PYRAMIDSFL: RESOURCE-AWARE SPLIT FEDERATED LEARNING VIA ADAPTIVE MODEL SPLITTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Split Federated Learning (SFL) has emerged as a promising distributed learning paradigm that offloads part of the on-device training workload to a resource-rich server while keeping raw data at the edge. However, its effectiveness is often challenged by heterogeneity in both data silos and client resources, which jointly undermine model performance and training efficiency. This paper proposes PyramidSFL, a system-efficient Stacked-Activation SFL (SA-SFL) framework that dynamically partitions the model between clients and the server according to each client's communication and computation profile. To counteract the accuracy drop caused by dynamic model splitting, PyramidSFL combines a warm-up strategy that stabilizes early global optimization with a weight-shrinking mechanism that reduces cross-layer update inconsistency. This enables efficient training without sacrificing model quality, leading to a better time-to-accuracy performance. Extensive experiments on real-world benchmarks show that PyramidSFL consistently improves the accuracy-latency trade-off over existing FL and SFL baselines, improving global model accuracy by up to 15.53% under the same wall-clock budget and reducing convergence latency by up to 72.86%. Its gains become more significant under tighter client resource constraints and deeper task models, highlighting its effectiveness for heterogeneous and resource-constrained edge training.

1 INTRODUCTION

The rapid proliferation of mobile devices equipped with sensing, processing, and networking capabilities (Gubbi et al., 2013; Fortino & Trunfio, 2014) has generated rich data at the edge, enabling a wide range of cyber-physical services (Wang et al., 2020b; Verbraeken et al., 2020). This trend has spurred growing research on distributed learning frameworks that collectively exploit data silos and resources across devices to train task models. Among them, Federated Learning (FL) (Konečnỳ et al., 2016; McMahan et al., 2017; Bonawitz et al., 2019) has emerged as a privacy-preserving paradigm where devices (i.e., clients) share locally updated model parameters rather than raw samples. The on-device model training in FL, however, poses practical challenges to edge devices with limited resources. To address this issue, prior work has explored collaborative training between resource-constrained clients and more powerful server clusters (Li et al., 2018; Eshratifar et al., 2019; Wang et al., 2021). In this context, Split Federated Learning (SFL) (Gupta & Raskar, 2018; Vepakomma et al., 2018b; Singh et al., 2019; Gao et al., 2020; Thapa et al., 2022) partitions a full model into client-side and server-side submodels that are separately updated on the edge and cloud, thereby offloading partial training workload to the resource-rich server while keeping raw data at the edge.

Beyond merely offloading training workload, the server in SFL also directly steers the learning pace of the server-side submodel(s), making the global model behave differently from–and potentially outperform–vanilla FedAvg (McMahan et al., 2017) without model splitting. Specifically, while the client-side submodels in SFL are updated in parallel (akin to FL), the update of the server-side submodel(s) varies: 1) in SFLV1 (Thapa et al., 2022), the server maintains a surrogate server-side submodel for each client and updates each client-server pair independently; 2) in SFLV2 (Thapa et al., 2022), the server instead maintains a single server-side submodel and updates it by sequential stochastic gradient descent (SGD) on the offloaded activations (also called *smashed data*) from clients. From a system perspective, SFLV1 incurs a high memory demand to store multiple surrogate submodels and optimizers, whereas SFLV2 suffers from high training latency due to its sequential updates, and both limitations are exacerbated as the scale of clients grows. Stacked-Activation SFL (SA-SFL) (Liao et al., 2024a) shown in Figure 1(left), can be viewed as a variant of SFLV2 where the

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

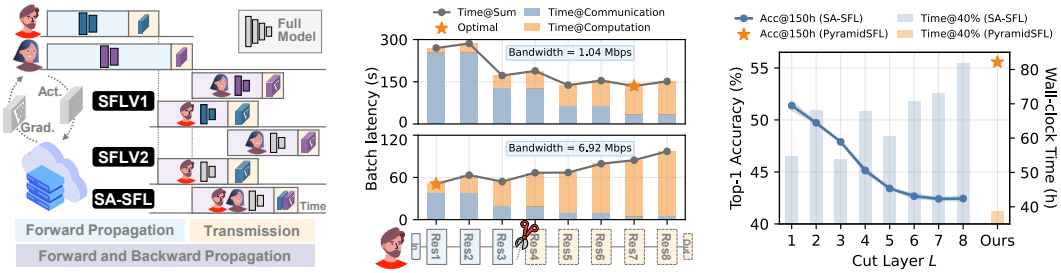

Figure 1: Motivation study with ResNet-18 on CIFAR-100. **SFL Frameworks (left):** Three SFL frameworks differ in the server-side submodel(s) update, while SA-SFL better balances latency and memory cost. **Latency-wise Splitting (middle):** Under heterogeneous client resources, the latency-optimal cut layer L at residual-block granularity varies across clients due to the trade-off between activations/gradients communication and on-device computation. **Accuracy-wise Splitting (right):** SA-SFL is sensitive to L. Our PyramidSFL is tailored to SA-SFL by enabling latency-aware splitting while improving global accuracy, thereby achieving better time-to-accuracy performance.

server updates a shared submodel on a stacked batch of activations from all involved clients at each training step. By doing so, it offers a better balance between training latency and memory demand.

In practical settings, devices are endowed with heterogeneous resources such as communication bandwidth and computing power, which make a significant impact on the optimal model splitting and the resulting performance. Figure 1(middle) shows, under two bandwidth conditions, the batch latency as a function of model splitting at different layers for training at the different clients and the server. The optimal cut layers across devices vary under different bandwidth conditions: higher bandwidth implies reduced communication latency for transmitting activations/gradients, which allows for more layers to be trained on the server (i.e., a smaller cut layer L). The cut layer further influences the performance of SFL, as shown in Figure 1(right). For instance, when all participating clients adopt a constant cut layer L, SA-SFL tends to yield better accuracy with shallower splitting, i.e., when a larger fraction of the model is assigned to the server (i.e., a smaller L).

In this paper, we propose PyramidSFL, a novel resource-aware SA-SFL that applies client-specific model splitting according to local resource profiles. Under the latency-optimal split, PyramidSFL updates a shared server-side submodel using activations offloaded from client-side submodels at varying depths. However, such dynamic splitting can noticeably degrade both model accuracy and convergence in SA-SFL for two main reasons: 1) SA-SFL inherently benefits from shallower splitting, and therefore, the latency-optimal split is not necessarily the one that is performance-effective; 2) dynamic splitting causes different layers of the shared server-side submodel to be updated with different mixtures of activation batches, as clients split at different depths traverse different server-side layers. This results in layer-wise training inconsistency for the server-side submodel. These observations raise the following question: *Can we enable dynamic model splitting to further reduce training latency–while counteracting the negative optimization inconsistency introduced across server-side submodel layers–thereby reaching high time-to-accuracy performance?*

To this end, PyramidSFL introduces two training strategies for server-side submodel optimization: 1) a *bandwidth-prioritized warm-up* strategy that bootstraps training with a static shallow split during the initial communication rounds, with a preference for sampling clients with high bandwidth. It stabilizes the early-stage search for the global optimum, at the cost of a small increase in overall wall-clock latency; and 2) a *layer-wise weight-shrinking* mechanism that regularizes server-side updates by accounting for both the size of effective activation subsets and the gradient variance across different layers. In this way, PyramidSFL retains the benefits of low memory overhead and stacked-batch training efficiency in SA-SFL, while explicitly mitigating the optimization inconsistency from dynamic splitting. Our contributions are as follows:

- We design PyramidSFL, a resource-aware SA-SFL framework that supports varying cut layers to better balance computation and communication efficiency under system heterogeneity. To mitigate the performance degradation caused by dynamic model splitting, we further introduce a warm-up strategy and a weight-shrinking mechanism to improve model performance.

- We present a theoretical analysis of SA-SFL and compare it with two representative SFL regimes, i.e., SFLV1, SFLV2, under a unified framework. Analysis clarifies their convergence behaviors and reveals how different server-side synchronization schemes affect the global optimization.
- Extensive experiments demonstrate that PyramidSFL consistently achieves significant gains in time-to-accuracy performance compared with strong baselines across diverse settings. The results show that our framework effectively improves both training efficiency and global model performance, highlighting its practical value for efficient Split Federated Learning.

2 PYRAMIDSFL DESIGN

Typically, a distributed learning system consists of a central server and a set of edge devices $\mathcal{J}$, where each device $j \in \mathcal{J}$ possesses a local data silo $\mathcal{D}_j$ and is endowed with uplink/downlink bandwidth b_j and computation capability o_j^d. The server coordinates the clients to collaboratively train a task model $f^{\mathcal{W}}$ with weight parameters $\mathcal{W}$. In SFL, given a cut layer index $L \in [|\mathcal{W}|]$ ($[|\mathcal{W}|] = \{1, \cdots, |\mathcal{W}|\}$), the full model is split into the client-side submodel $f^{\mathcal{W}_c}$ and the server-side submodel $f^{\mathcal{W}_s}$: $f^{\mathcal{W}}(\cdot) = f^{\mathcal{W}_s}(f^{\mathcal{W}_c}(\cdot))$, where $\mathcal{W} = [\mathcal{W}_c, \mathcal{W}_s] = [\mathcal{W}_1, \cdots, \mathcal{W}_L, \mathcal{W}_{L+1}, \cdots, \mathcal{W}_{|\mathcal{W}|}]$, satisfying $\mathcal{W} \in \mathbb{R}^d$, $\mathcal{W}_c \in \mathbb{R}^{d_c}$, $\mathcal{W}_s \in \mathbb{R}^{d_s}$, and $d = d_c + d_s$[1].

Data Heterogeneity. The local dataset at client j is drawn from a client-specific distribution $\mathcal{P}_j(x, y)$, and we allow $\mathcal{P}_{j_1}(x, y) \neq \mathcal{P}_{j_2}(x, y)$ for $j_1 \neq j_2$. Following the common non-identically and/or non-independently distributed (non-IID) taxonomy (Hsu et al., 2019; Li et al., 2022), we factorize $\mathcal{P}_j(x, y) = \mathcal{P}_j(x|y)\mathcal{P}_j(y)$ and focus on heterogeneity with *label skew* and *quantity skew*, where the marginal label distribution $\mathcal{P}_j(y)$ and sizes of data silos $|\mathcal{D}_j|$ differ across clients, respectively.

System Heterogeneity. Clients may differ significantly in both networking b_j and computing o_j^d. These resources may also vary over time in practice. We can quantify o_j^d in seconds per normalized unit of computation, such as on-device inference throughput (*s/frame*), for a fixed model and task. The server is assumed to be resource-rich, with computation coefficient o^s, where $o^s < o_j^d, \forall j \in \mathcal{J}$.

Problem Formulation. Given the local resource profiles, the server first generates the optimal cut layer for each participating client to reduce the training latency. Based on the latency-optimal model splitting, the server and clients collaboratively optimize the global model under the SA-SFL scheme. The global learning objective follows the standard federated risk minimization formulation:

$$\mathcal{W}^* = \arg\min_{\mathcal{W}} \mathfrak{L}(\mathcal{W}) = \arg\min_{\mathcal{W}} \sum_{j \in \mathcal{J}} p_j \, \mathfrak{L}_{\mathcal{D}_j}(\mathcal{W}), \tag{1}$$

wherein, the global objective $\mathfrak{L}(\mathcal{W})$ is the weighted sum of local objectives $\{\mathfrak{L}_{\mathcal{D}_j}(\mathcal{W})\}_{j \in \mathcal{J}}$, with weights satisfying $\sum_{j \in \mathcal{J}} p_j = 1$, e.g., the fraction of local samples. Therefore, the overall objective is to optimize the global model $\mathcal{W}$ while minimizing the wall-clock training time, thereby improving the time-to-accuracy performance under heterogeneous settings.

2.1 STACKED-ACTIVATION SFL

In round $n \in [N]$, considering the communication bottleneck and client availability, the central server randomly selects a subset of clients $\mathcal{J}^n \subseteq \mathcal{J}$. The client-side submodels $\{\mathcal{W}_{c,j}\}_{j \in \mathcal{J}^n}$ and the server-side submodel $\mathcal{W}_s$ are updated separately for T local steps. SA-SFL approaches the optimal global $\mathcal{W}^*$ by periodically synchronizing the local submodels and concatenating the aggregated local submodel with the shared server-side submodel, iterating over the N communication rounds.

At step τ, all the participating clients bootstrap the local SGD in parallel by forward propagation on mini-batch samples $\mathcal{B}_j^\tau \subseteq \mathcal{D}_j$, $\forall j \in \mathcal{J}^n$. The j-th client then offloads activations $\mathbf{A}_j^{(n,\tau)} = \{f^{\mathcal{W}_{c,j}}(\mathbf{x})\}_{\mathbf{x} \in \mathcal{B}_j^\tau}$ to the server. After collecting activations from all clients $\mathcal{J}^n$, the server stacks them into a batch $\tilde{\mathbf{A}}^{(n,\tau)} = \{\mathbf{A}_j^{(n,\tau)}\}_{j \in \mathcal{J}^n}$ and performs one-step forward pass $\hat{\mathbf{Y}} = \{f^{\mathcal{W}_s}(\mathbf{a})\}_{\mathbf{a} \in \tilde{\mathbf{A}}^{(n,\tau)}}$, followed by gradient descent on the server-side submodel $\mathcal{W}_s^{(n,\tau)} = \mathcal{W}_s^{(n,\tau-1)} - \eta \nabla \mathfrak{L}_{\tilde{\mathbf{A}}^{(n,\tau)}}(\mathcal{W}_s^{(n,\tau-1)})$ with the learning rate η. Specifically, the stochastic gradient is

[1]Without loss of generality, we assume that the full model is split into two parts, rather than three parts that leave both the bottom and top submodels with the clients to avoid label sharing.

$\nabla \mathfrak{L}_{\tilde{\mathbf{A}}^{(n,\tau)}}(\mathcal{W}_s^{(n,\tau-1)}) = \frac{1}{|\tilde{\mathbf{A}}^{(n,\tau)}|} \sum_{\mathbf{a} \in \tilde{\mathbf{A}}^{(n,\tau)}} \nabla \ell(\mathbf{a}; \mathcal{W}_s^{(n,\tau-1)})$, where $\ell(\cdot)$ is the loss function, e.g., cross-entropy loss. The server sends the gradients on the cut layer back to the corresponding clients, which then update their local submodels $\mathcal{W}_{c,j}^{(n,\tau-1)} \mapsto \mathcal{W}_{c,j}^{(n,\tau)}$ following the chain rule.

At the end of each communication round n, the client-side submodels are synchronized via weighted averaging[2] $\bar{\mathcal{W}}_c^n = \sum_{j \in \mathcal{J}^n} p_j \mathcal{W}_{c,j}^{(n,|\tau|)}$. The global model is then formed as $\mathcal{W}^n = [\bar{\mathcal{W}}_c^n, \mathcal{W}_s^{(n,|\tau|)}]$.

Convergence Analysis. Following Han et al. (2024), we analyze SA-SFL with a fixed cut layer under non-convex objectives and partial client participation, where m clients are sampled uniformly.

