rho_1 = 0$ and $\rho_2 = 1$ in problem $\mathcal{P}$. The ideal bandwidth allocation and model splitting strategy for any scheduled client subset $\mathcal{M}_i$ are determined by executing Algorithm 1 and 2. The related optimal latency is $\mathcal{T}_i^*(\mathcal{M}_i)$. Substituting $\mathcal{T}_i^*(\mathcal{M}_i)$ into problem $\mathcal{P}$ yields the corresponding problem:

$$\mathcal{P}_3: \quad \min_{\mathcal{M}_i} \ \mathcal{T}^*(\mathcal{M}_i) \quad (25)$$
$$\text{s. t.} \ \ |\mathcal{M}_i| = M, (18\mathrm{b}). \quad (25\mathrm{a})$$

In $\mathcal{P}_3$, we add a constraint (25a), while the empty client set can be utilized as the solution for the problem without it.

To address problem $\mathcal{P}_3$, an intuitive approach involves computing the optimal latency for all potential client sampling strategies and selecting the client with the lowest latency. Nevertheless, with a total of C_U^M potential strategies, and considering that the overall amount of clients is typically large while the clients participate in each round are fewer, the number of potential client subsets becomes extensive. Consequently, evaluating the latency for all potential strategies is impractical due to the prohibitive computational complexity.

To formulate the skewness-aware client sampling problem, set $\rho_1 = 1$ and $\rho_2 = 0$ in problem $\mathcal{P}$:

$$\mathcal{P}_4: \quad \min_{\mathcal{M}_i} \ \mathcal{G}(\mathcal{M}_i) \quad (26)$$
$$\text{s. t.} \ \ |\mathcal{M}_i| = M. \quad (26\mathrm{a})$$

Algorithm 3 L&S-Aware Client Sampling Algorithm

Initialize $\mathcal{M}_1 \leftarrow \emptyset$, $\mathcal{M}_2 \leftarrow \mathcal{U}$
for $u \in \mathcal{U}$ **do**
  Let $x_u \leftarrow (\max \mathcal{F}(\mathcal{M}_1) - \mathcal{F}(\mathcal{M}_1 \cup \{u\}, 0))$
  Let $y_u \leftarrow (\max \mathcal{F}(\mathcal{M}_2) - \mathcal{F}(\mathcal{M}_2 \setminus \{u\}, 0))$
  If $x_u = y_u = 0$, let $\frac{x_u}{x_u+y_u} = 1$
  With probability $\frac{x_u}{x_u+y_u}$ do $\mathcal{M}_1 \leftarrow \mathcal{M}_1 \cup \{u\}$ and $\mathcal{M}_2 \leftarrow \mathcal{M}_2$
  Otherwise $\mathcal{M}_1 \leftarrow \mathcal{M}_1$ and $\mathcal{M}_2 \leftarrow \mathcal{M}_2 \setminus \{u\}$
end for
Let $\mathcal{M}_i = \mathcal{M}_1$ (or $\mathcal{M}_i = \mathcal{M}_2$).
return The sampled client subset $\mathcal{M}_i$.

As in problem $\mathcal{P}_3$, a client set constraint is employed in problem $\mathcal{P}_4$ due to its monotone objective function with regard to client set size. Without limitations (26a), problem $\mathcal{P}_3$ can be solved by involving each client (i.e., $\mathcal{M}_i = \mathcal{U}$).

According to [15], by proving that problem $\mathcal{P}_3$ and $\mathcal{P}_4$ are submodular set cover problems, we adopt a near-optimal solution for both problem $\mathcal{P}_3$ and $\mathcal{P}_4$ by utilizing greedy algorithm. To this end, deriving from the submodular function's definition in [35], the Lemma 3 is introduced for ideal latency $\mathcal{T}_i^*(\mathcal{M}_i)$ and skewness function $\mathcal{G}(\mathcal{M}_i)$ as follows:

Lemma 3. *For each set $\mathcal{M}_1 \subseteq \mathcal{M}_2$, the optimal latency function $\mathcal{T}_i^*(\mathcal{M}_i)$ is monotonically increasing, while the objective function $\mathcal{G}(\mathcal{M}_i)$ is monotonically decreasing. Specifically, for any set $\mathcal{M}_1 \subseteq \mathcal{M}_2 \subseteq \mathcal{U}$ and $m \in \mathcal{U} \setminus \mathcal{M}_2$, we have $\mathcal{T}_i^*(\{m\} \cup \mathcal{M}_1) - T_i^*(\mathcal{M}_1) \leq \mathcal{T}_i^*(\{m\} \cup \mathcal{M}_2) - T_i^*(\mathcal{M}_2)$, and $\mathcal{G}(\{m\} \cup \mathcal{M}_1) - G(\mathcal{M}_1) \leq \mathcal{G}(\{m\} \cup \mathcal{M}_2) - G(\mathcal{M}_2)$.*

Proof: Please see Appendix C. ∎

Lemma 3 states that $\mathcal{P}_3$ and $\mathcal{P}_4$ are NP-Hard cardinality constraint submodular maximization problems. We use greedy algorithm [36] to discover a near-optimal solution for them. For $\mathcal{P}_3$, the algorithm starts from $\mathcal{M}_i = \emptyset$ and adds one client $m \in \mathcal{U} \setminus \mathcal{M}_i$ with the biggest marginal gain in each step, i.e. $u = \arg\min_{u \in \mathcal{U} \setminus \mathcal{M}_i} (\mathcal{T}_i^*(\{u\} \cup \mathcal{M}_i) - T_i^*(\mathcal{M}_i))$. In addition, for $\mathcal{P}_4$, the greedy algorithm begins with $\mathcal{M}_i = \emptyset$ and selets one client u with highest marginal gain, i.e., $u = \arg\min_{u \in \mathcal{U} \setminus \mathcal{M}_i} (\mathcal{G}(\{u\} \cup \mathcal{M}_i) - G(M_i))$ in each iteration.

However, clients have time-varying computation capability, channel conditions, and data skewness. To accelerate learning convergence, the client sampling mechanism should consider latency and skewness simultaneously. Leveraging latency and dataset skewness features, we design an efficient solution to balance clients' latency and skewness in problem $\mathcal{P}$.

Based on Lemma 2, the optimal latency for any client sampling subset $\mathcal{M}_i \subseteq \mathcal{U}$ can be determined through Algorithm 1, denoted $\mathcal{T}_i^*(\mathcal{M}_i)$. Substituting $\mathcal{T}_i^*(\mathcal{M}_i)$ into $\mathcal{P}$ yields the corresponding problem:

$$\widetilde{\mathcal{P}}: \quad \min_{\mathcal{M}_i} \ \mathcal{F}(\mathcal{M}_i) = \rho_1 \mathcal{G}(\mathcal{M}_i) + \rho_2 \mathcal{T}_i^*(\mathcal{M}_i)$$
$$\text{s. t.} \quad (18\text{d}). \tag{27}$$

Remark 2. According to Lemma 3, $-\mathcal{T}_i^*(\mathcal{M}_i)$ and $-\mathcal{G}(\mathcal{M}_i)$ are both monotonic submodular functions. Hence, the negation of the objective function of problem $\widetilde{\mathcal{P}}$, specifically, $-\rho_1 \mathcal{G}(\mathcal{M}_i) - \rho_2 \mathcal{T}_i^*(\mathcal{M}_i)$, constitutes a non-monotone submodular function. Consequently, problem $\widetilde{\mathcal{P}}$ is classified as an unconstrained non-monotone submodular maximization problem, generally acknowledged as NP-Hard.

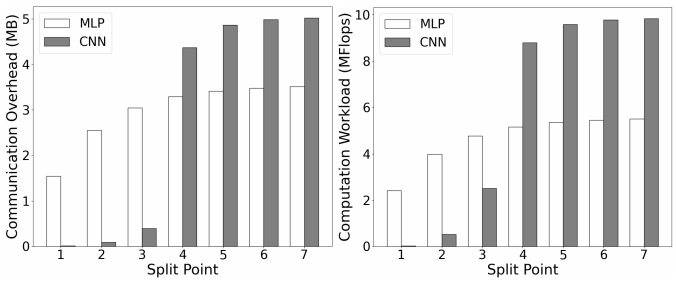
In accordance with Remark 2, for seeking a suboptimal solution for $\widetilde{\mathcal{P}}$, we employ the double greedy algorithm [37]. For clarity, we recapitulate the detailed steps in Algorithm 3, which requires solving $2U$ times model splitting and bandwidth allocation problem for finding the scheduled client subset. Therefore, each decision involves invoking these algorithms $2U$ times, and the complexity per client decision is $\mathcal{O}(2US \log_2 \frac{\mathcal{T}_{i,ub}(\mathcal{M}_i) - \mathcal{T}_{i,lb}(\mathcal{M}_i)}{\varepsilon})$. It is worth noting that the complexity primarily grows linearly with U, while the brute-force approach results in an overall complexity of $\mathcal{O}(2^U S \log_2 \frac{\mathcal{T}_{i,ub}(\mathcal{M}_i) - \mathcal{T}_{i,lb}(\mathcal{M}_i)}{\varepsilon})$. As U increases, the difference in computational load between brute-force and our algorithm becomes starkly apparent, because the brute-force method grows exponentially, while the linear growth of our algorithms are acceptable. Hence, our proposed algorithms significantly reduce computational demands as networks scale. The server initializes two client subsets, $\mathcal{M}_1 = \emptyset$ and $\mathcal{M}_2 = \mathcal{U}$, and then sequentially passes through the clients in $\mathcal{U}$. The algorithm determines whether to add client $u (u \in \mathcal{U})$ to $\mathcal{M}_1$ or remove it from $\mathcal{M}_2$. To implement the suggested algorithm, the server needs to gather clients' channel information to calculate appropriate model splitting strategies, bandwidth allocation policies, and latency. In addition, [37] indicates that the double greedy algorithm attain a close $1/2$ approximation to the optimum solution.

V. Numerical Results

In this section, we examine the effects of various cut layers in the second subsection. Furthermore, we assess the superiority of the proposed skewness-aware and L&S-aware algorithms over baseline approaches for image classification tasks in the third and forth subsections, respectively.

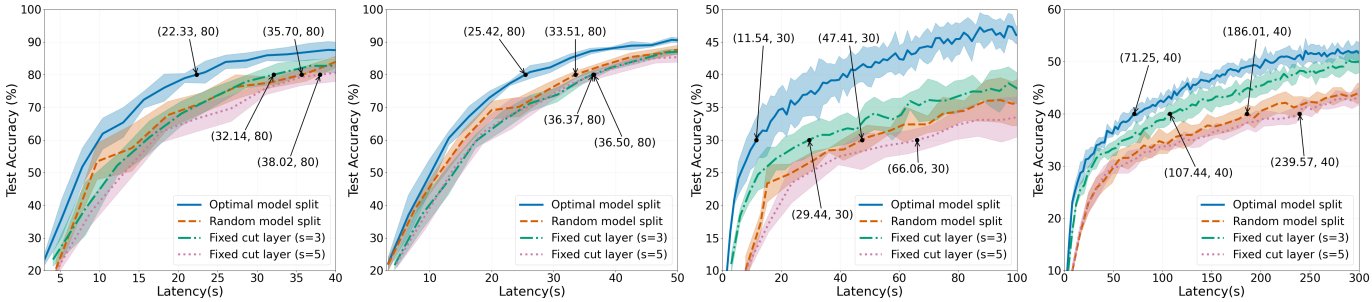
TABLE II
System Parameters

Parameter	Value	Parameter	Value
U	100	M	10
β	1MHz	B	64
σ^2	-174dBm	h_0	-30dBm
f_s	20	η	0.01
J	5	I	200



(a) Communication overhead (b) Computing workload

Fig. 3. The client's communication overhead and computing workload in SFL with varying split point for MLP and CNN.



(a) 5 clients on MNIST (b) 10 clients on MNIST (c) 5 clients on CIFAR-10 (d) 10 clients on CIFAR-10

Fig. 4. Comparison of learning performance for different model splitting strategies on MNIST and CIFAR-10.

A. Simulation Settings

Wireless setting: The simulation assumes an server at the center of a 500m-radius circular area serving $U = 100$ randomly distributed clients. To simulate client heterogeneity, the transmit power for each client is randomly picked from $\{0.01, 0.02, 0.03, 0.05\}$W, while the CPU frequency is randomly choosen from $\{0.5, 0.8, 1.0, 1.2, 1.6\}$GHz. Channel gain is modeled as $h_{u,i} = h_0 \rho_{u,i}(t) d_u^{-2}$, where h_0 is the constant of path loss, d_u is client-server distance, and $\rho_{u,i}(t) \sim \text{Exp}(1)$ is Rayleigh fading gain [31]. Table II shows the default system settings if not specified.

Datasets and Models: We assess the proposed mechanism on a variety of datasets, including MNIST handwritten digits, Fashion-MNIST clothing classification, CIFAR-10 picture classification tasks, and STL-10 image recognition tasks. Each client is randomly assigned a local dataset to simulate realistic data distribution scenarios. For MNIST, we utilize a multilayer perceptron (MLP) model featuring a 784-unit input layer, seven hidden layers with 512, 512, 256, 128, 128, and 64 units, and a 10-unit softmax output layer. Activation across the input and four hidden layers is managed by the ReLU function. This MLP model comprises 900,010 parameters, corresponding to the number of FLOPs required for gradient computation of a single data sample. The Fashion-MNIST dataset is also trained using the same MLP neural network as used for MNIST. For CIFAR-10, we configure a CNN with three 3×3 convolution layers, a 2×2 max-pooling layer, five fully connected layers (2048, 512, 256, 128, and 64 units), and a 10-unit softmax output layer, with activation by the ReLU function in each convolution and fully connected layer. The CNN holds 1314272 parameters, with 5644928 FLOPs necessary for processing one data sample. For the STL-10 dataset, which contains higher-resolution images, we employ a more complex AlexNet neural network. AlexNet for STL-10 is configured with five convolutional layers with 64, 192, 384, 256, and 256 filters and kernel sizes of 11×11, 5×5, and 3×3 respectively, followed by three max-pooling layers with a 3×3 kernel and stride of 2, an adaptive average pooling layer resizing to 6×6, and a classifier section comprising a dropout-augmented sequence of fully connected layers with 4096, 1024, 512, 256 units, culminating in a 10-unit output layer. Activation is achieved through the ReLU function throughout the network. As split point moves deeper, the part of the model residing on the client-side becomes larger. Hence, Fig. 3 illustrates the increasing trend of communication and computing workload for MLP and CNN models.

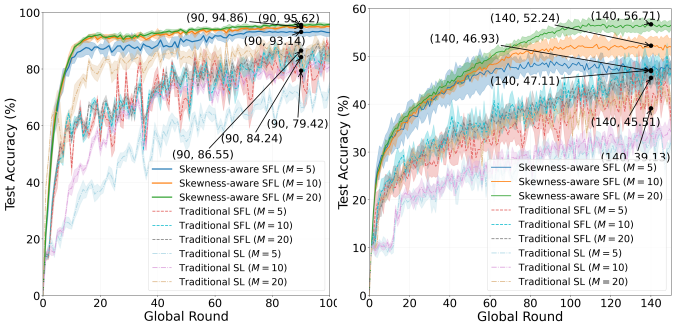
Datasets Allocation Schemes: Two primary types of data heterogeneity scenarios are considered to evaluate the performance of the proposed framework in varied and realistic environments: For the first scenario, we model the datasets exhibit a significant degree of heterogeneity. Half of the biased clients possess datasets that contain $Q/2$ categories, selected randomly from Q categories, while the remaining half of the clients are unbiased and have datasets that encompass all Q categories. The second data heterogeneity scenario utilizes a Dirichlet distribution to allocate data across clients.

B. Impact of Different Cut Layer

To investigate the effect of varying cut layers on learning performance, we assess accuracy across four distinct model splitting schemes. Fig. 4 juxtaposes the performance of the dynamic model splitting algorithm against three baseline strategies: (1) Random model split: the model split point is chosen randomly for each global round. (2) Fixed cut layers s = 3. (3) Fixed cut layers s = 5.

Fig. 4(a) examines the accuracy when 5 clients are trained on the MNIST dataset. Setting a target accuracy of 80%, the dynamic model splitting algorithm achieves this goal in 22.3 seconds, compared to 35.7 seconds, 32.14 seconds, and 38.02 seconds required by the random splitting, fixed cut layer s = 3, and s = 5 strategies, respectively. Consequently, the proposed algorithm yields time savings of 37.5%, 30.5% and 41.3% over the random splitting, fixed cut layer s = 3, and s = 5, respectively. Additionally, Fig. 4(b) evaluates performance with 10 clients targeting an accuracy of 80%, the proposed algorithm reduces training time by 24.1%, 30.1%, and 30.4% compared to the random splitting, fixed cut layer s = 3, and s = 5, respectively.

Fig. 4(c) assesses the accuracy with 5 clients on the CIFAR-10 dataset. When targeting an accuracy of 30%, our algorithm achieves a reduction in training time of 75.7%, 60.8%, and 82.5% compared to random splitting, fixed cut layer s = 3, and s = 5, respectively. Furthermore, Fig. 4(d) illustrates that with 10 clients and a target accuracy of 40%, our proposed algorithm reduces training time by 61.7%, 33.7%, and 70.3% relative to the three other approaches. The experimental findings convincingly display that our algorithm significantly surpasses



(a) On MNIST (b) On CIFAR-10

Fig. 5. Learning performance of three client sampling schemes on MNIST and CIFAR-10.