Assumption 2.1 (S-smoothness)**.** *For every $j \in \mathcal{J}$, the local objective $\mathfrak{L}_{\mathcal{D}_j}$ is S-smooth ($S > 0$), i.e., $\|\nabla \mathfrak{L}_{\mathcal{D}_j}(\mathcal{W}) - \nabla \mathfrak{L}_{\mathcal{D}_j}(\mathcal{W}')\| \le S\|\mathcal{W} - \mathcal{W}'\|$ for all $\mathcal{W}, \mathcal{W}'$.*

Assumption 2.2 (Unbiased stochastic gradients with bounded variance)**.** *Mini-batches are sampled independently across clients and local steps. The stochastic gradient is unbiased with respect to the gradient of the entire local dataset, i.e., for a mini-batch $\mathcal{B} \subseteq \mathcal{D}_j$, $\mathbb{E}[\nabla \mathfrak{L}_{\mathcal{B}}(\mathcal{W})] = \nabla \mathfrak{L}_{\mathcal{D}_j}(\mathcal{W})$, and $\mathbb{E}\|\nabla_{\mathcal{W}_c} \mathfrak{L}_{\mathcal{B}}(\mathcal{W}) - \nabla_{\mathcal{W}_c} \mathfrak{L}_{\mathcal{D}_j}(\mathcal{W})\|^2 \le \sigma_c^2$ and $\mathbb{E}\|\nabla_{\mathcal{W}_s} \mathfrak{L}_{\mathcal{B}}(\mathcal{W}) - \nabla_{\mathcal{W}_s} \mathfrak{L}_{\mathcal{D}_j}(\mathcal{W})\|^2 \le \sigma_s^2$.*

Assumption 2.3 (Bounded heterogeneity)**.** *The divergence between local and global gradients is bounded, i.e., for all $\mathcal{W}$, $\sum_j p_j \|\nabla_{\mathcal{W}_c} \mathfrak{L}_{\mathcal{D}_j}(\mathcal{W}) - \nabla_{\mathcal{W}_c} \mathfrak{L}(\mathcal{W})\|^2 \le \epsilon_c^2$ and $\sum_j p_j \|\nabla_{\mathcal{W}_s} \mathfrak{L}_{\mathcal{D}_j}(\mathcal{W}) - \nabla_{\mathcal{W}_s} \mathfrak{L}(\mathcal{W})\|^2 \le \epsilon_s^2$.*

Theorem 2.4 (SA-SFL: non-convex, partial participation)**.** *Let Assumptions 2.1–2.3 hold. Define $\chi = \frac{|\mathcal{J}|-m}{|\mathcal{J}|-1}$, which is zero under full participation, and let $\eta_{\max} = \max\{\eta_c, \eta_s\} \le \frac{1}{8ST}$. Then, with $\mathfrak{L}^* = \inf_{\mathcal{W}} \mathfrak{L}(\mathcal{W})$,*

$$\begin{aligned} &\frac{1}{N} \sum_{n=0}^{N-1} \mathbb{E}\Big[\eta_c \big\|\nabla_{\mathcal{W}_c} \mathfrak{L}(\mathcal{W}^n)\big\|^2 + \eta_s \big\|\nabla_{\mathcal{W}_s} \mathfrak{L}(\mathcal{W}^n)\big\|^2\Big] \\ &\quad \le \frac{5\big(\mathfrak{L}(\mathcal{W}^0) - \mathfrak{L}^*\big)}{NT} + \frac{5S}{m}\big(\eta_c^2 \sigma_c^2 + \eta_s^2 \sigma_s^2\big) + \frac{10ST\chi}{m}\big(\eta_c^2 \epsilon_c^2 + \eta_s^2 \epsilon_s^2\big) \\ &\qquad + 4\eta_{\max} S^2 T\Big[\eta_c^2\big(3\sigma_c^2 + 2T\epsilon_c^2\big) + \frac{\eta_s^2}{m}\big(3\sigma_s^2 + 2T\chi\epsilon_s^2\big)\Big]. \end{aligned} \tag{2}$$

The proof and convergence rate are given in Appendix D, with comparisons to SFLV1 and SFLV2 in Appendix D.5. Under identical effective step sizes, SA-SFL avoids the additional server-side drift caused by independent server replicas in SFLV1 and order-dependent drift from sequential updates in SFLV2; under full participation, its server-side heterogeneity term vanishes. Moreover, moving the cut deeper shifts more layers on the client side, where heterogeneity is not reduced by stacked server averaging, consistent with empirical preference of SA-SFL for shallower cuts.

Dynamic Splitting. In communication round n, suppose that SA-SFL adopts dynamic edge-server task allocation across participating clients, where client j splits its local model at cut layer L_j^n. Under client-specific cut layers $\mathbf{L}^n = [L_j^n]_{j \in \mathcal{J}^n}$, the server receives smashed activations generated at different depths of the local models: $\tilde{\mathbf{A}}^{(n,\tau)} = \bigcup_{l \in \mathcal{L}^n} \hat{\mathbf{A}}_l^{(n,\tau)}$ where $\mathcal{L}^n = \{L_j^n \mid j \in \mathcal{J}^n\} \subseteq [|\mathcal{W}|]$ denotes the set of distinct cut layers and $\hat{\mathbf{A}}_l^{(n,\tau)} = \{\mathbf{A}_j^{(n,\tau)}\}_{j \in \mathcal{J}_l^n}$ is the activation group collected from clients in $\mathcal{J}_l^n = \{j \in \mathcal{J}^n \mid L_j^n = l\}$ satisfying $\mathcal{J}^n = \bigcup_{l \in \mathcal{L}^n} \mathcal{J}_l^n$.

As shown in Figure 2, PyramidSFL retains a single server-side submodel instead of maintaining $|\mathcal{L}^n|$ separate server-side submodels for different activation groups $\hat{\mathbf{A}}_{l \in \mathcal{L}^n}^{(n,\tau)}$. The reason is twofold: 1) the memory cost for submodels and optimizer-state replicas would scale with $|\mathcal{L}^n|$, and 2) synchronizing multiple server-side submodels would reduce batching efficiency and make training sensitive to the aggregation frequency across submodels. These negative effects push the training closer to the SFLV1 scheme while forfeiting the benefits from SA-SFL, that is, updating of a shared submodel on stacked activations. In this way, the server-side training in PyramidSFL becomes layer-dependent under dynamic splitting. Specifically, the k-th layer is updated by the activation batch $\dot{\mathbf{A}}_k^{(n,\tau)} = \bigcup_{l \in \mathcal{L}^n : l < k} \hat{\mathbf{A}}_l^{(n,\tau)} \subseteq \tilde{\mathbf{A}}^{(n,\tau)}$. As a result, the effective activation subset $\dot{\mathbf{A}}_k^{(n,\tau)}$ increases with the depth of the server-side layer k: deeper layers are updated by a larger fraction of $\tilde{\mathbf{A}}^{(n,\tau)}$, and only the top few server-side layers satisfying $k \ge \max(\mathcal{L}^n)$ are updated using the entire stacked batch.

[2]Like SFLV1/SFLV2, SA-SFL also relies on a *fed server* to aggregate the client-side submodels. For simplicity, we let the central server perform the operations of the fed server.

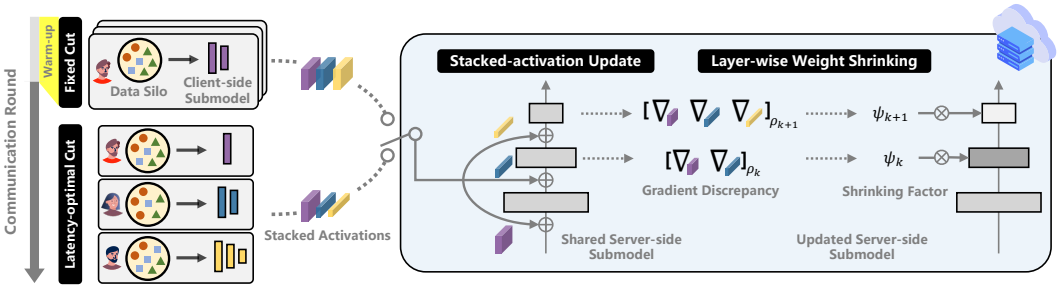

Figure 2: PyramidSFL overview: it bootstraps clients to perform fixed-cut warm-up and then switch to latency-optimal dynamic splitting according to their heterogeneous resource profiles. The offloaded activations from clients are grouped by cut layer and jointly used to update a shared server-side submodel. PyramidSFL further estimates cross-layer gradient discrepancy and applies layer-wise weight shrinking to mitigate the optimization inconsistency introduced by dynamic splitting.

2.2 Resources-adaptive Splitting

Since server-side updates in SA-SFL are triggered by the collection of activations from all participating clients $\mathcal{J}^n$, the synchronization of client-side forward passes becomes the primary bottleneck. Given the device-side computation coefficient o_j^d (*s/frame*) and bandwidth b_j (*Mbit/s*), the forward-pass latency of client j at layer L can be written as

$$t_j^{\mathrm{fw}}(L) = t_j^{\mathrm{fw-cmp}}(L) + t_j^{\mathrm{fw-com}}(L) = B o_j^d \frac{\sum_{l=1}^{L} c_l}{\|\mathbf{c}\|_1} + B\frac{a_L}{b_j}, \tag{3}$$

including the forward-propagation latency $t_j^{\mathrm{fw-cmp}}$ and transmission latency $t_j^{\mathrm{fw-com}}$. Here, $B = |\mathcal{B}_j^\tau|$ is the mini-batch size. We profile the layer-wise computational complexity (in *MACs*) and output size (in *bits*) of the task model $\mathcal{W}$ by vectors $\mathbf{c} \in \mathbb{R}_+^{|\mathcal{W}|}$ and $\mathbf{a} \in \mathbb{R}_+^{|\mathcal{W}|}$, where c_l and a_l denote the computation cost and activation size at the l-th layer, respectively.

Our objective is to reduce the synchronization bottleneck caused by straggling clients by searching for client-specific cut layers $\mathbf{L}^n$. This can be formulated as:

$$\min_{\mathbf{L}^n} \max_{j \in \mathcal{J}^n} t_j^{\mathrm{fw}}(L_j^n), \text{ s.t. } L_j^n \in \tilde{\mathcal{L}}, \forall j \in \mathcal{J}^n, \tag{4}$$

where $\tilde{\mathcal{L}}$ is the predefined search scope for the cut layer, e.g., the set of feasible split positions in the full task model, $\tilde{\mathcal{L}} = [|\mathcal{W}|]$. For a given cut layer L, the latency $t_j(L)$ depends only on the local resource profile of client j, and the constraints are loose across clients. Therefore, this finite-domain discrete min-max optimization problem decomposes naturally into a set of independent per-client subproblems: $L_j^{n*} = \arg\min_{L \in \tilde{\mathcal{L}}} t_j^{\mathrm{fw}}(L)$, which can be solved by exhaustive search over $\tilde{\mathcal{L}}$. However, the search cost scales linearly with the size of the search space $\tilde{\mathcal{L}}$, which can become non-negligible for deep models. To address this, we exploit a simple structural property of modern neural networks: consecutive layers or blocks may output activations of the same size. For instance, in ResNet (He et al., 2016), consecutive residual blocks within the same stage introduce identical communication latency, $t_j^{\mathrm{fw-com}}(L) = t_j^{\mathrm{fw-com}}(L+1)$, whereas the shallower cut provides lower client-side computation latency, $t_j^{\mathrm{fw-cmp}}(L) < t_j^{\mathrm{fw-cmp}}(L+1)$, by offloading a larger fraction of the model training workload to a more powerful server. Based on this observation, it reduces the search cost by replacing the original search scope $\tilde{\mathcal{L}}$ with a pruned space $\tilde{\mathcal{L}}' \subseteq \tilde{\mathcal{L}}$ while preserving the latency-optimal solution.

2.3 Dynamic-splitting Training

In the n-th round, PyramidSFL bootstraps collaborative training by latency-aware task allocation, where clients follow the dynamic splitting $\mathcal{L}^n = \mathcal{L}^* = \{L_j^{n*} \mid j \in \mathcal{J}^n\}$. Assuming that local resource profiles remain unchanged within one communication round, clients follow $\mathcal{L}^n$ across T training steps. The server therefore updates a server-side submodel in depth $|\mathcal{W}_s^n| = |\mathcal{W}| - \min(\mathcal{L}^n)$.

Server-side training under dynamic splitting is non-trivial from an optimization perspective. In SA-SFL, updating the server-side submodel on stacked activation $\tilde{\mathbf{A}}^{(n,\tau)}$ from multiple clients mitigates the training inconsistency with the global objective, since $\tilde{\mathbf{A}}^{(n,\tau)}$ forms a mixture of local

silo distributions and may approximate the global distribution. Dynamic splitting, however, introduces layer-wise update skew to the server-side submodel: the k-th layer is updated on an activation subset $\dot{\mathbf{A}}_k^{(n,\tau)} \subseteq \tilde{\mathbf{A}}^{(n,\tau)}$, and both the distribution and sample size of this subset can vary across layers. As a result, latency-optimal splitting incurs an additional structural inconsistency beyond the inherent data heterogeneity in the stacked activation batch. To address this, SA-SFL adopts two training strategies to improve the layer-wise update consistency under dynamic splitting.

Warm-up Strategy. It is motivated by two factors: 1) SA-SFL generally yields better training performance under shallower splitting, where a larger fraction of the model is allocated to the server; 2) such shallow splitting, however, incurs higher communication overhead, since bottom-layer activations and gradients are typically larger in size. In line with observations, PyramidSFL introduces a warm-up phase at the beginning of N^{wp} communication rounds. During warm-up, all participating clients are forced to adopt the same shallow cut layer $L^{\mathrm{wp}} \in [|\mathcal{W}|]$, i.e., the dynamic pattern is temporarily fixed to a static split with $\mathcal{L}^n = \{L^{\mathrm{wp}}\}, \forall n \in [N^{\mathrm{wp}}]$. Besides, PyramidSFL prioritizes clients with high communication bandwidth during the warm-up phase. To this end, each client $j \in \mathcal{J}$ is assigned a bandwidth-dependent weight $w_j = (b_j + \epsilon)^\alpha \in \mathbb{R}_+$, where ϵ is a small positive constant and α controls the extent of bandwidth prioritization. The server then samples participating clients $\mathcal{J}^n$ with probability $\mathrm{Prob}_j \propto w_j$, e.g., by normalizing the weights over all clients. After warm-up, the sampling policy reverts to uniform sampling with $\mathrm{Prob}_j = \frac{1}{|\mathcal{J}|}, \forall j \in \mathcal{J}$.

The choice of α introduces a trade-off between optimization stability and training latency under fixed model splitting. A larger α reduces the additional communication latency arising from the static split at L^{wp}, but biases client selection toward a small set of high-bandwidth clients. As a result, the server-side submodel may suffer from biased early optimization, since $\tilde{\mathbf{A}}^{(n,\tau)}$ is confined to a narrower mixture of client distributions over the N^{wp} warm-up rounds. In contrast, a smaller α improves sampling diversity at the cost of higher communication latency during warm-up. Therefore, N^{wp} and α jointly control the performance gain from the warm-up strategy at the early stage.

Weight-shrinking Strategy. After the warm-up stage, PyramidSFL switches to latency-optimal splitting, where the dynamic cut layers $\mathcal{L}^n$ reduce the training latency yet introduce layer-wise update inconsistency to the server-side submodel. Specifically, the stochastic gradient of the k-th server-side layer is calculated on stacked activation batch $\dot{\mathbf{A}}_k^{(n,\tau)} = \bigcup_{l \in \dot{\mathcal{L}}_k^n} \hat{\mathbf{A}}_l^{(n,\tau)}$, which is collected from clients split shallower than layer k, i.e., $\dot{\mathcal{L}}_k^n = \{l \in \mathcal{L}^n \mid l < k\}$. As a result, server-side layers are optimized with varying mixtures and sizes of stacked activations. It therefore necessitates imposing conservative constraints on server-side updates to regulate cross-split optimization bias.

To this end, PyramidSFL applies layer-wise weight shrinking to the server-side submodel. Unlike post-aggregation global model shrinking in FL (Li et al., 2023; Shi et al., 2025a), PyramidSFL tailors it to SA-SFL with dynamic splitting from three aspects: 1) weight shrinking is applied after each training step as PyramidSFL is free of periodic server-side aggregation; 2) the layer-specific shrinking factor focuses on the gradients discrepancy across activation groups $\{\hat{\mathbf{A}}_l^{(n,\tau)}\}_{l \in \dot{\mathcal{L}}_k^n}$, where each group $\hat{\mathbf{A}}_l^{(n,\tau)}$ forms a mixture and heterogeneous distribution of a client subset $\mathcal{J}_l^n$; and 3) the activation size $|\dot{\mathbf{A}}_k^{(n,\tau)}|$ varies across server-side layers, further exacerbating the layer-wise gradient discrepancy.