TABLE III
ACCURACY IMPROVEMENT

Dataset	M	Proposed	Best Baseline	Improvement
MNIST	5	93.14%	79.42%	13.72%
MNIST	10	94.86%	84.24%	10.62%
MNIST	20	95.62%	86.55%	9.07%
CIFAR-10	5	46.93%	39.13%	7.8%
CIFAR-10	10	52.24%	45.51%	6.73%
CIFAR-10	20	56.71%	47.11%	9.6%

the established baselines in both accuracy and convergence speed.

C. Performance of Skewness-Aware Client Sampling

For measuring the efficacy of the skewness-aware client sampling algorithm introduced in Section IV-C, Fig. 5 contrasts the learning outcomes of the proposed algorithm with those of traditional SFL and SL, both utilizing random sampling methods, across client subsets sizes of $M = 5$, $M = 10$, and $M = 20$. Note that M represents the number of clients in subset $\mathcal{M}$, which directly influences skewness.

In Table III, we present the test accuracy enhancements achieved by our skewness-aware algorithm compared to the best baseline, for client subset sizes of $M = 5$, $M = 10$, and $M = 20$. The results indicate that the proposed algorithm secures significant improvements over the best baselines, with the highest accuracy observed when $M = 20$.

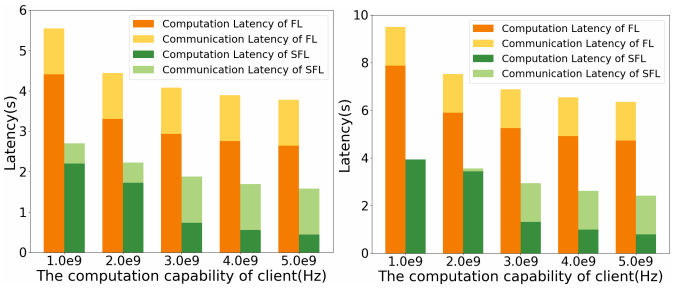
In Fig. 5(a), by the 90th round, the proposed algorithm achieves a test accuracy of 93.14%, 94.86%, 95.62%, when $M = 5$, $M = 10$, $M = 20$, respectively. The best baseline accuracies are 79.42%, 84.24%, 86.55%, when $M = 5$, $M = 10$, $M = 20$, respectively. Thus, the proposed algorithm exceeds the best baseline of traditional SFL and SL by margins of 13.72%, 10.62%, and 9.07%, respectively. Additionally, Fig. 5(b) compares the learning performance of our algorithm against conventional methods on the CIFAR-10 dataset, assessing accuracy by the 140th round. With five clients ($M = 5$), the Skewness-aware SFL achieves an accuracy of 46.93% by the 140th round, which surpasses the highest baseline accuracy of 39.13% by 7.8%. Furthermore, when operating with ten and twenty clients ($M = 10$ and $M = 20$, respectively), the proposed algorithm records accuracies of 52.24% and 56.71%. These results represent improvements of 6.73% and 9.6% over the best baseline accuracies of 45.51% and 47.11%, respectively.

D. Performance Evaluation of L&S-Aware Client Sampling

In this subsection, we evaluate the effectiveness of the proposed L&S-aware client sampling algorithm against five distinct baselines: (1) Skewness-aware: our proposed skewness-aware client sampling algorithm within the SFL framework. (2) Latency-aware: our proposed latency-aware client sampling algorithm in SFL. (3) Optimal B&MS: a scheme combining optimal bandwidth allocation, model splitting, and random client sampling within SFL. (4) Equal B&MS: equal bandwidth allocation with random model splitting and client sampling in SFL. and (5) Traditional FL: traditional federated learning approach using equal bandwidth allocation and random client sampling.

Fig. 6 and Fig. 7 evaluates the learning performance of our proposed client sampling algorithms against conventional schemes. We depict the best and worst performance for $M = 5$ and $M = 10$ by sequentially selecting ρ_1 and ρ_2 from 0.1 to 1.0.

Fig. 8 depicts the latency benefits of our proposed SFL framework compared to traditional FL across various client computational capabilities. This comparison is conducted for both MLP and CNN models. Notably, the SFL framework consistently outperforms traditional FL, demonstrating substantial time efficiencies under all tested conditions of computational capacity for the MLP and CNN models. Furthermore, Table IV details the time required to reach a designated accuracy level, comparing the proposed algorithm against the most effective baseline.

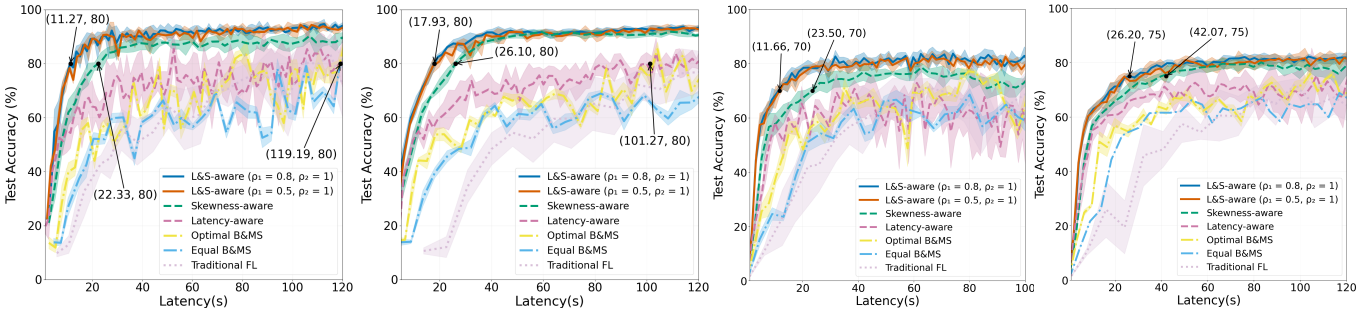


(a) Latency for MLP (b) Latency for CNN

Fig. 8. Latency comparison between proposed SFL framework and conventional FL across varying client computing capabilities.

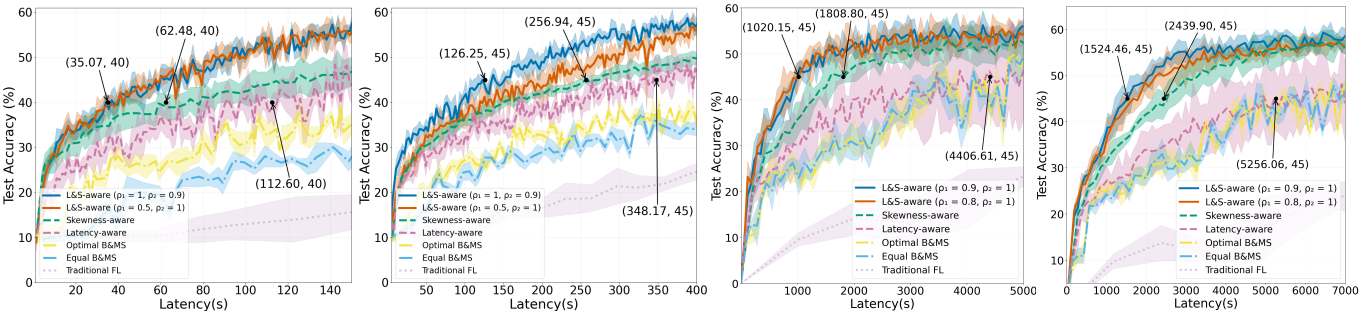
TABLE IV
REQUIRED TIME TO REACH A TARGET ACCURACY

Dataset	M	Target Accuracy	Proposed	Best Baseline	Saved Time
MNIST	5	80%	11.27	22.33	49.5%
MNIST	10	80%	17.93	26.1	31.3%
FMNIST	5	70%	11.66	23.5	50.4%
FMNIST	10	75%	26.2	42.07	37.7%
CIFAR-10	5	40%	35.07	62.48	43.8%
CIFAR-10	10	45%	126.25	256.94	50.9%
STL-10	5	45%	1020.15	1808.8	43.6%
STL-10	10	45%	1524.46	2439.9	37.5%



(a) 5 clients on MNIST (b) 10 clients on MNIST (c) 5 clients on Fashion-MNIST (d) 10 clients on Fashion-MNIST

Fig. 6. Comparison of the learning performance for different clients sampling schemes on the MNIST dataset and Fashion-MNIST dataset.



(a) 5 clients on CIFAR-10 (b) 10 clients on CIFAR-10 (c) 5 clients on STL-10 (d) 10 clients on STL-10

Fig. 7. Comparison of the learning performance for different clients sampling schemes on the CIFAR-10 dataset and STL-10 dataset.

Specifically, Fig. 6(a) evaluates accuracy on the MNIST dataset with 5 clients, setting $\rho_1 = 0.8$ and $\rho_2 = 1$. For reaching a target accuracy of 80%, the L&S-aware algorithm requires only 11.27 seconds, whereas the skewness-aware approach takes 22.33 seconds. Compared to the skewness-aware algorithm, the L&S-aware algorithm achieves a 49.5% reduction in training time, and a significant 90.5% reduction compared to the latency-aware algorithm. Additionally, Fig. 6(b) depicts a scenario with 10 clients where the L&S-aware algorithm takes 17.93 seconds to reach the 80% accuracy target, whereas the skewness-aware and latency-aware algorithms require 26.1 seconds and 101.27 seconds, respectively. Thus, the L&S-aware algorithm reduces training time by 31.3% and 82.3% compared to the skewness-aware and latency-aware algorithms, respectively.

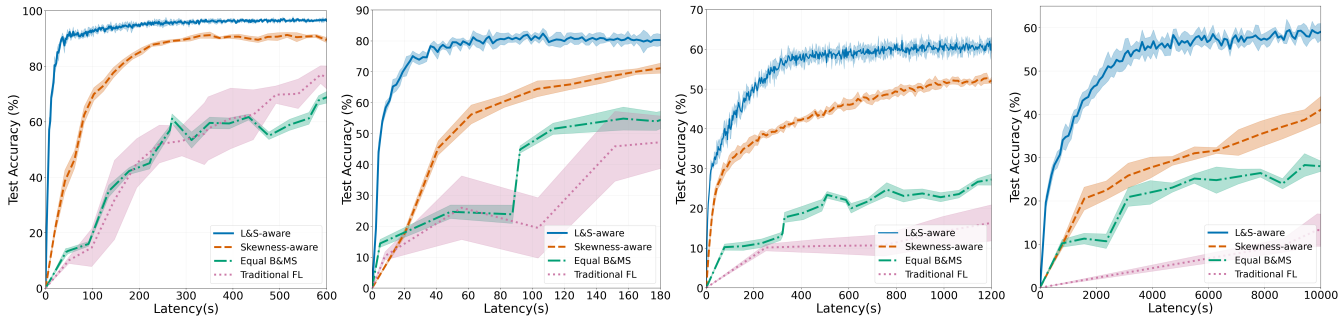
For Fashion-MNIST, Fig. 6(c) demonstrates that it only takes 11.66 seconds to reach a 70% target when 5 clients participate in training. Compared to the best baseline that needs 23.5 seconds to achieve this target, the proposed approach saves 50.4% time. In Fig. 6(d), with 10 clients participating and aiming for a 75% accuracy target, the proposed algorithm costs 26.2 seconds to achieve the target, while the best baseline takes 42.07 seconds, thus 37.7% training time is saved by implement proposed algorithm.

For CIFAR-10, Fig. 7(a) illustrates a learning scenario with five clients scheduled, where the L&S-aware algorithm requires 35.07 seconds to achieve a 40% accuracy target. In comparison, the skewness-aware and latency-aware algorithms require 62.48 seconds and 112.6 seconds, respectively. The proposed mechanism achieves a 43.8% reduction in learning time relative to the skewness-aware algorithm and a 68.9% reduction compared to the latency-aware algorithm. In Fig. 7(b), with ten clients participating and aiming for a 45% accuracy target, the L&S-aware algorithm reduces learning time by 50.9% compared to the skewness-aware and 63.7% compared to the latency-aware algorithms. It is noteworthy that the proposed algorithm not only converges significantly faster but also attains markedly higher accuracy in less time than three other baseline approaches: optimal bandwidth and model splitting (B&MS), equal bandwidth allocation, and FL.

For STL-10, Fig. 7(c) runs under the setting of 5 participants. When set the target accuracy as 45%, the proposed algorithm spends 1020.15 seconds to reach this target, while the best baseline needs 1808.8 seconds. As a result, our algorithm is able to save at least 43.6% time compare to other approaches. In Fig. 7(d), with 10 participants, time consumption is 1524.46 seconds for our algorithm to reach 45% accuracy, while the best baseline needs 2439.9 seconds. Hence, our algorithm saves minimum 37.5% time compared to other baselines.

E. Performance Evaluation under Extreme Conditions

In this subsection, we evaluate the effectiveness of our proposed SFL framework when some clients possess extremely low computational resources or poor network conditions. To this end, we have extended our experimental setup to include clients with notably low communication and computational capacities, where the set of random values for the clients' transmit power in our experimental setup to include $\{0.005, 0.01, 0.02, 0.03, 0.05\}$W. Simultaneously, the



(a) MNIST (b) Fashion-MNIST (c) CIFAR-10 (d) STL-10

Fig. 9. Comparison between proposed SFL framework and baselines under extreme conditions on four distinct datasets ($M = 10$).

set of random values for the CPU frequency was extended to $\{0.01, 0.5, 0.8, 1.0, 1.2, 1.6\}$GHz. As suggested, we have added experimental simulation on MNIST, Fashion-MNIST, CIFAR-10, and STL-10 towards the new setup.

In Fig. 9, we demonstrate the comparison between proposed SFL framework and three representative baselines on four distinct datasets. Fig. 9 illustrates the superior performance of our proposed framework under extreme conditions, notably in scenarios characterized by low computational resources or poor network conditions. Our L&S-aware algorithm markedly outpaces baselines skewness-aware, Equal B&MS, and Traditional FL in both convergence speed and test accuracy with configurations $M = 10$. The distinct advantage in time efficiency observed with our framework, especially under challenging conditions, underscores its capacity to excel in extreme scenarios, affirming its effectiveness and adaptability in real-world applications.

VI. Conclusion

In this work, we designed a novel learning mechanism that enhances learning accuracy and expedites convergence. Addressing the challenges of limited communication resources and heterogeneous clients' computing capabilities, we introduced both optimal bandwidth allocation algorithm and optimal model splitting algorithm to achieve adaptive bandwidth allocation and dynamic model split for reducing learning duration. Furthermore, we found a new metric called dataset skewness that affects learning. Thus, by proving the submodularity of skewness and latency-aware algorithms, we proposed a L&S-aware algorithm to solve the issue of heterogeneous clients' data, while reduce the learning time. Experiments shows our approach enhances learning performance.

Appendix

A. Proof of Theorem 1

Suppose the full model denotes $\boldsymbol{w} = [\boldsymbol{w}^c, \boldsymbol{w}^s]$. Let $\boldsymbol{w}_{u,i,j}$ represent the model parameters of client u following j local iterations in the i-th global round, and $\boldsymbol{w}_{i,0}$ represent the global model parameters at the start of the i-th round. We can define the following auxiliary variables in the scenario where FedAvg is employed as the SFL optimizer, and all clients perform j local iterations during each global round i. We have the local loss function $L_u(\boldsymbol{w}) = L_u(\boldsymbol{w}^c, \boldsymbol{w}^s) = \frac{1}{|D_u|}\sum_{\{\boldsymbol{x},y\}\in D_u} l_u(\boldsymbol{x}, y; \boldsymbol{w})$; The global loss function denote $L(\boldsymbol{w}) = L(\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_U) = L([\boldsymbol{w}_1^c, \boldsymbol{w}_1^s], [\boldsymbol{w}_2^c, \boldsymbol{w}_2^s], ..., [\boldsymbol{w}_U^c, \boldsymbol{w}_U^s]) = \frac{1}{|D|}\sum_{u=1}^{U} D_u L_u(\boldsymbol{w})$. Averaged Cumulative Stochastic Gradient is given by

$$\boldsymbol{h}_{u,i} = \frac{1}{J}\sum_{j=0}^{J-1} \nabla L_u(\boldsymbol{w}_{u,i,j}), \tag{28}$$

Averaged Class-wise Gradient is given by

$$\boldsymbol{h}_{u,i,q} = \frac{1}{J}\sum_{j=0}^{J-1} \nabla L_{u,q}(\boldsymbol{w}_{u,i,j}), \tag{29}$$

It is easy to verify that: $\boldsymbol{h}_{u,i} = \sum_{q=1}^{Q} a_{(u,q)} h_{u,i,q}$, and under Assumption 1, it can be demonstrated that $\mathbb{E}[\boldsymbol{h}_{u,i} - \boldsymbol{h}_{u,i}] = 0$. Moreover, as clients process independently, we derive that $\mathbb{E}\langle \boldsymbol{h}_{u,i} - \boldsymbol{h}_{u,i}, \boldsymbol{h}_{u',i} - \boldsymbol{h}_{u',i}\rangle = 0, \forall u \neq u'$. Recall that the update formula for the global model is expressed as follows:

$$\boldsymbol{w}_{i+1,0} - \boldsymbol{w}_{i,0} = -\eta \frac{\sum_{u\in\mathcal{M}} D_u \tilde{\boldsymbol{h}}_{u,i}}{\sum_{u\in\mathcal{M}} D_u}, \tag{30}$$