In line with this, denote $\tilde{\mathbf{g}}_{lk}^{(n,\tau)}$ as the contribution of batched activation $\hat{\mathbf{A}}_l^{(n,\tau)}$ to the update of the k-th server-side layer. For activation group traversing layer k, it contributes the stochastic gradient $\mathbf{g}_{lk}^{(n,\tau)} = \nabla \mathfrak{L}_{\hat{\mathbf{A}}_l^{(n,\tau)}}(\mathcal{W}_{sk}^{(n,\tau-1)})$; otherwise, it contributes no update to layer k on the server side:

$$\tilde{\mathbf{g}}_{lk}^{(n,\tau)} = \begin{cases} \mathbf{g}_{lk}^{(n,\tau)}, & l \in \dot{\mathcal{L}}_k^n, \\ \mathbf{0}, & l \in \mathcal{L}^n \setminus \dot{\mathcal{L}}_k^n. \end{cases} \tag{5}$$

The gradient discrepancy at the k-th layer is

$$\rho_k^{(n,\tau)} = \frac{\sum_{l \in \mathcal{L}^n} |\hat{\mathbf{A}}_l^{(n,\tau)}| \left\| \tilde{\mathbf{g}}_{lk}^{(n,\tau)} - \bar{\mathbf{g}}_k^{(n,\tau)} \right\|_2}{\sum_{l \in \mathcal{L}^n} |\hat{\mathbf{A}}_l^{(n,\tau)}|}, \tag{6}$$

where the mean gradient $\bar{\mathbf{g}}_k^{(n,\tau)}$ is a weighted average by the activation group size, that is, $\bar{\mathbf{g}}_k^{(n,\tau)} = \frac{\sum_{l \in \mathcal{L}^n} |\hat{\mathbf{A}}_l^{(n,\tau)}| \tilde{\mathbf{g}}_{lk}^{(n,\tau)}}{\sum_{l \in \mathcal{L}^n} |\hat{\mathbf{A}}_l^{(n,\tau)}|}$, satisfying $|\tilde{\mathbf{A}}^{(n,\tau)}| = \sum_{l \in \mathcal{L}^n} |\hat{\mathbf{A}}_l^{(n,\tau)}|$.

Table 1: Performance comparison between PyramidSFL and baselines across three benchmarks under the default $\mathrm{Dir}_{100}(0.2)$ data distribution. We report the average and standard deviation of *Top-1 accuracy* under a fixed wall-clock budget, the wall-clock time (**T**) required to reach the target accuracy, and the corresponding time reduction (↓) or excess (↑) relative to FL. All results are averaged over three trials, with **bold** indicating the best performance for each setup.

Method	L	CIFAR-10/DIR$_{100}$(0.2)		CIFAR-100/DIR$_{100}$(0.2)		Tiny-ImageNet/DIR$_{100}$(0.2)	
		Acc.@150h(%)	T@70%(h)	Acc.@150h(%)	T@40%(h)	Acc.@300h(%)	T@30%(h)
FL	w/o	72.74±0.31	127.85	48.55±0.22	64.29	21.62±0.51	486.96
SFLV1	L=2	69.42±1.16	166.58↑30.29%	44.39±0.51	65.70↑2.19%	30.80±0.79	287.56↓40.95%
SFLV1*	$\mathcal{L}^*$	72.81±0.29	122.89↓3.88%	48.61±0.18	61.70↓4.03%	31.49±0.58	265.25↓45.53%
SFLV2	L=2	72.05±0.86	113.39↓11.31%	46.45±0.23	57.81↓10.08%	19.05±0.54	–
SA-SFL	L=2	73.90±0.67	93.55↓26.83%	49.72±0.12	68.19↑6.07%	31.53±0.42	239.32↓50.85%
SA-SFL*	$\mathcal{L}^*$	73.23±0.20	110.02↓13.95%	47.01±0.26	50.97↓20.72%	35.98±0.96	181.62↓62.70%
MergeSFL	L=2	74.69±0.57	82.14↓35.75%	50.85±0.07	59.60↓7.30%	32.21±0.05	261.76↓46.25%
ParallelSFL	L=2	72.99±0.58	136.32↑6.62%	49.44±0.07	65.09↑1.24%	30.63±0.40	275.48↓43.43%
SMoFi	L=2	72.53±0.11	79.66↓37.69%	46.45±0.42	58.23↓9.43%	28.84±0.84	341.09↓29.96%
PyramidSFL	$\mathcal{L}^*$	**77.27**±0.53	**37.00**↓71.06%	**55.56**±0.55	**38.78**↓39.68%	**37.15**±0.59	**132.16**↓72.86%

After training step τ, the server first updates the submodel $\mathcal{W}_s^{(n,\tau-1)} \mapsto \mathcal{W}_s^{(n,\tau)}$. PyramidSFL then applies weight shrinking to each layer by $\mathcal{W}_{sk}^{(n,\tau)} \mapsto \psi_k^{(n,\tau)} \cdot \mathcal{W}_{sk}^{(n,\tau)}$ and $\mathcal{W}_{sk}^{(n,\tau)} \subseteq \mathcal{W}_s^{(n,\tau)}$. Specifically, the shrinking factor $\psi_k^{(n,\tau)}$ for the layer k is defined as

$$\psi_k^{(n,\tau)} = \frac{\left\|\mathcal{W}_{sk}^{(n,\tau-1)}\right\|_2}{\rho_k^{(n,\tau)}\beta\left\|\Delta\mathcal{W}_{sk}^{(n,\tau)}\right\|_2 + \left\|\mathcal{W}_{sk}^{(n,\tau-1)}\right\|_2}, \tag{7}$$

where β scales the shrinking strength and $\Delta\mathcal{W}_{sk}^{(n,\tau)} = \mathcal{W}_{sk}^{(n,\tau)} - \mathcal{W}_{sk}^{(n,\tau-1)}$ is the step-wise parameter change before shrinking. In this way, PyramidSFL assigns a smaller $\psi \in (0, 1]$ to the layer with larger gradient discrepancy (measured by ρ), thereby imposing a more conservative server-side update.

3 Experimental Results

Datasets and Models. We conduct experiments on three widely used image benchmarks: CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009), and Tiny-ImageNet (Le & Yang, 2015). We use ResNet-18 (He et al., 2016) for CIFAR-10 and CIFAR-100, and ResNet-34 for Tiny-ImageNet. For each model, the candidate cut set $\tilde{\mathcal{L}}$ includes residual blocks starting from the second residual block, i.e., $L \geq 2$. This includes the default fixed cut $L = 2$ used by the compared SFL baselines in the search space to ensure a fair comparison. Detailed hyperparameters are provided in the Appendix.

Heterogeneous Clients Setup. We simulate *data heterogeneity* in line with previous work (Hsu et al., 2019; Liao et al., 2024b; Wu et al., 2026), where data silo $\mathcal{D}_j$ follows distribution $\boldsymbol{q_j} \in \mathbb{R}^{\mathbf{c}}$ ($\mathbf{c}$ is the number of classes). We sample $\boldsymbol{q_j}$ from a Dirichlet distribution $\mathrm{Dir}_{|\mathcal{J}|}(\gamma)$ with a concentration parameter $\gamma > 0$ controlling the level of heterogeneity. A smaller γ indicates a more heterogeneous distribution setting. Specifically, we set $\mathrm{Dir}_{100}(0.2)$ as default across three benchmarks and the server randomly selects 20% of clients $\mathcal{J}^n \subseteq \mathcal{J}$ in CIFAR-10, CIFAR-100 and 10% in Tiny-ImageNet for participation. Each selected client performs $|\tau| = 20$ local training steps for CIFAR-10 and CIFAR-100, and $|\tau| = 70$ for Tiny-ImageNet. To model system heterogeneity, we sample client-side computation profiles p_j^d and network bandwidth b_j from AI Benchmark (Ignatov et al., 2019) and MobiPerf (Huang et al., 2011), respectively. We also visualize the system heterogeneity in the Appendix.

Baselines. We compare PyramidSFL against FL and representative SFL methods. We consider three basic SFL frameworks: **SFLV1**, **SFLV2** (Thapa et al., 2022), and **SA-SFL**. For SFLV1, we evaluate server-side aggregation every $\bar{\tau}$ local steps, using $\bar{\tau} = 5$ for CIFAR-10 and CIFAR-10, and $\bar{\tau} = 10$ for Tiny-ImageNet. We also include round-wise aggregation with $\bar{\tau} = |\tau|$, which resembles the FedAvg-style aggregation. In SFLV1 ($\bar{\tau} = |\tau|$), we further replace the fixed cut with latency-optimal splitting $\mathcal{L}^*$, denoted as **SFLV1***. Similarly, **SA-SFL*** extends SA-SFL to $\mathcal{L}^*$ by maintaining $|\mathcal{L}^*|$ surrogate server-side submodels for each active cut and updating them independently.

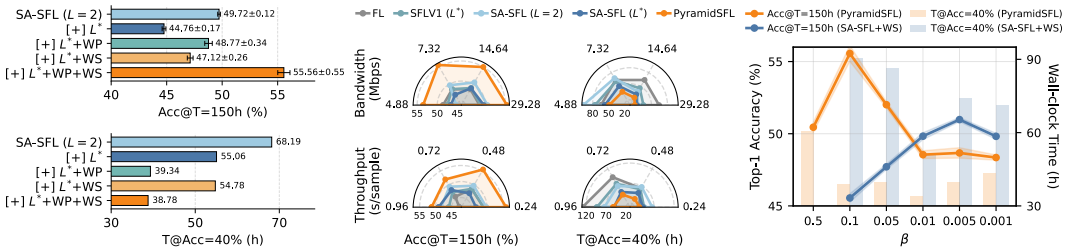

Figure 3: Ablation study and sensitivity analysis on CIFAR-100 with ResNet-18. **Left:** $\mathcal{L}^*$ reduces training latency but compromises accuracy; Warm-up (WP) and weight-shrinking (WS) jointly recover optimization quality while preserving the latency gain. **Middle:** Accuracy and time-to-accuracy under varying client bandwidth and computation throughput, illustrating the system-efficiency boundary between FL and SFL. **Right:** Sensitivity to the scaling factor β for PyramidSFL and SA-SFL with weight shrinking under the same grid-search space.

We additionally compare against three SFL methods targeting training efficiency or model performance: **MergeSFL** (Liao et al., 2024a), which adapts client batch sizes to resource profiles; **ParallelSFL** (Liao et al., 2024b), which clusters clients to mitigate stragglers; and **SMoFi** (Yang et al., 2026), which aligns momentum states across server-side optimizers under SFLV1.

Metrics. We evaluate both model quality and wall-clock convergence. For model quality, we report the mean *Top-1 accuracy* under a fixed wall-clock budget: $T = 150$ h for CIFAR-10/100 and $T = 300$ h for Tiny-ImageNet. For convergence efficiency, we report *time-to-accuracy*, i.e., the wall-clock time required to reach 70%, 40%, and 30% Top-1 accuracy on CIFAR-10, CIFAR-100, and Tiny-ImageNet, respectively. All experimental results are averaged over multiple runs. For consistency across methods, global aggregation latency is excluded from the reported training time.

3.1 Performance Evaluation

PyramidSFL Effectiveness. Table 1 compares PyramidSFL with FL and SFL baselines across the three benchmarks. Compared with FL, SFLV1* achieves comparable accuracy on CIFAR-10 and CIFAR-100 under the same wall-clock budget while reducing time-to-accuracy. This latency benefit becomes more significant on Tiny-ImageNet with the deeper ResNet-34, suggesting that dynamic splitting primarily improves system efficiency rather than learning quality. A similar pattern appears in SA-SFL: replacing the fixed cut with $\mathcal{L}^*$ does not necessarily improve time-to-accuracy. SA-SFL* maintains $|\mathcal{L}^*|$ cut-specific surrogate server-side submodels and updates them independently, weakening the update consistency that makes stacked-batch training effective.

PyramidSFL addresses this limitation by retaining a shared server-side model while allowing clients to adopt their latency-optimal cuts. To mitigate the resulting update inconsistency across server-side layers, PyramidSFL employs two mechanisms: 1) a warm-up phase with a fixed cut $L^{\text{wp}} = 2$ for the first $N^{\text{wp}} = 25$ communication rounds; and 2) weight shrinking during the subsequent rounds with dynamic splitting $\mathcal{L}^*$. We set the shrinking factor to $\beta = 0.1$ for CIFAR-10 and CIFAR-100, and $\beta = 0.05$ for Tiny-ImageNet. The results show that PyramidSFL consistently outperforms the baselines in both fixed-budget accuracy and time-to-accuracy across all three benchmarks.

Ablation Study. To validate the key components of SA-SFL, Figure 3(Left) reports an ablation study of latency-optimal splitting ($\mathcal{L}^*$), warm-up (WP), and weight-shrinking (WS) on CIFAR-100. Introducing dynamic model splitting alone reduces the time to reach the target accuracy, but noticeably degrades global model accuracy due to cross-layer update inconsistency in the shared server-side submodel. WP largely mitigates this issue by stabilizing server-side optimization during the initial communication rounds, improving both accuracy and convergence. WS provides a complementary benefit by reducing the optimization divergence introduced by dynamic splitting while largely preserving its latency advantage. By combining WP and WS, PyramidSFL offers better model accuracy and training latency for SA-SFL under dynamic model splitting.

Sensitivity Study. We further investigate the performance gain of PyramidSFL under varying communication and computation capabilities of clients, as shown in Figure 3(Middle). We observe that the SFL favors systems with richer communication resources (i.e., larger bandwidth) and more

constrained client-side computation resources (i.e., larger inference latency), where the benefit of computation offloading outweighs the communication overhead of transmitting intermediate activations/gradients. Latency-optimal splitting expands this sweet spot of SFL, but cannot eliminate the fundamental system-efficiency boundary between SFL and FL. In contrast, PyramidSFL consistently maintains higher model accuracy while substantially reducing convergence latency over existing SFL variants across all heterogeneous system settings.

We also report the sensitivity of PyramidSFL to the scaling factor $\beta \in [0.001, 0.5]$ in Figure 3(Right). The parameter β controls the strength of weight shrinking, with smaller values weakening the layer-wise shrinking factor ψ applied to each layer. We observe a clear trade-off between model accuracy and convergence when β varies. When β becomes sufficiently small ($\beta \leq 0.01$), the effect of weight shrinking gradually diminishes, and performance falls back to PyramidSFL without weight shrinking. For a more extensive evaluation, we also incorporate fixed-split SA-SFL with weight shrinking in line with the work (Shi et al., 2025a). At each training step, layer-wise shrinking factors are obtained from the gradient discrepancy across client activations and applied to the server-side submodel. Although fixed-split SA-SFL benefits from the weight-shrinking mechanism as well, PyramidSFL substantially outperforms it when both methods tune β over the same search space.

4 Related Work

Split Federated Learning. As a collaborative learning paradigm, Split Learning (Gupta & Raskar, 2018; Vepakomma et al., 2018a) is proposed to offload part of the on-device training workload to the server without exposing raw data. SplitFed (Thapa et al., 2022) further combines split learning with federated aggregation, enabling parallel client-side training. Following this scheme, works (Li & Lyu, 2023; Han et al., 2024) provide theoretical convergence analyses on SFL, especially the SFLV1 and SFLV2, while concurrent works (Pasquini et al., 2021; Li et al., 2024; Wang et al., 2025) discuss its privacy and security implications under model splitting and adversarial attacks. Compared with FL, SFL is particularly favorable for deep, computation-intensive task models deployed across resource-constrained clients at large scale (Singh et al., 2019). This motivates recent works to extend SFL to distributed Large Language Model (LLM) fine-tuning tasks (Lin et al., 2024a; Li et al., 2026), highlighting its great potential for scalable training of versatile foundation models.

Heterogeneity-resilient Training. Similar to FL, SFL also faces data heterogeneity issues, where existing data-driven methods (Chen et al., 2023; Zhu et al., 2024; 2026) and model-driven methods (Li et al., 2020; Wang et al., 2020a; Shi et al., 2025a;b) can often be integrated into SFL frameworks. System heterogeneity is more specific to SFL, with a particular focus on computation latency and communication bottlenecks. Hierarchical SFL frameworks alleviate straggler effects through worker clustering (Liao et al., 2024b) or multi-tier collaboration (Khan et al., 2023; Lin et al., 2025b). Besides, works reduce smashed-data communication via local-loss learning (Han et al., 2021; Oh et al., 2022), and lower synchronization overhead through adaptive model aggregation (Lin et al., 2024b; 2025a). Resource-aware workload allocation has also been explored through dynamic batch sizes (Liao et al., 2023; 2024a), bandwidth-aware communication topologies (Tang et al., 2022), and adaptive model splitting (Kim et al., 2023; Wu et al., 2023; Wen et al., 2025). As a resource-aware dynamic splitting framework tailored to SA-SFL, PyramidSFL is complementary to these system-level optimizations and can incorporate additional constraints such as energy budgets and wireless channel conditions.

5 Conclusion

In this paper, we present PyramidSFL, a system-efficient training framework tailored to SA-SFL with latency-optimal dynamic model splitting. By adapting the cut layer to heterogeneous client resource profiles, PyramidSFL reduces collaborative training latency, while its warm-up and weight-shrinking mechanisms mitigate the optimization inconsistency introduced by dynamic splitting. As a result, PyramidSFL improves both time-to-accuracy and global model performance over existing FL and SFL baselines across heterogeneous system conditions. Importantly, PyramidSFL is client-transparent and requires no modification to the standard client-side SFL training pipeline, introducing no additional client-side system overhead or privacy risk. This also allows PyramidSFL to be readily integrated with other federated training mechanisms as a plug-in approach.

AI USE STATEMENT

Large language models (LLMs) were used as assistive tools during the development of this work. Specifically, we used generative AI tools to assist with code implementation and debugging, checking mathematical derivations, discussing experimental design, preparing figures, and refining the language and presentation of the manuscript.

All research questions, methodological choices, experimental protocols, analyses, and conclusions were determined and validated by the authors. AI-assisted code, derivations, experimental suggestions, figures, and manuscript text were reviewed and, where necessary, revised by the authors before being incorporated into the work. The authors take full responsibility for the scientific content, experimental results, and claims presented in this paper.

REPRODUCIBILITY STATEMENT

Section 2 describes the SFL setting, optimization objective, and the proposed dynamic splitting and server-side training strategies. The experimental section reports datasets, model architectures, baselines, resource heterogeneity settings, and training configurations. The appendix provides the assumptions and proofs underlying the convergence analysis, together with additional experimental details.

REFERENCES

Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1: 374–388, 2019.

Haokun Chen, Ahmed Frikha, Denis Krompass, Jindong Gu, and Volker Tresp. Fraug: Tackling federated learning with non-iid features via representation augmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4849–4859, 2023.

Amir Erfan Eshratifar, Mohammad Saeed Abrishami, and Massoud Pedram. Jointdnn: An efficient training and inference engine for intelligent mobile cloud computing services. *IEEE Transactions on Mobile Computing*, 20(2):565–576, 2019.

Giancarlo Fortino and Paolo Trunfio. *Internet of things based on smart objects: Technology, middleware and applications*. Springer, 2014.

Yansong Gao, Minki Kim, Sharif Abuadbba, Yeonjae Kim, Chandra Thapa, Kyuyeon Kim, Seyit A Camtepe, Hyoungshick Kim, and Surya Nepal. End-to-end evaluation of federated learning and split learning for internet of things. *arXiv preprint arXiv:2003.13376*, 2020.

Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660, 2013.

Otkrist Gupta and Ramesh Raskar. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, 116:1–8, 2018.

Dong-Jun Han, Hasnain Irshad Bhatti, Jungmoon Lee, and Jaekyun Moon. Accelerating federated learning with split learning on locally generated losses. In *ICML 2021 workshop on federated learning for user privacy and data confidentiality. ICML Board*, volume 2, 2021.

Pengchao Han, Chao Huang, Geng Tian, Ming Tang, and Xin Liu. Convergence analysis of split federated learning on heterogeneous data. *Advances in Neural Information Processing Systems*, 37:103476–103544, 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

Junxian Huang, Cheng Chen, Yutong Pei, Zhaoguang Wang, Zhiyun Qian, Feng Qian, Birjodh Tiwana, Qiang Xu, Z Mao, Ming Zhang, et al. Mobiperf: Mobile network measurement system. *Technical Report. University of Michigan and Microsoft Research*, 2011.

Andrey Ignatov, Radu Timofte, Andrei Kulik, Seungsoo Yang, Ke Wang, Felix Baum, Max Wu, Lirong Xu, and Luc Van Gool. Ai benchmark: All about deep learning on smartphones in 2019. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pp. 3617–3635. IEEE, 2019.

Latif U Khan, Mohsen Guizani, Ala Al-Fuqaha, Choong Seon Hong, Dusit Niyato, and Zhu Han. A joint communication and learning framework for hierarchical split federated learning. *IEEE Internet of Things Journal*, 11(1):268–282, 2023.

Minsu Kim, Alexander DeRieux, and Walid Saad. A bargaining game for personalized, energy efficient split learning over wireless networks. In *2023 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6. IEEE, 2023.

Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Fan Lai, Yinwei Dai, Sanjay Singapuram, Jiachen Liu, Xiangfeng Zhu, Harsha Madhyastha, and Mosharaf Chowdhury. Fedscale: Benchmarking model and system performance of federated learning at scale. In *International Conference on Machine Learning*, pp. 11814–11827. PMLR, 2022.

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Ji Li, Hui Gao, Tiejun Lv, and Yueming Lu. Deep reinforcement learning based computation offloading and resource allocation for mec. In *2018 IEEE wireless communications and networking conference (WCNC)*, pp. 1–6. IEEE, 2018.

Jingtao Li, Xing Chen, Li Yang, Adnan Siraj Rakin, Deliang Fan, and Chaitali Chakrabarti. Emgan: Early-mix-gan on extracting server-side model in split federated learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 13545–13553, 2024.

Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th international conference on data engineering (ICDE)*, pp. 965–978. IEEE, 2022.

Tao Li, Yulin Tang, Yiyang Song, Cong Wu, Xihui Liu, Pan Li, and Xianhao Chen. Splitcom: Communication-efficient split federated fine-tuning of llms via temporal compression. *arXiv preprint arXiv:2602.10564*, 2026.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

Yipeng Li and Xinchen Lyu. Convergence analysis of sequential federated learning on heterogeneous data. *Advances in Neural Information Processing Systems*, 36:56700–56755, 2023.

Zexi Li, Tao Lin, Xinyi Shang, and Chao Wu. Revisiting weighted aggregation in federated learning with neural networks. In *International Conference on Machine Learning*, pp. 19767–19788. PMLR, 2023.

Yunming Liao, Yang Xu, Hongli Xu, Zhiwei Yao, Lun Wang, and Chunming Qiao. Accelerating federated learning with data and model parallelism in edge computing. *IEEE/ACM Transactions on Networking*, 32(1):904–918, 2023.

Yunming Liao, Yang Xu, Hongli Xu, Lun Wang, Zhiwei Yao, and Chunming Qiao. Mergesfl: Split federated learning with feature merging and batch size regulation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pp. 2054–2067. IEEE, 2024a.

Yunming Liao, Yang Xu, Hongli Xu, Zhiwei Yao, Liusheng Huang, and Chunming Qiao. Parallelsfl: A novel split federated learning framework tackling heterogeneity issues. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, pp. 845–860, 2024b.

Zheng Lin, Xuanjie Hu, Yuxin Zhang, Zhe Chen, Zihan Fang, Xianhao Chen, Ang Li, Praneeth Vepakomma, and Yue Gao. Splitlora: A split parameter-efficient fine-tuning framework for large language models. *arXiv preprint arXiv:2407.00952*, 2024a.

Zheng Lin, Guangyu Zhu, Yiqin Deng, Xianhao Chen, Yue Gao, Kaibin Huang, and Yuguang Fang. Efficient parallel split learning over resource-constrained wireless edge networks. *IEEE Transactions on Mobile Computing*, 23(10):9224–9239, 2024b.

Zheng Lin, Guanqiao Qu, Wei Wei, Xianhao Chen, and Kin K Leung. Adaptsfl: Adaptive split federated learning in resource-constrained edge networks. *IEEE Transactions on Networking*, 2025a.

Zheng Lin, Wei Wei, Zhe Chen, Chan-Tong Lam, Xianhao Chen, Yue Gao, and Jun Luo. Hierarchical split federated learning: Convergence analysis and system optimization. *IEEE Transactions on Mobile Computing*, 2025b.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 2017.

Seungeun Oh, Jihong Park, Praneeth Vepakomma, Sihun Baek, Ramesh Raskar, Mehdi Bennis, and Seong-Lyun Kim. Locfedmix-sl: Localize, federate, and mix for improved scalability, convergence, and latency in split learning. In *Proceedings of the ACM Web Conference 2022*, pp. 3347–3357, 2022.

Dario Pasquini, Giuseppe Ateniese, and Massimo Bernaschi. Unleashing the tiger: Inference attacks on split learning. In *Proceedings of the 2021 ACM SIGSAC conference on computer and communications security*, pp. 2113–2129, 2021.

Changlong Shi, Jinmeng Li, He Zhao, Dandan Guo, and Yi Chang. Fedlws: Federated learning with adaptive layer-wise weight shrinking. 2025a.

Changlong Shi, He Zhao, Bingjie Zhang, Mingyuan Zhou, Dandan Guo, and Yi Chang. Fedawa: adaptive optimization of aggregation weights in federated learning using client vectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 30651–30660, 2025b.

Abhishek Singh, Praneeth Vepakomma, Otkrist Gupta, and Ramesh Raskar. Detailed comparison of communication efficiency of split learning and federated learning. *arXiv preprint arXiv:1909.09145*, 2019.

Zhenheng Tang, Shaohuai Shi, Bo Li, and Xiaowen Chu. Gossipfl: A decentralized federated learning framework with sparsified and adaptive communication. *IEEE Transactions on Parallel and Distributed Systems*, 34(3):909–922, 2022.

Chandra Thapa, Pathum Chamikara Mahawaga Arachchige, Seyit Camtepe, and Lichao Sun. Splitfed: When federated learning meets split learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8485–8493, 2022.

Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018a.

Praneeth Vepakomma, Tristan Swedish, Ramesh Raskar, Otkrist Gupta, and Abhimanyu Dubey. No peek: A survey of private distributed deep learning. *arXiv preprint arXiv:1812.03288*, 2018b.

Joost Verbraeken, Matthijs Wolting, Jonathan Katzy, Jeroen Kloppenburg, Tim Verbelen, and Jan S Rellermeyer. A survey on distributed machine learning. *Acm computing surveys (csur)*, 53(2): 1–33, 2020.

Fei Wang, Yan Zhu, and Baochun Li. Unraveling elevated data leakage in split learning for fine-tuning stable diffusion models. In *Proceedings of the 20th ACM Asia Conference on Computer and Communications Security*, pp. 501–516, 2025.

Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020a.

Song Wang, Xinyu Zhang, Hiromasa Uchiyama, and Hiroki Matsuda. Hivemind: Towards cellular native machine learning model splitting. *IEEE Journal on Selected Areas in Communications*, 40 (2):626–640, 2021.

Xiaofei Wang, Yiwen Han, Victor CM Leung, Dusit Niyato, Xueqiang Yan, and Xu Chen. Convergence of edge computing and deep learning: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 22(2):869–904, 2020b.

Yao Wen, Guopeng Zhang, Kezhi Wang, and Kun Yang. Training latency minimization for model-splitting allowed federated edge learning. *IEEE Transactions on Network Science and Engineering*, 2025.

Wen Wu, Mushu Li, Kaige Qu, Conghao Zhou, Xuemin Shen, Weihua Zhuang, Xu Li, and Weisen Shi. Split learning over wireless networks: Parallel design and resource management. *IEEE Journal on Selected Areas in Communications*, 41(4):1051–1066, 2023.

Xinghao Wu, Xuefeng Liu, Jianwei Niu, Guogang Zhu, Mingjia Shi, Shaojie Tang, and Jing Yuan. Tackling feature-classifier mismatch in federated learning via prompt-driven feature transformation. *Advances in Neural Information Processing Systems*, 38:73655–73691, 2026.

Mingkun Yang, Ran Zhu, Qing Wang, and Jie Yang. Smofi: Step-wise momentum fusion for split federated learning on heterogeneous data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 27574–27582, 2026.

Ran Zhu, Mingkun Yang, and Qing Wang. Shufflefl: addressing heterogeneity in multi-device federated learning. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 8(2):1–34, 2024.

Ran Zhu, Mingkun Yang, Shiqiang Wang, Jie Yang, and Qing Wang. Flick: Empowering federated learning with commonsense knowledge. *Advances in Neural Information Processing Systems*, 38: 1235–1267, 2026.

A Implementation Details

A.1 Hyperparameter Settings

CIFAR10 and CIFAR100. We use ResNet-18 as the task model for both CIFAR-10 and CIFAR-100. Training uses SGDM with momentum 0.9, batch size $B = 32$, an initial learning rate of 5×10^{-2}, and weight decay 5×10^{-4}. The learning rate decays by a factor of 0.998 after each communication round. Each dataset contains 50,000 training samples, which are partitioned across $|\mathcal{J}| = 100$ clients according to a Dirichlet distribution with concentration parameter γ.

Tiny-ImageNet. For Tiny-ImageNet, we use ResNet-34 with SGDM, momentum 0.9, and batch size $B = 32$. The initial learning rate is 5×10^{-2} and decays by a factor of 0.998 per communication round, with weight decay 10^{-3}. The 100,000 training samples are distributed across 100 clients using the same Dirichlet partitioning scheme. By default, 10 clients participate in each round, each performing $|\tau| = 70$ local training steps, for a total of 200 communication rounds.

For all task models, the candidate cut set $\tilde{\mathcal{L}}$ contains residual-block boundaries starting from the second residual block, i.e., $L \geq 2$. This ensures that the dynamic splitting search space does not use a shallower cut than the default $L = 2$ adopted by the fixed-split baselines.

All experiments are executed on NVIDIA A40 and A100 GPUs. For PyramidSFL, we use a warm-up period of $N^{\text{wp}} = 25$ communication rounds with bandwidth-prioritization factor $\alpha = 0.5$ across all benchmarks. The weight-shrinking scaling factor is set to $\beta = 0.1$ for CIFAR-10 and CIFAR-100 and $\beta = 0.05$ for Tiny-ImageNet.

A.2 Visualization of System Heterogeneity

Our evaluation considers both data and system heterogeneity when estimating wall-clock training time. Beyond non-IID client data, clients differ in their computation and communication capabilities, which directly affect the latency of each candidate cut layer.

Specifically, each client $j \in \mathcal{J}$ is assigned a computation profile o_j^d, measured as the full-model inference latency (*s/sample*), and a communication bandwidth b_j (*Mbit/s*). We sample these profiles from AI Benchmark (Ignatov et al., 2019) and MobiPerf (Huang et al., 2011), respectively, to emulate heterogeneous edge devices and network conditions. We assume a resource-rich central server whose computation latency is

$$o^s = \frac{1}{\kappa |\mathcal{J}|} \sum_{j \in \mathcal{J}} o_j^d, \tag{8}$$

where κ controls the relative server-side computation capability. We set $\kappa = 100$ for all benchmarks.

For client j using cut layer L, the latency of one collaborative training step consists of client-side computation, server-side computation, and client–server communication:

$$t_j^{\text{d-cmp}}(L) = 3Bo_j^d \frac{\sum_{l=1}^{L} c_l}{\sum_{l=1}^{|\mathcal{W}|} c_l}, \tag{9}$$

$$t_j^{\text{s-cmp}}(L) = 3Bo^s \left(1 - \frac{\sum_{l=1}^{L} c_l}{\sum_{l=1}^{|\mathcal{W}|} c_l}\right), \tag{10}$$

and

$$t_j^{\text{com}}(L) = 2Ba_L \frac{1}{b_j}. \tag{11}$$

Here, c_l denotes the computational complexity of layer l, while a_L denotes the activation size at cut layer L. The factor of 3 in Equations 9–10 assumes that backward propagation requires twice the computation of forward inference (Lai et al., 2022); the factor of 2 in Equation 11 accounts for transmitting activations and returning their gradients. We profile layer-wise MACs and activation sizes using DeepSpeed.