Because of Lipschitz-smooth assumption for global loss function, we have

$$\begin{aligned}
&\mathbb{E}\left[L\left(\boldsymbol{w}_{i+1,0}^c, \boldsymbol{w}_{i+1,0}^s\right)\right] - L\left(\boldsymbol{w}_{i,0}^c, \boldsymbol{w}_{i,0}^s\right) \\
&\leq -\eta_c \underbrace{\mathbb{E}\left[\left\langle \nabla_{\boldsymbol{w}^c} L\left(\boldsymbol{w}_{i,0}^c, \boldsymbol{w}_{i,0}^s\right), \frac{\sum_{u\in\mathcal{M}} D_u \tilde{\boldsymbol{h}}_{u,i}^c}{\sum_{u\in\mathcal{M}} D_u}\right\rangle\right]}_{A_1} \\
&+ {\eta_c}^2 \frac{1+\chi}{2} L_c \underbrace{\mathbb{E}\left[\left\|\frac{\sum_{u\in\mathcal{M}} D_u \tilde{\boldsymbol{h}}_{u,i}^c}{\sum_{u\in\mathcal{M}} D_u}\right\|^2\right]}_{A_2} \\
&- \eta_s \underbrace{\mathbb{E}\left[\left\langle \nabla_{\boldsymbol{w}^s} L\left(\boldsymbol{w}_{i,0}^c, \boldsymbol{w}_{i,0}^s\right), \frac{\sum_{u\in\mathcal{M}} D_u \tilde{\boldsymbol{h}}_{u,i}^s}{\sum_{u\in\mathcal{M}} D_u}\right\rangle\right]}_{A_3} \\
&+ {\eta_s}^2 \frac{1+\chi}{2} L_s \underbrace{\mathbb{E}\left[\left\|\frac{\sum_{u\in\mathcal{M}} D_u \tilde{\boldsymbol{h}}_{u,i}^s}{\sum_{u\in\mathcal{M}} D_u}\right\|^2\right]}_{A_4},
\end{aligned} \tag{31}$$

We first bound the A_1 , and we should notice that

$$A_1 \overset{(a)}{=} \frac{1}{2}\left\|\nabla_{\boldsymbol{w}^c} L\left(\boldsymbol{w}_{i,0}^c, \boldsymbol{w}_{i,0}^s\right)\right\|^2 + \frac{1}{2}\mathbb{E}\left[\left\|\frac{\sum_{u\in\mathcal{M}} D_u \boldsymbol{h}_{u,i}^c}{\sum_{u\in\mathcal{M}} D_u}\right\|^2\right]$$

$$-\frac{1}{2}\mathbb{E}\left[\left\|\nabla_{\boldsymbol{w}^{\mathrm{c}}}L\left(\boldsymbol{w}_{i,0}^{\mathrm{c}},\boldsymbol{w}_{i,0}^{\mathrm{s}}\right)-\frac{\sum_{u\in\mathcal{M}}D_u\boldsymbol{h}_{u,i}^{\mathrm{c}}}{\sum_{u\in\mathcal{M}}D_u}\right\|^2\right], \tag{32}$$

Where (a) is obtained by adding and subtracting $\frac{\sum_{u\in\mathcal{M}}D_u\boldsymbol{h}_{u,i}^{\mathrm{c}}}{\sum_{u\in\mathcal{M}}D_u}$ into $\frac{\sum_{u\in\mathcal{M}}D_u\tilde{\boldsymbol{h}}_{u,i}^{\mathrm{c}}}{\sum_{u\in\mathcal{M}}D_u}$, and by applying the equation that $2\langle a,b\rangle = \|a\|^2+\|b\|^2-\|a-b\|^2$. Next, we focus on bounding A_2,

$$\begin{aligned}A_2 &\overset{(a)}{\leq} 2\kappa_1^2\mathbb{E}\left[\frac{\sum_{u\in\mathcal{M}}D_u^2}{\left(\sum_{u\in\mathcal{M}}D_u\right)^2}\right]+2\mathbb{E}\left[\left\|\frac{\sum_{u\in\mathcal{M}}D_u\boldsymbol{h}_{u,i}^{\mathrm{c}}}{\sum_{u\in\mathcal{M}}D_u}\right\|^2\right]\\ &\overset{(b)}{\leq} 2\kappa_1^2+2\mathbb{E}\left[\left\|\frac{\sum_{u\in\mathcal{M}}D_u\boldsymbol{h}_{u,i}^{\mathrm{c}}}{\sum_{u\in\mathcal{M}}D_u}\right\|^2\right],\end{aligned} \tag{33}$$

where (a) follows assumption 2 and the fact $\|a+b\|^2 \leq 2\|a\|^2+2\|b\|^2$, (b) comes from $\mathbb{E}[\frac{\sum_{u\in\mathcal{M}}{D_u}^2}{(\sum_{u\in\mathcal{M}}D_u)^2}] = 1$. A_3 and A_4 similar with A_1 and A_2, respectively. Substituting A_1, A_2, A_3, A_4 back into (31), and set $\eta_c \leq \frac{1}{2(1+\chi)L_c}$ and $\eta_s \leq \frac{1}{2(1+\chi)L_s}$, we have $\mathbb{E}[L(\boldsymbol{w}_{i+1,0}^{\mathrm{c}},\boldsymbol{w}_{i+1,0}^{\mathrm{s}})] - L(\boldsymbol{w}_{i,0}^{\mathrm{c}},\boldsymbol{w}_{i,0}^{\mathrm{s}}) \leq -\frac{1}{2}\eta_c\|\nabla_{\boldsymbol{w}^{\mathrm{c}}}L(\boldsymbol{w}_{i,0}^{\mathrm{c}},\boldsymbol{w}_{i,0}^{\mathrm{s}})\|^2 + {\eta_c}^2\kappa_1^2(1+\chi)L_c + \frac{1}{2}\eta_c\mathbb{E}[\|\nabla_{\boldsymbol{w}^{\mathrm{c}}}L(\boldsymbol{w}_{i,0}^{\mathrm{c}},\boldsymbol{w}_{i,0}^{\mathrm{s}}) - \frac{\sum_{u\in\mathcal{M}}D_u\boldsymbol{h}_{u,i}^{\mathrm{c}}}{\sum_{u\in\mathcal{M}}D_u}\|^2] - \frac{1}{2}\eta_s\|\nabla_{\boldsymbol{w}^{\mathrm{s}}}L(\boldsymbol{w}_{i,0}^{\mathrm{c}},\boldsymbol{w}_{i,0}^{\mathrm{s}})\|^2 + {\eta_s}^2\kappa_2^2(1+\chi)L_s + \frac{1}{2}\eta_s\mathbb{E}[\|\nabla_{\boldsymbol{w}^{s}}L(\boldsymbol{w}_{i,0}^{\mathrm{c}},\boldsymbol{w}_{i,0}^{\mathrm{s}}) - \frac{\sum_{u\in\mathcal{M}}D_u\boldsymbol{h}_{u,i}^{\mathrm{s}}}{\sum_{u\in\mathcal{M}}D_u}\|^2]$. Now we focus on bounding

$$\begin{aligned}&\mathbb{E}\left[\left\|\nabla_{\boldsymbol{w}^{\mathrm{c}}}L\left(\boldsymbol{w}_{i,0}^{\mathrm{c}},\boldsymbol{w}_{i,0}^{\mathrm{s}}\right)-\frac{\sum_{u\in\mathcal{M}}D_u\boldsymbol{h}_{u,i}^{\mathrm{c}}}{\sum_{u\in\mathcal{M}}D_u}\right\|^2\right]\\ &\overset{(a)}{=} \underbrace{2\,\mathbb{E}\left[\left\|\sum_{q=1}^{Q}(\frac{1}{Q}-\frac{\sum_{u\in\mathcal{M}}D_u a_{(u,q)}}{\sum_{u\in\mathcal{M}}D_u})\nabla_{\boldsymbol{w}^{\mathrm{c}}}L_q\left(\boldsymbol{w}_{i,0}^{\mathrm{c}},\boldsymbol{w}_{i,0}^{\mathrm{s}}\right)\right\|^2\right]}_{B_1}\\ &+\underbrace{2\,\mathbb{E}\left[\left\|\sum_{q=1}^{Q}\frac{\sum_{u\in\mathcal{M}}D_u a_{(u,q)}[\nabla_{\boldsymbol{w}^{\mathrm{c}}}L_q\left(\boldsymbol{w}_{i,0}^{\mathrm{c}},\boldsymbol{w}_{i,0}^{\mathrm{s}}\right)-\boldsymbol{h}_{u,i,q}^{\mathrm{c}}]}{\sum_{u\in\mathcal{M}}D_u}\right\|^2\right]}_{B_2},\end{aligned} \tag{34}$$

where (a) is based on the principle that $\|a+b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$. And $\mathbb{E}[\|\nabla_{\boldsymbol{w}^{s}}L(\boldsymbol{w}_{i,0}^{\mathrm{c}},\boldsymbol{w}_{i,0}^{\mathrm{s}}) - \frac{\sum_{u\in\mathcal{M}}D_u\boldsymbol{h}_{u,i}^{\mathrm{s}}}{\sum_{u\in\mathcal{M}}D_u}\|^2]$ has the similar derivation process. For B_1, according to Cauchy-Schwarz inequality and Assumption 3, we have