Figures 4 and 5 visualize the resulting latency-optimal cuts $\mathcal{L}^*$ across seven heterogeneous system configurations. The selected cut varies substantially across clients as their computation and communication profiles change. In general, clients with more constrained computation favor shallower cuts to

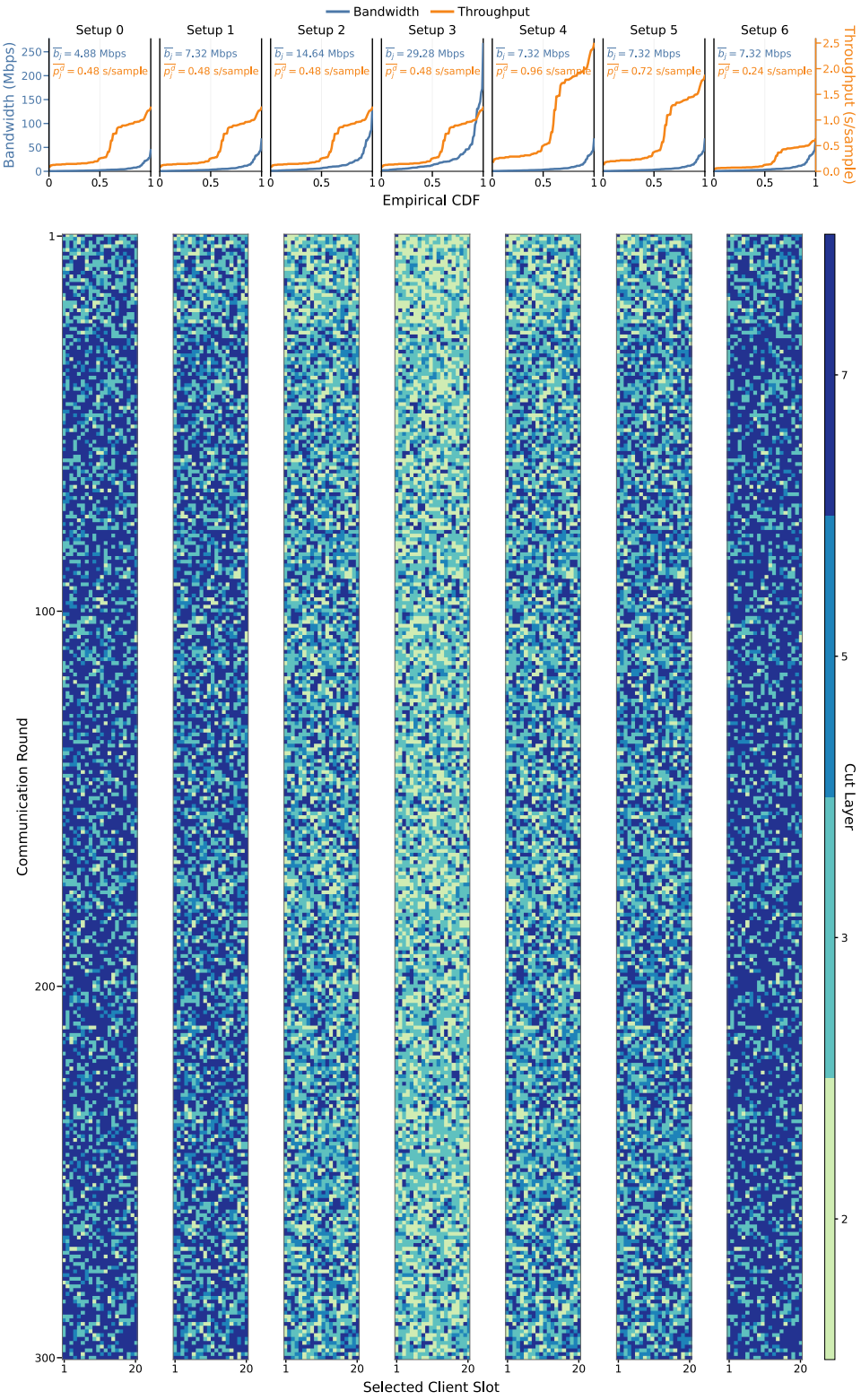

Figure 4: Visualization of latency-optimal $\mathcal{L}^*$ over 7 system heterogeneity settings for ResNet-18.

offload a larger portion of training to the server, whereas communication-constrained clients favor deeper cuts to reduce activation-transfer overhead. This variation illustrates the reason why a single static cut cannot simultaneously minimize training latency for heterogeneous clients and motivates the client-adaptive splitting used by PyramidSFL.

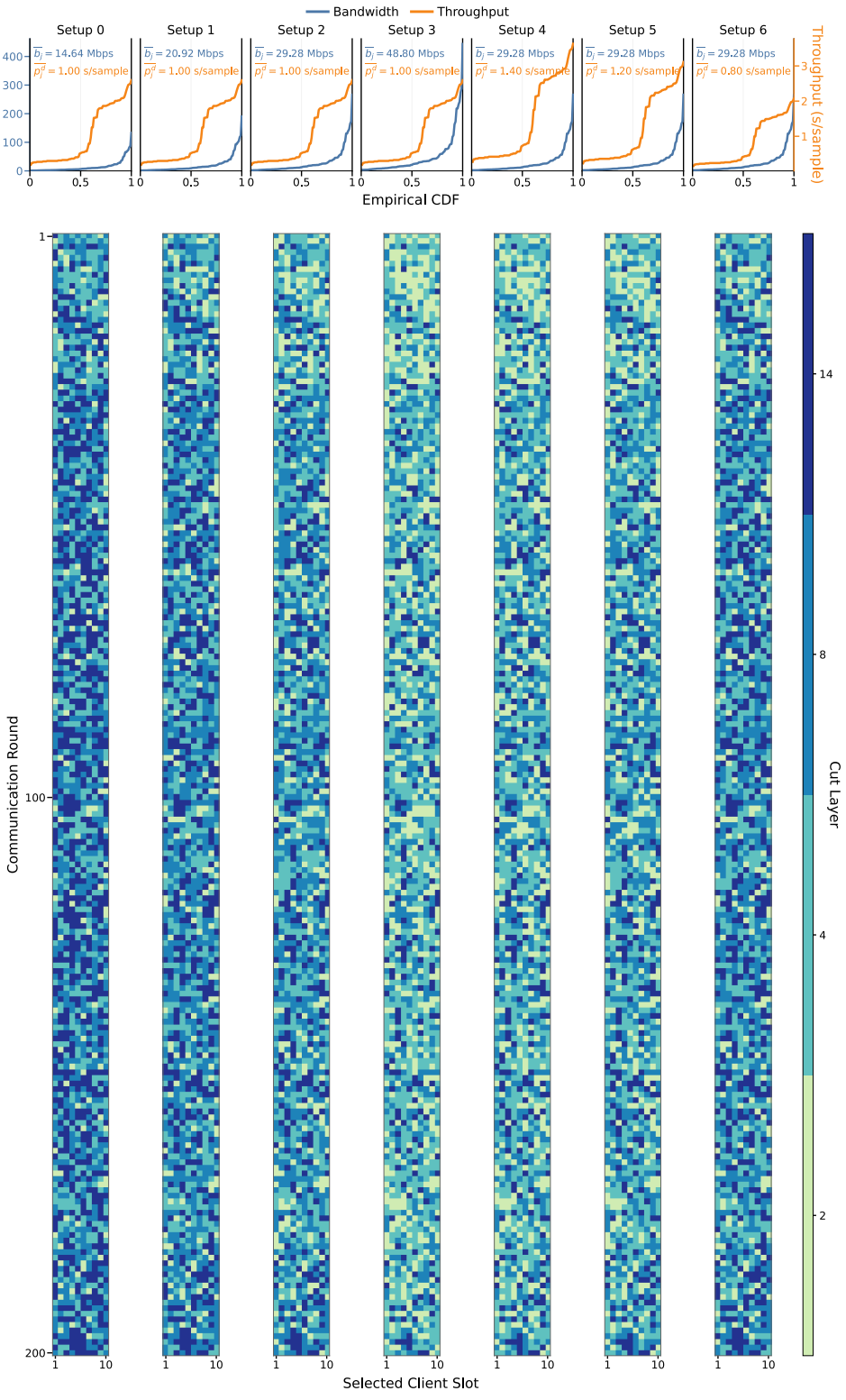

Figure 5: Visualization of latency-optimal $\mathcal{L}^*$ over 7 system heterogeneity settings for ResNet-34.

B PyramidSFL Workflow

Table 2 summarizes the notation used throughout this paper. We also provide the complete training workflow of PyramidSFL in Algorithm 1. At communication round n, the server first selects the

Table 2: Symbols and notations in the paper.

Symbol	Explanation
$\mathcal{J}$	Set of clients
$\mathcal{J}^n$	Subset of selected clients at round n
$\mathcal{J}^{(n,\tau)}$	Subset of active clients at training step τ
$\mathcal{J}_l^n$	Subset of selected clients assigned to cut layer l
$\mathcal{D}_j$	Local dataset of client j
$\mathcal{B}_j^\tau$	Mini-batch of client j at training step τ
$\mathcal{W}/\mathcal{W}_c/\mathcal{W}_s$	Weights of global/client-side/server-side model
L_j^n	Cut layer assigned to client j at round n
$\mathbf{L}^n$	Cut-layer assignments of selected clients at round n
$\widetilde{\mathcal{L}}$	Set of candidate cut layers
$\mathcal{L}^n$	Set of active cut layers at round n
L^{wp}	Fixed cut layer during warm-up
$\mathbf{A}_j$	A batch of Activations of client j
$\tilde{\mathbf{A}}$	Stacked activations from all participating clients
$\widehat{\mathbf{A}}_l$	Stacked activations from clients with cut layer l
$\dot{\mathbf{A}}_k$	Effective activations used to update server-side layer k
ρ_k	Gradient discrepancy of server-side layer k
ψ_k	Weight-shrinking factor of server-side layer k
$\mathfrak{L}$	Objective function
η	Learning rate
α	Bandwidth-aware sampling coefficient during warm-up
β	Weight-shrinking scaling factor
N/N^{wp}	Number of total/warm-up communication rounds
n/τ	Index of communication round/training step

participating clients $\mathcal{J}^n$. During the first N^{wp} warm-up rounds **(row 3)**, all selected clients train with the fixed cut L^{wp}, providing a stable optimization trajectory before dynamic model splitting is introduced. After warm-up, each client j adopts its latency-optimal cut L_j^* according to its current computation and communication profile **(row 7)**.

For each local training step, clients perform forward propagation up to their assigned cut layers and transmit the activations to the server **(row 11-13)**. Activations are grouped according to their cut layers and inserted into the corresponding layers of a shared server-side submodel. PyramidSFL then jointly optimizes this shared model using the stacked activations from all participating clients **(row 17)**, preserving the stacked-batch update of SA-SFL despite heterogeneous cut layers. The activation gradients are sent back to their corresponding clients for client-side backpropagation.

Dynamic splitting causes different server-side layers to receive updates from different subsets of client activations, which can introduce cross-layer update inconsistency. PyramidSFL therefore measures the gradient discrepancy for different cut groups and applies layer-wise weight shrinking after each server-side update **(row 22-23)**. The shrinking strength is calculated jointly by the observed gradient discrepancy and the scaling factor β, such that layers exposed to stronger update inconsistency receive stronger regularization.

Global Model Aggregation. After completing the local training steps, client-side submodels and the updated shared server-side model are aggregated and concatenated for the updated global model, which is used for the next communication round. Since clients may adopt different cut layers, their locally updated client-side submodels have different depths and cannot be directly aggregated. PyramidSFL addresses this by first reconstructing a full-length model for every participating client. For client j with cut layer L_j^*, its local update is concatenated with the corresponding suffix of the shared server-side submodel:

$$\tilde{\mathcal{W}}_j^{(n)} = \mathcal{W}_{c,j,1:L_j^*}^{(n,|\tau|)} \oplus \mathcal{W}_{s,L_j^*+1:|\mathcal{W}|}^{(n,|\tau|)}. \tag{12}$$

Algorithm 1: PyramidSFL Training

Input: Client set $\mathcal{J}$; number of rounds N; candidate cut layers $\tilde{\mathcal{L}}$; warm-up configuration $(N^{\mathrm{wp}}, L^{\mathrm{wp}}, \alpha)$; local steps $|\tau|$; initial global model $\mathcal{W}^{(0)}$; learning rate η; weight-shrinking scaling factor β.

Output: Updated global model $\mathcal{W}^{(N)}$.

for *round* $n = 1, 2, \ldots, N$ **do**
  if $n \leq N^{\mathrm{wp}}$ **then**
    // Warm-up Phase
    $\mathcal{J}^n \leftarrow$ Select a subset of clients from $\mathcal{J}$ with bandwidth-aware sampling;
    $L_j^n \leftarrow L^{\mathrm{wp}}, \quad \forall j \in \mathcal{J}^n$;
  else
    // Latency-optimal Splitting Phase
    $\mathcal{J}^n \leftarrow$ randomly select a subset of clients from $\mathcal{J}$;
    $\mathbf{L}^n \leftarrow$ By solving Equation 4;
  $\mathcal{L}^n \leftarrow \{L_j^n \mid j \in \mathcal{J}^n\}, \mathcal{J}_l^n \leftarrow \{j \in \mathcal{J}^n \mid L_j^n = l\}, \forall l \in \mathcal{L}^n$;
  $\mathcal{W}_{c,j}^{(n,0)}, \forall j \in \mathcal{J}^n \leftarrow \mathcal{W}^{(n-1)}, \mathcal{W}_s^{(n,0)} \leftarrow \mathcal{W}^{(n-1)}$;
  // Dynamic-Splitting Training
  for $\tau = 0, 1, \ldots, |\tau|-1$ **do**
    // Client-side Forward Propagation
    for $j \in \mathcal{J}^{(n,\tau)}$ ***in parallel*** **do**
      $\mathbf{A}_j^{(n,\tau)} \leftarrow \mathrm{Forward}\big(\mathcal{W}_{c,j}^{(n,\tau)}, \mathcal{B}_j^{\tau}\big)$;
      Send $\mathbf{A}_j^{(n,\tau)}$ to the server;
    // Activation Grouping
    for $l \in \mathcal{L}^n$ **do**
      $\hat{\mathbf{A}}_l^{(n,\tau)} \leftarrow \mathrm{Stack}\{\mathbf{A}_j^{(n,\tau)} \mid L_j^n = l\}$;
    $\tilde{\mathbf{A}}^{(n,\tau)} \leftarrow \bigcup_{l \in \mathcal{L}^n} \hat{\mathbf{A}}_l^{(n,\tau)}$;
    // Server-side Forward/Backward Propagation
    $\mathcal{W}_s^{(n,\tau+1)} \leftarrow \mathcal{W}_s^{(n,\tau)} - \eta \nabla_{\tilde{\mathbf{A}}^{(n,\tau)}}(\mathcal{W}_s^{(n,\tau)})$;
    // Layer-wise Weight Shrinking
    if $n > N^{\mathrm{wp}}$ **then**
      for *server-side layer* k **do**
        $\dot{\mathcal{L}}_k^n \leftarrow \{l \in \mathcal{L}^n \mid l < k\}$;
        $\dot{\mathbf{A}}_k^{(n,\tau)} \leftarrow \bigcup_{l \in \dot{\mathcal{L}}_k^n} \hat{\mathbf{A}}_l^{(n,\tau)}$;
        $\rho_k^{(n,\tau)} \leftarrow$ Equation 6;
        $\psi_k^{(n,\tau+1)} \leftarrow$ Equation 7;
        $\mathcal{W}_{s,k}^{(n,\tau+1)} \leftarrow \psi_k^{(n,\tau+1)} \mathcal{W}_{s,k}^{(n,\tau+1)}$;
    // Client-side Backpropagation
    for $j \in \mathcal{J}^{(n,\tau)}$ ***in parallel*** **do**
      Receive $\nabla_{\mathbf{A}_j^{(n,\tau)}}$ from the server;
      $\mathcal{W}_{c,j}^{(n,\tau+1)} \leftarrow \mathcal{W}_{c,j}^{(n,\tau)} - \eta \nabla_{\mathcal{B}_j^{\tau}}(\mathcal{W}_{c,j}^{(n,\tau)})$;
  // Global Aggregation
  $\bar{\mathcal{W}}_c^{(n)} \leftarrow$ Weighted averaging of $\{\mathcal{W}_{c,j}^{(n,|\tau|)}\}_{j \in \mathcal{J}}$;
  $\mathcal{W}^{(n)} \leftarrow [\bar{\mathcal{W}}_c^{(n)}, \mathcal{W}_s^{(n,|\tau|)}]$;

All reconstructed models consequently have the same structure as the global model. PyramidSFL then performs sample-weighted averaging:

$$\mathcal{W}^{(n)} = \sum_{j \in \mathcal{J}^n} \frac{\mathcal{D}_j^{(n)}}{\sum_{j' \in \mathcal{J}^n} \mathcal{D}_{j'}^{(n)}} \tilde{\mathcal{W}}_j^{(n)}. \tag{13}$$

This preserves a single global optimization trajectory under dynamic model splitting to heterogeneous client resource profiles. Therefore, PyramidSFL is a resource-aware SA-SFL framework that

Table 3: Performance comparison between PyramidSFL and baselines across three benchmarks under the more heterogeneous $\mathrm{Dir}_{100}(0.1)$ data distribution. We report the average and standard deviation of *Top-1 accuracy* under a fixed wall-clock budget, the wall-clock time (**T**) required to reach the target accuracy, and the corresponding time reduction (↓) or excess (↑) relative to FL. All results are averaged over three trials, with **bold** indicating the best performance for each setup.

Method	L	CIFAR-10/DIR$_{100}$(0.1)		CIFAR-100/DIR$_{100}$(0.1)		Tiny-ImageNet/DIR$_{100}$(0.1)	
		Acc.@150h(%)	T@60%(h)	Acc.@150h(%)	T@40%(h)	Acc.@300h(%)	T@25%(h)
FL	w/o	61.18±0.80	142.91	44.67±1.08	96.67	17.68±0.25	474.09
SFLV1	$L=2$	58.39±2.24	201.78↑41.19%	42.49±0.33	90.58↓6.30%	25.99±0.81	287.76↓39.30%
SFLV1*	$\mathcal{L}^*$	61.95±1.57	137.59↓3.72%	44.67±1.07	92.96↓3.84%	26.42±0.03	258.65↓45.44%
SFLV2	$L=2$	61.55±0.76	126.16↓11.72%	39.90±0.01	202.54↑109.52%	15.28±0.71	–
SA-SFL	$L=2$	65.07±2.76	95.81↓32.96%	43.43±0.50	111.03↑14.85%	25.27±0.01	281.05↓40.72%
SA-SFL*	$\mathcal{L}^*$	63.19±0.11	83.57↓41.52%	44.64±0.01	90.14↓6.75%	29.27±0.08	221.42↓53.30%
MergeSFL	$L=2$	68.32±1.77	86.77↓39.28%	43.75±1.00	105.53↑9.17%	24.51±0.04	312.90↓34.00%
ParallelSFL	$L=2$	55.18±2.16	178.22↑24.71%	43.20±0.92	116.69↑20.71%	21.47±0.04	345.50 ↓27.12%
SMoFi	$L=2$	60.40±1.39	160.47↑12.29%	43.19±0.90	80.94↓16.27%	23.95±0.61	354.08↓25.31%
PyramidSFL	$\mathcal{L}^*$	**72.27**±1.17	**47.99**↓66.42%	**48.11**±0.89	**69.69**↓27.91%	**32.35**±0.06	**176.18**↓62.84%

Table 4: Performance comparison between PyramidSFL and baselines across three benchmarks under the less heterogeneous $\mathrm{Dir}_{100}(0.5)$ data distribution. We report the average and standard deviation of *Top-1 accuracy* under a fixed wall-clock budget, the wall-clock time (**T**) required to reach the target accuracy, and the corresponding time reduction (↓) or excess (↑) relative to FL. All results are averaged over three trials, with **bold** indicating the best performance for each setup.

Method	L	CIFAR-10/DIR$_{100}$(0.5)		CIFAR-100/DIR$_{100}$(0.5)		Tiny-ImageNet/DIR$_{100}$(0.5)	
		Acc.@150h(%)	T@70%(h)	Acc.@150h(%)	T@40%(h)	Acc.@300h(%)	T@30%(h)
FL	w/o	79.56±0.07	60.40	50.11±0.36	50.47	26.61±0.17	364.17
SFLV1	$L=2$	76.24±0.06	66.73↑10.48%	47.00±0.98	51.96↑2.95%	35.40±0.06	216.01↓40.68%
SFLV1*	$\mathcal{L}^*$	80.01±0.09	58.24↓3.58%	50.11±0.36	48.55↓3.80%	34.71±0.58	195.89↓46.21%
SFLV2	$L=2$	78.25±0.12	44.95↓25.58%	50.19±0.51	35.68↓29.30%	23.69±0.90	–
SA-SFL	$L=2$	77.03±0.22	41.65↓31.04%	51.83±0.18	55.07↑9.11%	35.58±0.17	160.96↓55.80%
SA-SFL*	$\mathcal{L}^*$	77.50±0.03	35.21↓41.71%	48.47±0.28	37.16↓26.37%	39.60±0.56	119.75↓67.12%
MergeSFL	$L=2$	79.02±0.60	42.26↓30.03%	54.06±0.12	40.79↓19.18%	36.60±0.17	157.97↓56.62%
ParallelSFL	$L=2$	77.77±0.32	46.88↓22.38%	51.44±0.28	43.45↓13.91%	38.01±0.48	182.08↓50.00%
SMoFi	$L=2$	78.12±0.10	38.74↓35.86%	51.08±0.02	41.87↓17.04%	33.21±0.60	172.85↓52.54%
PyramidSFL	$\mathcal{L}^*$	**80.03**±0.46	**28.24**↓53.25%	**55.48**±0.10	**30.86**↓38.85%	**40.76**±0.09	**87.25**↓76.04%

supports varying cut layers to better balance computation and communication efficiency under system heterogeneity, while mitigating the performance degradation caused by dynamic model splitting.

C More Results

We provide additional experiments to validate whether the main observations remain consistent across different levels of data heterogeneity and across all three benchmarks. Specifically, we evaluate $\gamma \in \{0.1, 0.5\}$ in addition to the default $\gamma = 0.2$, and conduct the same ablation and sensitivity studies on CIFAR-10 and Tiny-ImageNet.

More Data Distributions. Tables 3 and 4 report results under stronger and more moderate data heterogeneity, respectively. Under the more challenging $\mathrm{Dir}_{100}(0.1)$ setting, some methods cannot reliably reach the accuracy targets used in the main experiment within the training setup. We therefore report T@60% for CIFAR-10 and T@25% for Tiny-ImageNet, while retaining T@40% for CIFAR-100. For $\mathrm{Dir}_{100}(0.5)$, we use the same target accuracies as in the main evaluation.

The overall observations remain consistent with the default $\mathrm{Dir}_{100}(0.2)$ setting: latency-optimal splitting improves the system efficiency of conventional SFL, but does not consistently provide the best global-model performance, whereas PyramidSFL achieves the best overall accuracy-latency trade-off across all three benchmarks, consistently outperforming the baselines in both fixed-budget accuracy and time-to-accuracy. This advantage persists across different levels of non-IID data and becomes more significant under the highly heterogeneous setting of $\gamma = 0.1$.

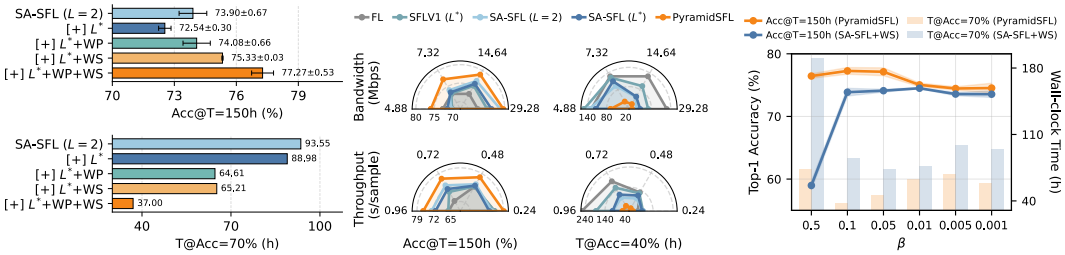

Figure 6: Additional ablation and sensitivity analysis on CIFAR-10 with ResNet-18. **Left:** Ablation of latency-optimal splitting $\mathcal{L}^*$, warm-up (WP), and weight shrinking (WS). **Middle:** Accuracy and time-to-accuracy under varying client communication and computation capabilities. **Right:** Sensitivity to the weight-shrinking scaling factor β.

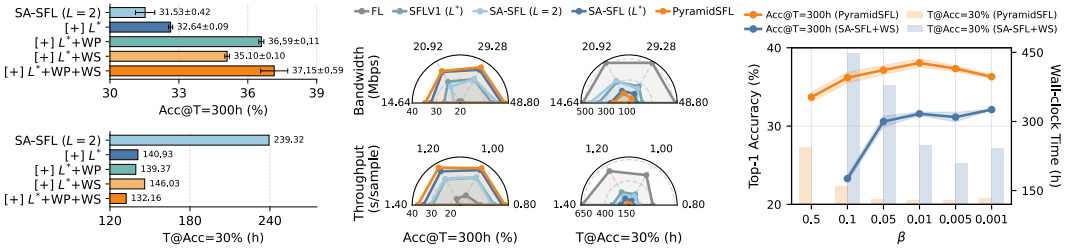

Figure 7: Additional ablation and sensitivity analysis on Tiny-ImageNet with ResNet-34. **Left:** Ablation of latency-optimal splitting $\mathcal{L}^*$, warm-up (WP), and weight shrinking (WS). **Middle:** Accuracy and time-to-accuracy under varying client communication and computation capabilities. **Right:** Sensitivity to the weight-shrinking scaling factor β.

Additional Ablation and Sensitivity Analysis. Figures 6 and 7 extend the analyses in the main experiment to CIFAR-10 and Tiny-ImageNet. The results show the same overall behavior as on CIFAR-100. In CIFAR-10, latency-optimal splitting alone improves training efficiency but can compromise global-model accuracy. In both CIFAR-10 and Tiny-ImageNet, warm-up and weight shrinking complement each other in recovering optimization quality. PyramidSFL also remains robust across varying communication and computation profiles, validating that its performance gain is not tied to a particular system configuration.

Under the same grid search over β, the best-tuned PyramidSFL outperforms SA-SFL with weight shrinking in both accuracy and convergence, showing that the performance gain of PyramidSFL is attributed to a combination of key components.

D Convergence Analysis of SA-SFL

This section proves Theorem 2.4 for fixed-cut SA-SFL with SGD and uniform partial participation. Following Han et al. (2024), we first write the client- and server-side updates in a common round-level form, then control the within-round model drift caused by stochastic gradients, data heterogeneity, and client sampling. We finally compare the resulting bound with SFLV1 and SFLV2 under the same setting.

D.1 Setup and Update Rules

We use the notation of Section 2. Under the uniform participation model, $p_j = 1/|\mathcal{J}|$, $m = |\mathcal{J}^n|$, and

$$\chi := \frac{|\mathcal{J}| - m}{|\mathcal{J}| - 1}. \tag{14}$$

Let $\mathcal{W}^{n-1} = [\mathcal{W}_c^{n-1}, \mathcal{W}_s^{n-1}]$ be the global model at the beginning of round n, and let $\mathcal{W}_j^{(n,\tau)} = [\mathcal{W}_{c,j}^{(n,\tau)}, \mathcal{W}_s^{(n,\tau)}]$ be the model used by client j at local step τ. For its mini-batch $\mathcal{B}_j^{(n,\tau)}$, define

$$\mathbf{g}_{c,j}^{(n,\tau)} := \nabla_{\mathcal{W}_c} \mathfrak{L}_{\mathcal{B}_j^{(n,\tau)}}(\mathcal{W}_j^{(n,\tau)}), \qquad \mathbf{g}_{s,j}^{(n,\tau)} := \nabla_{\mathcal{W}_s} \mathfrak{L}_{\mathcal{B}_j^{(n,\tau)}}(\mathcal{W}_j^{(n,\tau)}). \tag{15}$$

We use effective client- and server-side step sizes η_c and η_s, and $\eta_{\max} := \max\{\eta_c, \eta_s\}$.

At each step, the shared server-side submodel is updated using the mean gradient of the stacked activations,

$$\mathcal{W}_s^{(n,\tau+1)} = \mathcal{W}_s^{(n,\tau)} - \eta_s \frac{1}{m} \sum_{j \in \mathcal{J}^n} \mathbf{g}_{s,j}^{(n,\tau)}, \tag{16}$$

whereas each client updates

$$\mathcal{W}_{c,j}^{(n,\tau+1)} = \mathcal{W}_{c,j}^{(n,\tau)} - \eta_c \mathbf{g}_{c,j}^{(n,\tau)}. \tag{17}$$

The round starts from $\mathcal{W}_{c,j}^{(n,0)} = \mathcal{W}_c^{n-1}$ and $\mathcal{W}_s^{(n,0)} = \mathcal{W}_s^{n-1}$. After T local steps,

$$\mathcal{W}_c^n = \frac{1}{m} \sum_{j \in \mathcal{J}^n} \mathcal{W}_{c,j}^{(n,T)}, \qquad \mathcal{W}_s^n = \mathcal{W}_s^{(n,T)}. \tag{18}$$

Therefore,

$$\mathcal{W}_c^n - \mathcal{W}_c^{n-1} = -\eta_c \sum_{\tau=0}^{T-1} \frac{1}{m} \sum_{j \in \mathcal{J}^n} \mathbf{g}_{c,j}^{(n,\tau)}, \tag{19}$$

$$\mathcal{W}_s^n - \mathcal{W}_s^{n-1} = -\eta_s \sum_{\tau=0}^{T-1} \frac{1}{m} \sum_{j \in \mathcal{J}^n} \mathbf{g}_{s,j}^{(n,\tau)}. \tag{20}$$

The analysis permits arbitrary (η_c, η_s). With the implementation in Section 2, the stacked loss averages over all m client mini-batches. Thus, using a common optimizer learning rate η without rescaling the returned cut-layer gradients corresponds to the effective step sizes $(\eta_c, \eta_s) = (\eta/m, \eta)$.

Throughout the proof, expectations within round n are conditional on the history before that round; hence $\mathcal{W}^{n-1}$ is fixed.

D.2 Auxiliary Bounds

Lemma D.1 (Per-round smoothness decomposition)**.** *Under Assumption 2.1,*

$$\begin{aligned}\mathbb{E}[\mathfrak{L}(\mathcal{W}^n)] - \mathfrak{L}(\mathcal{W}^{n-1}) \le{}& \mathbb{E}\langle \nabla_{\mathcal{W}_c} \mathfrak{L}(\mathcal{W}^{n-1}), \mathcal{W}_c^n - \mathcal{W}_c^{n-1} \rangle + \frac{S}{2} \mathbb{E}\|\mathcal{W}_c^n - \mathcal{W}_c^{n-1}\|^2 \\ &+ \mathbb{E}\langle \nabla_{\mathcal{W}_s} \mathfrak{L}(\mathcal{W}^{n-1}), \mathcal{W}_s^n - \mathcal{W}_s^{n-1} \rangle + \frac{S}{2} \mathbb{E}\|\mathcal{W}_s^n - \mathcal{W}_s^{n-1}\|^2.\end{aligned} \tag{21}$$

Proof. A weighted sum of S-smooth local objectives is S-smooth. Applying the standard smoothness inequality to $\mathcal{W}^{n-1}$ and $\mathcal{W}^n$, and separating the two parameter blocks, gives Equation 21. □

Lemma D.2 (Uniform client sampling)**.** *Let $\bar{\mathbf{X}} = \sum_{j \in \mathcal{J}} p_j \mathbf{X}_j$, where the vectors $\mathbf{X}_j$ are independent of the sampled cohort $\mathcal{J}^n$. If m clients are drawn uniformly without replacement, then*

$$\mathbb{E}\Big[\frac{1}{m} \sum_{j \in \mathcal{J}^n} \mathbf{X}_j\Big] = \bar{\mathbf{X}}, \qquad \mathbb{E}\Big\|\frac{1}{m} \sum_{j \in \mathcal{J}^n} \mathbf{X}_j - \bar{\mathbf{X}}\Big\|^2 = \frac{\chi}{m} \sum_{j \in \mathcal{J}} p_j \|\mathbf{X}_j - \bar{\mathbf{X}}\|^2. \tag{22}$$

Proof. Let $K = |\mathcal{J}|$ and $I_j = \mathbf{1}\{j \in \mathcal{J}^n\}$. Uniform sampling gives $\mathbb{E}[I_j] = m/K$ and $\mathbb{E}[I_i I_j] = m(m-1)/(K(K-1))$ for $i \neq j$. Expanding the squared norm of $\frac{1}{m}\sum_j I_j(\mathbf{X}_j - \bar{\mathbf{X}})$ and using $\sum_j (\mathbf{X}_j - \bar{\mathbf{X}}) = \mathbf{0}$ yields Equation 22. □

Applying Lemma D.2 at the fixed model $\mathcal{W}^{n-1}$ and using Assumption 2.3 gives

$$\mathbb{E}\Big[\frac{1}{m}\sum_{j\in\mathcal{J}^n}\|\nabla_{\mathcal{W}_c}\mathfrak{L}_{\mathcal{D}_j}(\mathcal{W}^{n-1})-\nabla_{\mathcal{W}_c}\mathfrak{L}(\mathcal{W}^{n-1})\|^2\Big]\le\epsilon_c^2, \tag{23}$$

and analogously for the server block. Moreover,

$$\mathbb{E}\Big[\frac{1}{m}\sum_{j\in\mathcal{J}^n}\nabla_{\mathcal{W}_s}\mathfrak{L}_{\mathcal{D}_j}(\mathcal{W}^{n-1})\Big]=\nabla_{\mathcal{W}_s}\mathfrak{L}(\mathcal{W}^{n-1}), \tag{24}$$

$$\mathbb{E}\Big\|\frac{1}{m}\sum_{j\in\mathcal{J}^n}\nabla_{\mathcal{W}_s}\mathfrak{L}_{\mathcal{D}_j}(\mathcal{W}^{n-1})-\nabla_{\mathcal{W}_s}\mathfrak{L}(\mathcal{W}^{n-1})\Big\|^2\le\frac{\chi\epsilon_s^2}{m}, \tag{25}$$

with the same statements for the client block.