$$B_1 \overset{(a)}{\leq} Q\delta_1\left\|\nabla_{\boldsymbol{w}^{\mathrm{c}}}L\left(\boldsymbol{w}_{i,0}^{\mathrm{c}},\boldsymbol{w}_{i,0}^{\mathrm{s}}\right)\right\|^2\mathbb{E}\left[\mathcal{G}(\mathcal{M})\right]+{\psi_1}^2\mathbb{E}\left[\mathcal{G}(\mathcal{M})\right], \tag{35}$$

where (a) follows Assumption 3 and Cauchy-Schwarz inequality. For B_2, according to our Assumption 4, we have

$$\begin{aligned}B_2 &\overset{(a)}{\leq} 2\zeta^2+\\ &\underbrace{2\,\mathbb{E}\left[\left\|\sum_{q=1}^{Q}\frac{\sum_{u\in\mathcal{M}}D_u a_{(u,q)}[\nabla_{\boldsymbol{w}^{\mathrm{c}}}L_{u,q}\left(\boldsymbol{w}_{i,0}^{\mathrm{c}},\boldsymbol{w}_{i,0}^{\mathrm{s}}\right)-h_{u,i,q}^{\mathrm{c}}]}{\sum_{u\in\mathcal{M}}D_u}\right\|^2\right]}_{C_1},\end{aligned} \tag{36}$$

where (a) is derived by the operations including adding and subtracting $\nabla_{\boldsymbol{w}^{\mathrm{c}}}L_{u,q}\left(\boldsymbol{w}_{i,0}^{\mathrm{c}},\boldsymbol{w}_{i,0}^{\mathrm{s}}\right)$, using $\|a+b\|^2 \leq 2\|a\|^2+2\|b\|^2$, and $\sum_{q=1}^{Q}\frac{\sum_{u\in\mathcal{M}}D_u a_{(u,q)}}{\sum_{u\in\mathcal{M}}D_u} = 1$. B_4 has the similar result to B_2. where $\zeta = \max_{\{u,q\}}\zeta_{\{u,q\}}$ and $\varsigma = \max_{\{u,q\}}\varsigma_{\{u,q\}}$. For C_1, we have

$$\begin{aligned}&\mathbb{E}[\|\nabla_{\boldsymbol{w}^{\mathrm{c}}}L_{(u,q)}(\boldsymbol{w}_{i,0}^{\mathrm{c}},\boldsymbol{w}_{i,0}^{\mathrm{s}})-\boldsymbol{h}_{u,i,q}^{\mathrm{c}}\|^2]\\ &\overset{(a)}{=}\mathbb{E}[\|\frac{1}{J}\sum_{j=0}^{J-1}(\nabla_{\boldsymbol{w}^{\mathrm{c}}}L_{(u,q)}(\boldsymbol{w}_{i,0}^{\mathrm{c}},\boldsymbol{w}_{i,0}^{\mathrm{s}})-\nabla L_{(u,q)}(\boldsymbol{w}_{u,i,j}^{\mathrm{c}}))\|^2]\\ &\overset{(b)}{\leq}\frac{L_c^2}{J}\sum_{j=0}^{J-1}\{\mathbb{E}[\|\boldsymbol{w}_{i,0}^{\mathrm{c}}-\boldsymbol{w}_{u,i,j}^{\mathrm{c}}\|^2]\},\end{aligned} \tag{37}$$

where (a) is derived by substituting $\boldsymbol{h}_{u,i,q}^{\mathrm{c}}$ with $\frac{1}{J}\sum_{j=0}^{J-1}\nabla L_{(u,q)}(\boldsymbol{w}_{u,i,j}^{\mathrm{c}})$, (b) uses Jensens Inequality and Assumption 1. Now we shift our focus to bounding the difference between global model and local model. We focus on $\mathbb{E}[\|\boldsymbol{w}_{i,0}-\boldsymbol{w}_{u,i,j}\|^2]$, plugging into the update rule:

$$\begin{aligned}&\mathbb{E}[\|\boldsymbol{w}_{i,0}-\boldsymbol{w}_{u,i,j}\|^2] = \eta_c^2\mathbb{E}[\|\tilde{\nabla}L_{(u,q)}(\boldsymbol{w}_{u,i,j}^{\mathrm{c}})\|^2]\\ &\leq 2\eta_c^2\kappa_c^2+2\eta_c^2\mathbb{E}[\|\nabla L_{(u,q)}(\boldsymbol{w}_{u,i,j}^{\mathrm{c}})\|^2].\end{aligned} \tag{38}$$

We can bound the second term by the subsequent inequality:

$$\begin{aligned}&\mathbb{E}\left[\left\|\nabla L_u(\boldsymbol{w}_{u,i,j}^{\mathrm{c}})\right\|^2\right]\\ &\leq 2L_c^2\mathbb{E}\left[\left\|\boldsymbol{w}_{i,0}^{\mathrm{c}}-\boldsymbol{w}_{u,i,j}^{\mathrm{c}}\right\|^2\right]+2\mathbb{E}\left[\left\|\nabla L_u(\boldsymbol{w}_{i,0}^{\mathrm{c}})\right\|^2\right].\end{aligned} \tag{39}$$

Substituting (39) into (38), and finishing minor rearranging, it follows that

$$\begin{aligned}&\frac{1}{J}\sum_{j=0}^{J-1}\left\{\mathbb{E}\left[\left\|\boldsymbol{w}_{i,0}^{\mathrm{c}}-\boldsymbol{w}_{u,i,j}^{\mathrm{c}}\right\|^2\right]\right\}\leq\frac{2\eta_c^2\kappa_1^2(J-1)}{1-4\eta_c^2L_c^2J(J-1)}\\ &+\frac{4\eta_c^2J(J-1)}{1-4\eta_c^2L_c^2J(J-1)}\mathbb{E}\left[\left\|\nabla L_{(u,q)}(\boldsymbol{w}_{i,0}^{\mathrm{c}})\right\|^2\right].\end{aligned} \tag{40}$$

Define $K_1 = 4{\eta_c}^2{L_c}^2J\,(J-1)$, and according to the proof C.5 in [16], we have

$$\begin{aligned}&\mathbb{E}\left[\left\|\nabla_{\boldsymbol{w}^{\mathrm{c}}}L_{u,q}\left(\boldsymbol{w}_{i,0}^{\mathrm{c}},\boldsymbol{w}_{i,0}^{\mathrm{s}}\right)-\boldsymbol{h}_{u,i,q}^{\mathrm{c}}\right\|^2\right]\\ &\leq\frac{L_c^2}{J}\sum_{j=0}^{J-1}\mathbb{E}\left[\|\boldsymbol{w}_{i,0}-\boldsymbol{w}_{u,i,j}\|^2\right]\overset{(a)}{\leq}\frac{2\eta_c^2L_c^2\kappa_1^2}{1-K_1}(J-1)\\ &+\frac{2K_1}{1-K_1}\mathbb{E}\left[\left\|\nabla_{\boldsymbol{w}^{\mathrm{c}}}L\left(\boldsymbol{w}_{i,0}^{\mathrm{c}},\boldsymbol{w}_{i,0}^{\mathrm{s}}\right)\right\|^2\right]+\frac{1}{Q}\frac{2K_1}{1-K_1}\zeta^2,\end{aligned} \tag{41}$$

where $L_c = \max_{\{u,q\}}L_{c,\{u,q\}}$, (a) comes from Assumption 4 and using $\|a+b\|^2 \leq 2\|a\|^2+2\|b\|^2$. Combine (35), (36), (41), and server-side formulas. Plugging into $\mathbb{E}[L(\boldsymbol{w}_{i+1,0}^{c},\boldsymbol{w}_{i+1,0}^{s})] - L(\boldsymbol{w}_{i,0}^{c},\boldsymbol{w}_{i,0}^{s})$, it is easy to derive that

$$\begin{aligned}&\mathbb{E}\left[L\left(\boldsymbol{w}_{i+1,0}^{\mathrm{c}},\boldsymbol{w}_{i+1,0}^{\mathrm{s}}\right)\right]-L\left(\boldsymbol{w}_{i,0}^{\mathrm{c}},\boldsymbol{w}_{i,0}^{\mathrm{s}}\right)\\ &\leq-\eta_c(\frac{1}{2}-B\delta_1\mathbb{E}\left[\mathcal{G}(\mathcal{M})\right]-\frac{4K_1}{1-K_1})\left\|\nabla_{\boldsymbol{w}^{\mathrm{c}}}L\left(\boldsymbol{w}_{i,0}^{\mathrm{c}},\boldsymbol{w}_{i,0}^{\mathrm{s}}\right)\right\|^2\\ &+\frac{4\eta_c^3L_c^2\kappa_1^2}{1-K_1}(J-1)+\frac{1}{B}\frac{2K_1}{1-K_1}\eta_c\zeta^2+{\eta_c}^2\kappa_1^2(1+\chi)L_c\\ &+{\psi_1}^2\eta_c\mathbb{E}\left[\mathcal{G}(\mathcal{M})\right]+4\eta_c\zeta^2+4\eta_s\varsigma^2+{\eta_s}^2\kappa_2^2(1+\chi)L_s\\ &-\eta_s(\frac{1}{2}-B\delta_2\mathbb{E}[\mathcal{G}(\mathcal{M})]-\frac{4K_2}{1-K_2})\|\nabla_{\boldsymbol{w}^{\mathrm{s}}}L(\boldsymbol{w}_{i,0}^{\mathrm{c}},\boldsymbol{w}_{i,0}^{\mathrm{s}})\|^2\end{aligned}$$

$$+\frac{1}{B}\frac{2K_2}{1-K_2}\eta_s\varsigma^2+\frac{4\eta_s^3L_s^2\kappa_2^2}{1-K_2}(J-1)+{\psi_2}^2\eta_s\mathbb{E}[\mathcal{G}(\mathcal{M})]. \tag{42}$$