Lemma D.3 (Accumulated stochastic noise)**.** *Under Assumption 2.2, the stochastic gradients are conditionally unbiased. In addition, for any $\tau\le T$,*

$$\mathbb{E}\Big\|\sum_{\tau'=0}^{\tau-1}\frac{1}{m}\sum_{j\in\mathcal{J}^n}\big(\mathbf{g}_{s,j}^{(n,\tau')}-\nabla_{\mathcal{W}_s}\mathfrak{L}_{\mathcal{D}_j}(\mathcal{W}_j^{(n,\tau')})\big)\Big\|^2\le\frac{\tau\sigma_s^2}{m}, \tag{26}$$

$$\mathbb{E}\Big\|\sum_{\tau'=0}^{\tau-1}\big(\mathbf{g}_{c,j}^{(n,\tau')}-\nabla_{\mathcal{W}_c}\mathfrak{L}_{\mathcal{D}_j}(\mathcal{W}_j^{(n,\tau')})\big)\Big\|^2\le\tau\sigma_c^2. \tag{27}$$

The averaged analogue of Equation 26 also holds for the client block with variance σ_c^2.

Proof. At step τ, the current model depends only on previously sampled mini-batches. By Assumption 2.2, the new mini-batch gradient has zero-mean noise conditioned on that history. The noises of different client–step pairs are therefore uncorrelated. Summing their variances gives Equation 26–Equation 27. □

The following lemma is the main SA-SFL-specific step. It quantifies how far the client-specific full model can move away from the round-start model.

Lemma D.4 (Within-round drift of SA-SFL)**.** *Let Assumptions 2.1–2.3 hold and $\eta_{\max}\le 1/(8ST)$. Then*

$$\begin{aligned}\sum_{\tau=0}^{T-1}\frac{1}{m}\sum_{j\in\mathcal{J}^n}\mathbb{E}\|\mathcal{W}_j^{(n,\tau)}-\mathcal{W}^{n-1}\|^2&\le 3T^2\Big(\eta_c^2\sigma_c^2+\frac{\eta_s^2\sigma_s^2}{m}\Big)+2T^3\Big(\eta_c^2\epsilon_c^2+\frac{\eta_s^2\chi\epsilon_s^2}{m}\Big)\\&\quad+2T^3\Big(\eta_c^2\|\nabla_{\mathcal{W}_c}\mathfrak{L}(\mathcal{W}^{n-1})\|^2+\eta_s^2\|\nabla_{\mathcal{W}_s}\mathfrak{L}(\mathcal{W}^{n-1})\|^2\Big).\end{aligned} \tag{28}$$

Proof. Let

$$\Delta_\tau:=\frac{1}{m}\sum_{j\in\mathcal{J}^n}\mathbb{E}\|\mathcal{W}_j^{(n,\tau)}-\mathcal{W}^{n-1}\|^2. \tag{29}$$

Unrolling Equation 17 and Equation 16, we decompose each accumulated gradient into: (i) stochastic noise, (ii) the change of the local gradient from its value at $\mathcal{W}^{n-1}$, (iii) local heterogeneity or cohort-sampling error, and (iv) the global gradient. For the client side, the heterogeneity term is that of an individual client; for the server side, all clients share the same server model and their gradients are averaged, so the corresponding term is the cohort-sampling error. Using $\|\sum_{r=1}^4 x_r\|^2\le 4\sum_r\|x_r\|^2$, Lemma D.3, Lemma D.2, and Assumption 2.1, we obtain

$$\begin{aligned}\Delta_\tau&\le 4\tau\Big(\eta_c^2\sigma_c^2+\frac{\eta_s^2\sigma_s^2}{m}\Big)+4\tau^2\Big(\eta_c^2\epsilon_c^2+\frac{\eta_s^2\chi\epsilon_s^2}{m}\Big)\\&\quad+4\tau^2\Big(\eta_c^2\|\nabla_{\mathcal{W}_c}\mathfrak{L}(\mathcal{W}^{n-1})\|^2+\eta_s^2\|\nabla_{\mathcal{W}_s}\mathfrak{L}(\mathcal{W}^{n-1})\|^2\Big)+4\eta_{\max}^2S^2\tau\sum_{\tau'=0}^{\tau-1}\Delta_{\tau'}.\end{aligned} \tag{30}$$

The last term follows from

$$\|\nabla \mathfrak{L}_{\mathcal{D}_j}(\mathcal{W}_j^{(n,\tau')}) - \nabla \mathfrak{L}_{\mathcal{D}_j}(\mathcal{W}^{n-1})\|^2 \le S^2 \|\mathcal{W}_j^{(n,\tau')} - \mathcal{W}^{n-1}\|^2. \tag{31}$$

Summing Equation 30 over $\tau = 0, \dots, T-1$ and using $\sum_\tau \tau \le T^2/2$, $\sum_\tau \tau^2 \le T^3/3$, and

$$\sum_{\tau=0}^{T-1} \tau \sum_{\tau'=0}^{\tau-1} \Delta_{\tau'} \le \frac{T^2}{2} \sum_{\tau=0}^{T-1} \Delta_\tau, \tag{32}$$

gives

$$\begin{aligned}\sum_{\tau=0}^{T-1} \Delta_\tau \le\ & 2T^2 \Big(\eta_c^2 \sigma_c^2 + \frac{\eta_s^2 \sigma_s^2}{m}\Big) + \frac{4T^3}{3}\Big(\eta_c^2 \epsilon_c^2 + \frac{\eta_s^2 \chi \epsilon_s^2}{m}\Big) \\ & + \frac{4T^3}{3}\Big(\eta_c^2 \|\nabla_{\mathcal{W}_c} \mathfrak{L}(\mathcal{W}^{n-1})\|^2 + \eta_s^2 \|\nabla_{\mathcal{W}_s} \mathfrak{L}(\mathcal{W}^{n-1})\|^2\Big) + 2\eta_{\max}^2 S^2 T^2 \sum_{\tau=0}^{T-1} \Delta_\tau. \end{aligned} \tag{33}$$

Since $\eta_{\max} \le 1/(8ST)$, $2\eta_{\max}^2 S^2 T^2 \le 1/32$. Moving the last term to the left and slightly relaxing the constants yields Equation 28. □

D.3 Proof of Theorem 2.4

For $b \in \{c, s\}$, define

$$D_b^n := \sum_{\tau=0}^{T-1} \frac{1}{m} \sum_{j \in \mathcal{J}^n} \mathbb{E}\|\nabla_{\mathcal{W}_b} \mathfrak{L}_{\mathcal{D}_j}(\mathcal{W}_j^{(n,\tau)}) - \nabla_{\mathcal{W}_b} \mathfrak{L}_{\mathcal{D}_j}(\mathcal{W}^{n-1})\|^2. \tag{34}$$

Using the round-level updates Equation 19–Equation 20, conditional unbiasedness, Lemma D.2, and $-\langle x, y\rangle \le \frac{T}{2}\|x\|^2 + \frac{1}{2T}\|y\|^2$ gives, for either block,

$$\mathbb{E}\langle \nabla_{\mathcal{W}_b} \mathfrak{L}(\mathcal{W}^{n-1}), \mathcal{W}_b^n - \mathcal{W}_b^{n-1}\rangle \le -\frac{\eta_b T}{2}\|\nabla_{\mathcal{W}_b} \mathfrak{L}(\mathcal{W}^{n-1})\|^2 + \frac{\eta_b}{2} D_b^n. \tag{35}$$

Similarly, decomposing the summed stochastic gradient into stochastic noise, gradient change, sampling error, and the global gradient gives

$$\begin{aligned}\frac{S}{2}\mathbb{E}\|\mathcal{W}_b^n - \mathcal{W}_b^{n-1}\|^2 \le\ & \frac{S\eta_b^2 T \sigma_b^2}{m} + 2S\eta_b^2 T^2 \|\nabla_{\mathcal{W}_b} \mathfrak{L}(\mathcal{W}^{n-1})\|^2 \\ & + \frac{2S\eta_b^2 T^2 \chi \epsilon_b^2}{m} + 2S\eta_b^2 T D_b^n, \end{aligned} \tag{36}$$

where (σ_b, ϵ_b) denotes (σ_c, ϵ_c) or (σ_s, ϵ_s). Note that the $1/m$ factor in Equation 36 is due to the round-level average over participating clients. The distinction between the client and server sides appears in the within-round drift: the client-side local models evolve independently, whereas the server-side model is shared and updated by the stacked gradient.

Substituting Equation 35–Equation 36 for both blocks into Lemma D.1 yields

$$\begin{aligned}\mathbb{E}[\mathfrak{L}(\mathcal{W}^n)] - \mathfrak{L}(\mathcal{W}^{n-1}) \le\ & -\frac{T}{2} \sum_{b \in \{c,s\}} \eta_b \|\nabla_{\mathcal{W}_b} \mathfrak{L}(\mathcal{W}^{n-1})\|^2 + 2ST^2 \sum_{b \in \{c,s\}} \eta_b^2 \|\nabla_{\mathcal{W}_b} \mathfrak{L}(\mathcal{W}^{n-1})\|^2 \\ & + \frac{ST}{m}(\eta_c^2 \sigma_c^2 + \eta_s^2 \sigma_s^2) + \frac{2ST^2 \chi}{m}(\eta_c^2 \epsilon_c^2 + \eta_s^2 \epsilon_s^2) \\ & + \sum_{b \in \{c,s\}} \Big(\frac{\eta_b}{2} + 2S\eta_b^2 T\Big) D_b^n. \end{aligned} \tag{37}$$

Because $\eta_{\max} \le 1/(8ST)$,

$$2ST^2 \eta_b^2 \le \frac{T}{4}\eta_b, \qquad \frac{\eta_b}{2} + 2S\eta_b^2 T \le \frac{3}{4}\eta_{\max}. \tag{38}$$

Moreover, by Assumption 2.1,

$$D_c^n + D_s^n \le S^2 \sum_{\tau=0}^{T-1} \frac{1}{m} \sum_{j \in \mathcal{J}^n} \mathbb{E}\|\mathcal{W}_j^{(n,\tau)} - \mathcal{W}^{n-1}\|^2. \tag{39}$$

Applying Lemma D.4 to Equation 37 and absorbing the remaining gradient-dependent drift term into the descent term gives

$$\begin{aligned}\mathbb{E}[\mathfrak{L}(\mathcal{W}^n)] \le \mathfrak{L}(\mathcal{W}^{n-1}) &- \frac{T}{5}\Big(\eta_c\|\nabla_{\mathcal{W}_c}\mathfrak{L}(\mathcal{W}^{n-1})\|^2 + \eta_s\|\nabla_{\mathcal{W}_s}\mathfrak{L}(\mathcal{W}^{n-1})\|^2\Big) \\ &+ \frac{ST}{m}(\eta_c^2\sigma_c^2 + \eta_s^2\sigma_s^2) + \frac{2ST^2\chi}{m}(\eta_c^2\epsilon_c^2 + \eta_s^2\epsilon_s^2) \\ &+ \frac{9}{4}\eta_{\max}S^2T^2\Big(\eta_c^2\sigma_c^2 + \frac{\eta_s^2\sigma_s^2}{m}\Big) + \frac{3}{2}\eta_{\max}S^2T^3\Big(\eta_c^2\epsilon_c^2 + \frac{\eta_s^2\chi\epsilon_s^2}{m}\Big).\end{aligned} \tag{40}$$

Taking total expectation, summing Equation 40 over $n = 1, \dots, N$, and using $\mathbb{E}[\mathfrak{L}(\mathcal{W}^N)] \ge \mathfrak{L}^*$ telescopes the objective values. Multiplying by $5/(NT)$ yields

$$\begin{aligned}&\frac{1}{N}\sum_{n=0}^{N-1}\mathbb{E}\Big[\eta_c\|\nabla_{\mathcal{W}_c}\mathfrak{L}(\mathcal{W}^n)\|^2 + \eta_s\|\nabla_{\mathcal{W}_s}\mathfrak{L}(\mathcal{W}^n)\|^2\Big] \\ &\quad\le \frac{5(\mathfrak{L}(\mathcal{W}^0) - \mathfrak{L}^*)}{NT} + \frac{5S}{m}(\eta_c^2\sigma_c^2 + \eta_s^2\sigma_s^2) + \frac{10ST\chi}{m}(\eta_c^2\epsilon_c^2 + \eta_s^2\epsilon_s^2) \\ &\qquad+ \frac{45}{4}\eta_{\max}S^2T\Big(\eta_c^2\sigma_c^2 + \frac{\eta_s^2\sigma_s^2}{m}\Big) + \frac{15}{2}\eta_{\max}S^2T^2\Big(\eta_c^2\epsilon_c^2 + \frac{\eta_s^2\chi\epsilon_s^2}{m}\Big).\end{aligned} \tag{41}$$

Finally, $\frac{45}{4} \le 12$ and $\frac{15}{2} \le 8$, which gives exactly Equation 2. □

D.4 Convergence Rate

Corollary D.5 (Convergence rate of SA-SFL)**.** *Under Theorem 2.4, let* $\eta_c = \eta_s = \eta = \frac{1}{S}\sqrt{\frac{m}{NT}}$ *with* $N \ge 64mT$*, and define* $\sigma^2 = \sigma_c^2 + \sigma_s^2$ *and* $\epsilon^2 = \epsilon_c^2 + \epsilon_s^2$*. Then*

$$\begin{aligned}\frac{1}{N}\sum_{n=0}^{N-1}\mathbb{E}\|\nabla\mathfrak{L}(\mathcal{W}^n)\|^2 \le \frac{5\big(S(\mathfrak{L}(\mathcal{W}^0) - \mathfrak{L}^*) + \sigma^2\big)}{\sqrt{mNT}} + 10\chi\epsilon^2\sqrt{\frac{T}{mN}} \\ + \frac{12(m\sigma_c^2 + \sigma_s^2) + 8T(m\epsilon_c^2 + \chi\epsilon_s^2)}{N}.\end{aligned} \tag{42}$$

Proof. The condition $N \ge 64mT$ implies $\eta \le 1/(8ST)$. Divide Equation 2 by η and substitute $\eta = \frac{1}{S}\sqrt{m/(NT)}$; collecting the resulting terms gives Equation 42. □

Thus, for fixed m and T, the expected stationarity measure converges to zero as N increases. Under full participation ($\chi = 0$), the leading term scales as $\mathcal{O}(1/\sqrt{mNT})$ and the server-side heterogeneity term disappears. Under partial participation, the additional heterogeneity term scales as $\mathcal{O}(\chi\epsilon^2\sqrt{T/(mN)})$: it decreases with the number of rounds and cohort size, but increases with the number of local steps.