Now we denote $L(\boldsymbol{w}_i)$ the global model at i-th global round, and subtracting $L(\boldsymbol{w}^*)$ into both $L(\boldsymbol{w}_{i+1})$ and $L(\boldsymbol{w}_i)$. By utilizing the L-smooth of loss functions, changing the notations and substituting them into (42):

$$\begin{aligned}&L(\boldsymbol{w}_{i+1})-L(\boldsymbol{w}^*)\le H_1\left(L(\boldsymbol{w}_i)-L(\boldsymbol{w}^*)\right)+H_2\\&+({\psi_1}^2\eta_c+{\psi_2}^2\eta_s)\mathbb{E}[\mathcal{G}(\mathcal{M})],\end{aligned} \tag{43}$$

where $H_1=1-L_c\eta_c(1-\frac{8K_1}{1-K_1})-L_s\eta_s(1-\frac{8K_2}{1-K_2})+(L_c\eta_c\delta_1+L_s\eta_s\delta_2)2B\mathbb{E}[\mathcal{G}(\mathcal{M})]$, and $H_2=4\eta_c\zeta^2+4\eta_s\xi^2+(\frac{\eta_c^3L_c^2\kappa_1^2}{1-K_1}+\frac{\eta_s^3L_s^2\kappa_2^2}{1-K_2})4(J-1)+\frac{1}{B}\frac{2K_1}{1-K_1}\eta_c\zeta^2+\frac{1}{B}\frac{2K_2}{1-K_2}\eta_s\xi^2+{\eta_c}^2L_c\kappa_1^2(1+\chi)+{\eta_s}^2L_s\kappa_2^2(1+\chi)$. By applying Inequalities Iteratively, we have $L(\boldsymbol{w}_I)-L(\boldsymbol{w}^*)\le H_1^I(L(\boldsymbol{w}_0)-L(\boldsymbol{w}^*))+\sum_{i=0}^{I-1}H_1^i\{H_2+({\psi_1}^2\eta_c+{\psi_2}^2\eta_s)\mathbb{E}[\mathcal{G}(\mathcal{M})]\}$. $\sum_{i=0}^{I-1}H_1^i$ is geometric series, if $H_1\neq1$, $\sum_{i=0}^{I-1}H_1^i=\frac{1-H_1^I}{1-H_1}$. The global loss after I rounds is $L(\boldsymbol{w}_I)-L(\boldsymbol{w}^*)\le H_1^I(L(\boldsymbol{w}_0)-L(\boldsymbol{w}^*))+\frac{1-H_1^I}{1-H_1}\{H_2+({\psi_1}^2\eta_c+{\psi_2}^2\eta_s)\mathbb{E}[\mathcal{G}(\mathcal{M})]\}$, we set function $f(H_1)=\frac{1-H_1^I}{1-H_1}(I\ge1)$, for proving that when $H_1\neq1$, as H_1 increases, the value of the entire function increases. The differentiation of the original function is given by $f'(H_1)=\frac{IH_1^I(H_1-1)-H_1(H_1^I-1)}{H_1(H_1-1)^2}$, where $H_1(H_1-1)^2$ is positive for $H_1>1$ and $0<H_1<1$. This general proof outlines the intuitive reasoning behind why $f(H_1)$ is monotonically increasing with an increase in H_1. As a result, $\mathcal{G}(\mathcal{M})$ increases, H_1 increases as well, the right part of (43) will increase, vice versa. Proof complete.

B. Proof of Lemma 2

It is evident from (8) that $T_{u,i}^{\mathrm{CM}}$ is a monotonically decreasing function with respect to $n_{u,i}$. Each client needs different time consumption for learning due to heterogenous communication and computing capability, thus, the one-round latency is dictated by the slowest client. For the clients complete the local gradient computation earlier than others, we can redistribute some of their bandwidth to the slower clients until all clients finish learning simultaneously. Consequently, the optimal solution for $\mathcal{P}_1$ is realized when the total bandwidth is distributed among all scheduled clients to ensure they complete together. Thus, the optimal bandwidth allocation policy fulfills

$$\begin{cases}T_{u,i}^{\mathrm{CM}}+T_{u,i}^{\mathrm{CP}}=\mathcal{T}_i^*(\mathcal{M}_i), & \forall u\in\mathcal{M}_i,\\ \sum_{u\in\mathcal{M}_i}n_u=1,\end{cases} \tag{44}$$

where $\mathcal{T}_i^*(\mathcal{M}_i)$ is the optimal latency in round i. By solving (44), the proof is completed.

C. Proof of Lemma 3

To simplify the explanation, we initially define the smashed data and client-side model uploading latency for client u as $T_{u,i}^{\mathrm{CP,min}}=\frac{\nu_{u,i}}{\beta\log_2(1+\frac{p_uh_{u,i}}{\sigma^2})}$, which is calculated assuming client u utilizes the entire bandwidth for transmission. We then demonstrate that the optimal latency $\mathcal{T}_i^*(\mathcal{M})$ is a function that increases monotonically in relation to the client subset $\mathcal{M}$. Based on Lemma 2, for client subsets $\mathcal{M}_1\subseteq\mathcal{M}_2\subseteq\mathcal{U}$, we have $\sum_{u\in\mathcal{M}_1}\frac{T_{u,i}^{\mathrm{CM,min}}}{\mathcal{T}_i^*(\mathcal{M}_1)-T_{u,i}^{\mathrm{CP}}}=\sum_{u\in\mathcal{M}_1}\frac{T_{u,i}^{\mathrm{CM,min}}}{\mathcal{T}_i^*(\mathcal{M}_2)-T_{u,i}^{\mathrm{CP}}}+\sum_{u\in\mathcal{M}_2\setminus\mathcal{M}_1}\frac{T_{u,i}^{\mathrm{CM,min}}}{\mathcal{T}_i^*(\mathcal{M}_2)-T_{u,i}^{\mathrm{CP}}}$. By rearranging this equation, we have

$$\begin{aligned}&\sum\nolimits_{u\in\mathcal{M}_1}\left(\frac{T_{u,i}^{\mathrm{CM,min}}}{\mathcal{T}_i^*(\mathcal{M}_1)-T_{u,i}^{\mathrm{CP}}}-\frac{T_{u,i}^{\mathrm{CM,min}}}{\mathcal{T}_i^*(\mathcal{M}_2)-T_{u,i}^{\mathrm{CP}}}\right)\\&=\sum\nolimits_{u\in\mathcal{M}_2\setminus\mathcal{M}_1}\frac{T_{u,i}^{\mathrm{CM,min}}}{\mathcal{T}_i^*(\mathcal{M}_2)-T_{u,i}^{\mathrm{CP}}}>0.\end{aligned} \tag{45}$$

Thus, we have $\mathcal{T}_i^*(\mathcal{M}_1)\le\mathcal{T}_i^*(\mathcal{M}_2)$. That is, $\mathcal{T}_i^*(\mathcal{M})$ increases monotonically with the selected client set $\mathcal{M}$. Likewise, for client $m\in\mathcal{U}\setminus\mathcal{M}_2$, we have

$$\begin{aligned}&\sum\nolimits_{u\in\mathcal{M}_1}\left(\frac{T_{u,i}^{\mathrm{CM,min}}}{\mathcal{T}_i^*(\mathcal{M}_1+m)-T_{u,i}^{\mathrm{CP}}}-\frac{T_{u,i}^{\mathrm{CM,min}}}{\mathcal{T}_i^*(\mathcal{M}_1)-T_{u,i}^{\mathrm{CP}}}\right)\\&+\frac{T_{m,i}^{\mathrm{CM,min}}}{\mathcal{T}_i^*(\mathcal{M}_1+m)-T_{u,i}^{\mathrm{CP}}}=0.\end{aligned} \tag{46}$$

By rearranging the above equation, we have $\mathcal{T}_i^*(\mathcal{M}_1+m)-\mathcal{T}_i^*(\mathcal{M}_1)=\frac{T_{m,i}^{\mathrm{CM,min}}}{1+\sum_{u\in\mathcal{M}_1}\frac{T_{u,i}^{\mathrm{CM,min}}}{\mathcal{T}_i^*(\mathcal{M}_1)-T_{u,i}^{\mathrm{CP}}}\frac{T_{u,i}^{\mathrm{CP}}-T_{m,i}^{\mathrm{CP}}}{\mathcal{T}_i^*(\mathcal{M}_1+m)-T_{u,i}^{\mathrm{CP}}}}$. Since $\mathcal{T}_i^*(\mathcal{M}_1+m)<\mathcal{T}_i^*(\mathcal{M}_2+m)$, based on (45), we have $\mathcal{T}_i^*(\mathcal{M}_1+m)-\mathcal{T}_i^*(\mathcal{M}_1)<\mathcal{T}_i^*(\mathcal{M}_2+m)-\mathcal{T}_i^*(\mathcal{M}_2)$.

In addition, for $\mathcal{G}(\mathcal{M}_i)$, following the proof of Lemma 3 in [15], we have $\min_{m\in\mathcal{M}_1}\|\nabla L_u(\boldsymbol{w}_i)-\nabla L_m(\boldsymbol{w}_i)\|\ge\min_{m\in\mathcal{M}_2}\|\nabla L_u(\boldsymbol{w}_i)-\nabla L_m(\boldsymbol{w}_i)\|$. Hence, $\mathcal{G}(\{m\}\cup\mathcal{M}_1)-\mathcal{G}(\mathcal{M}_1)\le\mathcal{G}(\{m\}\cup\mathcal{M}_2)-\mathcal{G}(\mathcal{M}_2)$, the proof is completed.

REFERENCES

[1] C. Xie, Z. Chen, W. Yi, H. Shin, and A. Nallanathan, "Heterogeneity-Aware client scheduling for split federated learning in Resource-Scarce networks," in *2024 IEEE Globecom Workshops (GC Wkshps)*, Cape Town, South Africa, Dec. 2024.

[2] L. U. Khan, W. Saad, Z. Han, E. Hossain, and C. S. Hong, "Federated learning for internet of things: Recent advances, taxonomy, and open challenges," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 1759–1799, 2021.

[3] T. Wu, C. Pan, K. Zhi, H. Ren, M. Elkashlan, C.-X. Wang, R. Schober, and X. You, "Exploit high-dimensional ris information to localization: What is the impact of faulty element?" *IEEE J. Sel. Areas Commun.*, pp. 1–1, 2024.

[4] X. Chen, G. Zhu, H. Ding, L. Zhang, H. Zhang, and Y. Fang, "End-to-end service auction: A general double auction mechanism for edge computing services," *IEEE/ACM Trans. Netw.*, vol. 30, no. 6, pp. 2616–2629, 2022.

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

[6] Z. Yang, M. Chen, K.-K. Wong, H. V. Poor, and S. Cui, "Federated learning for 6G: Applications, challenges, and opportunities," *Engineering*, vol. 8, pp. 33–41, 2022.

[7] 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

[8] 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

[9] W. Ni, H. Ao, H. Tian, Y. C. Eldar, and D. Niyato, "Fedsl: Federated split learning for collaborative healthcare analytics on resource-constrained wearable iomt devices," *IEEE Internet Things J.*, vol. 11, no. 10, pp. 18 934–18 935, 2024.

[10] D. Makhija, N. Ho, and J. Ghosh, "Federated self-supervised learning for heterogeneous clients," 2022. [Online]. Available: https://arxiv.org/abs/2205.12493

[11] J. Dan, W. Liu, C. Xie, H. Yu, S. Dong, and Y. Tan, "Tfgda: Exploring topology and feature alignment in semi-supervised graph domain adaptation through robust clustering," in *The Thirty-eighth Annual Conference on Neural Inf. Process. Syst. (NIPS)*.

[12] J. Dan, Y. Liu, J. Deng, H. Xie, S. Li, B. Sun, and S. Luo, "Topofr: A closer look at topology alignment on face recognition," *arXiv preprint arXiv:2410.10587*, 2024.

[13] X. Li, Z. Song, and J. Yang, "Federated adversarial learning: A framework with convergence analysis," 2022. [Online]. Available: https://arxiv.org/abs/2208.03635

[14] J. Dan, M. Liu, C. Xie, J. Yu, H. Xie, R. Li, and S. Dong, "Similar norm more transferable: Rethinking feature norms discrepancy in adversarial domain adaptation," *Knowledge-Based Systems*, vol. 296, p. 111908, 2024.

[15] Z. Chen, W. Yi, and A. Nallanathan, "Exploring representativity in device scheduling for wireless federated learning," *IEEE Trans. Wireless Commun.*, vol. 23, no. 1, pp. 720–735, 2024.

[16] L. Wang, S. Xu, X. Wang, and Q. Zhu, "Addressing class imbalance in federated learning," 2020. [Online]. Available: https://arxiv.org/abs/2008.06217

[17] W. Tong and G. Y. Li, "Nine challenges in artificial intelligence and wireless communications for 6g," *IEEE Wireless Commun.*, vol. 29, no. 4, pp. 140–145, 2022.

[18] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 269–283, 2021.

[19] 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.

[20] M. M. Amiri and D. Gndz, "Federated learning over wireless fading channels," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3546–3557, 2020.

[21] J. Zhang, A. Li, M. Tang, J. Sun, X. Chen, F. Zhang, C. Chen, Y. Chen, and H. Li, "Fed-cbs: A heterogeneity-aware client sampling mechanism for federated learning via class-imbalance reduction," 2023. [Online]. Available: https://arxiv.org/abs/2209.15245

[22] M. Yang, X. Wang, H. Zhu, H. Wang, and H. Qian, "Federated learning with class imbalance reduction," in *2021 29th European Signal Processing Conference (EUSIPCO)*, 2021, pp. 2174–2178.

[23] R. Ye, M. Xu, J. Wang, C. Xu, S. Chen, and Y. Wang, "Feddisco: Federated learning with discrepancy-aware collaboration," 2023. [Online]. Available: https://arxiv.org/abs/2305.19229

[24] B. Yin, Z. Chen, and M. Tao, "Predictive gan-powered multi-objective optimization for hybrid federated split learning," *IEEE Trans. Commun.*, vol. 71, no. 8, pp. 4544–4560, 2023.

[25] Z. Lin, G. Qu, W. Wei, X. Chen, and K. K. Leung, "Adaptsfl: Adaptive split federated learning in resource-constrained edge networks," 2024. [Online]. Available: https://arxiv.org/abs/2403.13101

[26] C. Xu, J. Li, Y. Liu, Y. Ling, and M. Wen, "Accelerating split federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, vol. 23, no. 6, pp. 5587–5599, 2024.

[27] E. Abbasnejad, J. Q. Shi, and A. van den Hengel, "Deep lipschitz networks and dudley gans," 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:125599652

[28] K. Scaman and A. Virmaux, "Lipschitz regularity of deep neural networks: analysis and efficient estimation," 2019. [Online]. Available: https://arxiv.org/abs/1805.10965

[29] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," 2020. [Online]. Available: https://arxiv.org/abs/1907.02189

[30] Y. J. Cho, J. Wang, and G. Joshi, "Client selection in federated learning: Convergence analysis and power-of-choice selection strategies," 2020. [Online]. Available: https://arxiv.org/abs/2010.01243

[31] 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.

[32] J. Ren, Y. He, D. Wen, G. Yu, K. Huang, and D. Guo, "Scheduling for cellular federated edge learning with importance and channel awareness," *IEEE Trans. Wireless Commun.*, vol. 19, no. 11, pp. 7690–7703, 2020.

[33] Z. Chen, W. Yi, H. Shin, A. Nallanathan, and G. Y. Li, "Efficient wireless federated learning with partial model aggregation," *IEEE Trans. Commun.*, vol. 72, no. 10, pp. 6271–6286, 2024.

[34] 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.

[35] A. Rafiey and Y. Yoshida, "Fast and private submodular and k-submodular functions maximization with matroid constraints," 2020. [Online]. Available: https://arxiv.org/abs/2006.15744

[36] A. Krause and D. Golovin, "Submodular function maximization." *Tractability*, vol. 3, no. 71-104, p. 3, 2014.

[37] X. Pan, S. Jegelka, J. E. Gonzalez, J. K. Bradley, and M. I. Jordan, "Parallel double greedy submodular maximization," in *Proc. Neural Inf. Process. Syst. (NIPS)*, vol. 27, 2014, pp. 1–9.