D.5 Comparison with SFLV1 and SFLV2

To isolate the optimization effect of stacked server updates, we apply the same analysis to SFLV1 and SFLV2 under the same fixed-cut, SGD, and client-participation setting. The three methods have the same round-level form Equation 19–Equation 20; they differ only in how the server-side model evolves within a round.

For SFLV1($\bar{\tau}$), the server maintains one server-side replica per participating client and averages these replicas every $\bar{\tau}$ steps, where $\bar{\tau}$ divides T. For SFLV2, one shared server model processes the mT client–server interactions sequentially; we assume a uniformly random ordering of these interactions.

Proposition D.6 (Comparison with SFLV1 and SFLV2)**.** *Under the conditions of Theorem 2.4, SFLV1($\bar{\tau}$) and SFLV2 satisfy the same bound as SA-SFL, except that the last two local-drift quantities are replaced by* $12\eta_{\max}S^2T\,\Sigma_\sigma + 8\eta_{\max}S^2T^2\,\Sigma_\epsilon$*, where Table 5 gives* $(\Sigma_\sigma, \Sigma_\epsilon)$*.*

Table 5: Local-drift quantities in Theorem 2.4 and Proposition D.6. The first three terms of the bound are identical for the three frameworks.

Framework	Σ_σ	Σ_ϵ
SFLV1 ($\bar{\tau}$)	$\eta_c^2\sigma_c^2 + \eta_s^2\sigma_s^2\left(\frac{1}{m} + \frac{\bar{\tau}-1}{T}\right)$	$\eta_c^2\epsilon_c^2 + \eta_s^2\epsilon_s^2\left(\frac{\chi}{m} + \frac{\bar{\tau}(\bar{\tau}-1)}{T^2}\right)$
SFLV2	$\eta_c^2\sigma_c^2 + \eta_s^2\frac{\sigma_s^2+\epsilon_s^2}{m}$	$\eta_c^2\epsilon_c^2 + \eta_s^2\frac{\chi\epsilon_s^2}{m}$
SA-SFL (Theorem 2.4)	$\eta_c^2\sigma_c^2 + \eta_s^2\frac{\sigma_s^2}{m}$	$\eta_c^2\epsilon_c^2 + \eta_s^2\frac{\chi\epsilon_s^2}{m}$

Proof. The proof of Theorem 2.4 depends on the training framework only through three ingredients: the round-level updates Equation 19–Equation 20, Lemma D.3, and the within-round drift bound of Lemma D.4. We verify the first two for SFLV1($\bar{\tau}$) and SFLV2 and then replace the third.

Update rules. Client-side training and aggregation are the same as in SA-SFL. The two frameworks differ only on the server side.

- In SFLV1($\bar{\tau}$), the replica of client $j \in \mathcal{J}^n$ starts from $\mathcal{W}_{s,j}^{(n,0)} = \mathcal{W}_s^{n-1}$ and is updated by $\mathcal{W}_{s,j}^{(n,\tau+1)} = \mathcal{W}_{s,j}^{(n,\tau)} - \eta_s \mathbf{g}_{s,j}^{(n,\tau)}$. At every step that is a multiple of $\bar{\tau}$, all replicas are replaced by their average. Client j uses the model $\mathcal{W}_j^{(n,\tau)} = [\mathcal{W}_{c,j}^{(n,\tau)}, \mathcal{W}_{s,j}^{(n,\tau)}]$.
- In SFLV2, let $\tilde{\mathcal{W}}_s^{(n,k)}$ be the shared server-side model after k of the mT interactions, with $\tilde{\mathcal{W}}_s^{(n,0)} = \mathcal{W}_s^{n-1}$. If the $(k+1)$-th interaction is local step τ of client j, then $\mathcal{W}_j^{(n,\tau)} = [\mathcal{W}_{c,j}^{(n,\tau)}, \tilde{\mathcal{W}}_s^{(n,k)}]$ and $\tilde{\mathcal{W}}_s^{(n,k+1)} = \tilde{\mathcal{W}}_s^{(n,k)} - \frac{\eta_s}{m}\mathbf{g}_{s,j}^{(n,\tau)}$. With a common learning rate η per interaction, this corresponds to $(\eta_c, \eta_s) = (\eta, m\eta)$.

Round-level updates. Since client-side training is unchanged, Equation 19 holds. For SFLV1($\bar{\tau}$), averaging is linear. Starting from an averaged model, the average of the replicas after the next $\bar{\tau}$ steps equals that model minus η_s times the sum of the averaged stochastic gradients over those steps. Since $\bar{\tau}$ divides T, the last averaging takes place at $\tau = T$, and Equation 20 follows. For SFLV2, summing the mT interactions gives $\mathcal{W}_s^n = \mathcal{W}_s^{n-1} - \frac{\eta_s}{m}\sum_{j\in\mathcal{J}^n}\sum_{\tau=0}^{T-1}\mathbf{g}_{s,j}^{(n,\tau)}$, which is again Equation 20.

Stochastic noise. In both frameworks, each stochastic gradient is evaluated at a model determined by the mini-batches used before it: the earlier steps in SFLV1, and the earlier interactions in SFLV2. Its own mini-batch is drawn independently of them, so the proof of Lemma D.3 applies verbatim.

Within-round drift. All steps of the proof of Theorem 2.4 other than Lemma D.4 therefore carry over unchanged. It remains to show that

$$\sum_{\tau=0}^{T-1}\frac{1}{m}\sum_{j\in\mathcal{J}^n}\mathbb{E}\|\mathcal{W}_j^{(n,\tau)} - \mathcal{W}^{n-1}\|^2 \le 3T^2\,\Sigma_\sigma + 2T^3\,\Sigma_\epsilon + 2T^3\Big(\eta_c^2\|\nabla_{\mathcal{W}_c}\mathfrak{L}(\mathcal{W}^{n-1})\|^2 + \eta_s^2\|\nabla_{\mathcal{W}_s}\mathfrak{L}(\mathcal{W}^{n-1})\|^2\Big), \tag{43}$$

with $(\Sigma_\sigma, \Sigma_\epsilon)$ from Table 5. For SA-SFL, this is exactly Equation 28. The client side of the drift is the same in all three frameworks. As in the proof of Lemma D.4, we unroll Equation 17 and split each client-side stochastic gradient into the stochastic noise, the change of the local gradient, the heterogeneity of client j, and the global gradient. Using Lemma D.3 and Equation 23, this gives, for

every τ,

$$\begin{aligned}
&\frac{1}{m}\sum_{j\in\mathcal{J}^n}\mathbb{E}\|\mathcal{W}_{c,j}^{(n,\tau)}-\mathcal{W}_c^{n-1}\|^2\\
&\quad\le 4\eta_c^2\Big[\tau\sigma_c^2+\tau^2\epsilon_c^2+\tau^2\|\nabla_{\mathcal{W}_c}\mathfrak{L}(\mathcal{W}^{n-1})\|^2\\
&\qquad\qquad+\tau\sum_{\tau'=0}^{\tau-1}\frac{1}{m}\sum_{j\in\mathcal{J}^n}\mathbb{E}\|\nabla_{\mathcal{W}_c}\mathfrak{L}_{\mathcal{D}_j}(\mathcal{W}_j^{(n,\tau')})-\nabla_{\mathcal{W}_c}\mathfrak{L}_{\mathcal{D}_j}(\mathcal{W}^{n-1})\|^2\Big].
\end{aligned}\tag{44}$$

Lemmas D.7 and D.9 bound the server side and prove Equation 43 for SFLV1($\bar\tau$) and SFLV2, respectively. Substituting Equation 43 for Equation 28 in the remainder of the proof of Theorem 2.4 yields the stated bound. □

Lemma D.7 (Within-round drift of SFLV1($\bar\tau$))**.** *Under the conditions of Theorem 2.4, SFLV1($\bar\tau$) satisfies Equation 43 with the SFLV1 row of Table 5.*

Proof. Fix a client $j\in\mathcal{J}^n$ and a step τ, and let $\tau_0:=\bar\tau\lfloor\tau/\bar\tau\rfloor$ be the last averaging step not later than τ. All replicas coincide at τ_0, so

$$\mathcal{W}_{s,j}^{(n,\tau)}-\mathcal{W}_s^{n-1}=-\eta_s\Big(\sum_{\tau'=0}^{\tau_0-1}\frac{1}{m}\sum_{i\in\mathcal{J}^n}\mathbf{g}_{s,i}^{(n,\tau')}+\sum_{\tau'=\tau_0}^{\tau-1}\mathbf{g}_{s,j}^{(n,\tau')}\Big).\tag{45}$$

The first sum is shared by all replicas, as in SA-SFL. The second sum is trained on the data of client j only. In each term we add and subtract the local gradient at the current model, the local gradient at $\mathcal{W}^{n-1}$, and the global gradient $\nabla_{\mathcal{W}_s}\mathfrak{L}(\mathcal{W}^{n-1})$. This splits the displacement into five parts:

(i) the stochastic noise, whose second moment is at most $\big(\frac{\tau_0}{m}+\tau-\tau_0\big)\sigma_s^2$ by Lemma D.3;

(ii) the change of local gradients over the τ steps. By the Cauchy–Schwarz and Jensen inequalities, its second moment is at most τ times the sum of the τ corresponding squared gradient differences;

(iii) τ_0 times the sampling error $\frac{1}{m}\sum_{i\in\mathcal{J}^n}\nabla_{\mathcal{W}_s}\mathfrak{L}_{\mathcal{D}_i}(\mathcal{W}^{n-1})-\nabla_{\mathcal{W}_s}\mathfrak{L}(\mathcal{W}^{n-1})$, whose second moment is at most $\frac{\tau_0^2\chi\epsilon_s^2}{m}$ by Equation 25;

(iv) $\tau-\tau_0$ times the heterogeneity $\nabla_{\mathcal{W}_s}\mathfrak{L}_{\mathcal{D}_j}(\mathcal{W}^{n-1})-\nabla_{\mathcal{W}_s}\mathfrak{L}(\mathcal{W}^{n-1})$ of client j;

(v) τ times the global gradient $\nabla_{\mathcal{W}_s}\mathfrak{L}(\mathcal{W}^{n-1})$.

We apply $\|\sum_{r=1}^5 x_r\|^2\le 5\sum_r\|x_r\|^2$, average over $j\in\mathcal{J}^n$, and bound part (iv) by the server-block version of Equation 23. After averaging over j, the two sums in part (ii) combine into a single sum over $\tau'<\tau$. This gives

$$\begin{aligned}
&\frac{1}{m}\sum_{j\in\mathcal{J}^n}\mathbb{E}\|\mathcal{W}_{s,j}^{(n,\tau)}-\mathcal{W}_s^{n-1}\|^2\\
&\quad\le 5\eta_s^2\Big[\Big(\frac{\tau_0}{m}+\tau-\tau_0\Big)\sigma_s^2+\Big(\frac{\chi\tau_0^2}{m}+(\tau-\tau_0)^2\Big)\epsilon_s^2+\tau^2\|\nabla_{\mathcal{W}_s}\mathfrak{L}(\mathcal{W}^{n-1})\|^2\\
&\qquad\qquad+\tau\sum_{\tau'=0}^{\tau-1}\frac{1}{m}\sum_{j\in\mathcal{J}^n}\mathbb{E}\|\nabla_{\mathcal{W}_s}\mathfrak{L}_{\mathcal{D}_j}(\mathcal{W}_j^{(n,\tau')})-\nabla_{\mathcal{W}_s}\mathfrak{L}_{\mathcal{D}_j}(\mathcal{W}^{n-1})\|^2\Big].
\end{aligned}\tag{46}$$

Compared with SA-SFL, where $\tau_0=\tau$, the local part adds the terms $(\tau-\tau_0)\sigma_s^2$ and $(\tau-\tau_0)^2\epsilon_s^2$.

Let Δ_τ be as in the proof of Lemma D.4. We add Equation 44, relaxing its constant 4 to 5. We then combine the two gradient-change terms by Assumption 2.1, as in the proof of Lemma D.4. This gives

$$\Delta_\tau\le 5\Big[\tau\eta_c^2\sigma_c^2+\eta_s^2\Big(\frac{\tau_0}{m}+\tau-\tau_0\Big)\sigma_s^2+\tau^2\eta_c^2\epsilon_c^2+\eta_s^2\Big(\frac{\chi\tau_0^2}{m}+(\tau-\tau_0)^2\Big)\epsilon_s^2$$

$$+\tau^2\Big(\eta_c^2\|\nabla_{\mathcal{W}_c}\mathfrak{L}(\mathcal{W}^{n-1})\|^2+\eta_s^2\|\nabla_{\mathcal{W}_s}\mathfrak{L}(\mathcal{W}^{n-1})\|^2\Big)\Big]+5\eta_{\max}^2S^2\tau\sum_{\tau'=0}^{\tau-1}\Delta_{\tau'}. \quad (47)$$

We sum over $\tau=0,\dots,T-1$, using $\sum_\tau\tau_0\le\frac{T^2}{2}$, $\sum_\tau\tau_0^2\le\frac{T^3}{3}$, $\sum_\tau(\tau-\tau_0)=\frac{T(\bar\tau-1)}{2}$, and

$$\sum_{\tau=0}^{T-1}(\tau-\tau_0)^2=\frac{T(\bar\tau-1)(2\bar\tau-1)}{6}\le\frac{T^3}{3}\cdot\frac{\bar\tau(\bar\tau-1)}{T^2}. \quad (48)$$

This gives

$$\sum_{\tau=0}^{T-1}\Delta_\tau\le\frac{5}{2}T^2\,\Sigma_\sigma+\frac{5}{3}T^3\Big(\Sigma_\epsilon+\eta_c^2\|\nabla_{\mathcal{W}_c}\mathfrak{L}(\mathcal{W}^{n-1})\|^2+\eta_s^2\|\nabla_{\mathcal{W}_s}\mathfrak{L}(\mathcal{W}^{n-1})\|^2\Big)$$
$$+\frac{5}{2}\eta_{\max}^2S^2T^2\sum_{\tau=0}^{T-1}\Delta_\tau. \quad (49)$$

Since $\eta_{\max}\le1/(8ST)$, we have $\frac{5}{2}\eta_{\max}^2S^2T^2\le\frac{5}{128}$. Moving the last term to the left, and using $\frac{128}{123}\cdot\frac{5}{2}\le3$ and $\frac{128}{123}\cdot\frac{5}{3}\le2$, yields Equation 43. □

Lemma D.8 (Random reshuffling)**.** *Let $\mathbf{y}_1,\dots,\mathbf{y}_M$ satisfy $\sum_{i=1}^M\mathbf{y}_i=\mathbf{0}$, and let $(i_1,\dots,i_M)$ be a uniformly random permutation of $\{1,\dots,M\}$. Then, for $k\le M$,*

$$\mathbb{E}\Big\|\sum_{\ell=1}^k\mathbf{y}_{i_\ell}\Big\|^2=\frac{k(M-k)}{M-1}\cdot\frac{1}{M}\sum_{i=1}^M\|\mathbf{y}_i\|^2\le\frac{k}{M}\sum_{i=1}^M\|\mathbf{y}_i\|^2. \quad (50)$$

Proof. Each i_ℓ is uniform on $\{1,\dots,M\}$, and each pair $(i_\ell,i_{\ell'})$ with $\ell\ne\ell'$ is uniform over ordered pairs of distinct indices. We also have $\sum_{i\ne i'}\langle\mathbf{y}_i,\mathbf{y}_{i'}\rangle=\|\sum_i\mathbf{y}_i\|^2-\sum_i\|\mathbf{y}_i\|^2=-\sum_i\|\mathbf{y}_i\|^2$. Hence

$$\mathbb{E}\Big\|\sum_{\ell=1}^k\mathbf{y}_{i_\ell}\Big\|^2=\frac{k}{M}\sum_i\|\mathbf{y}_i\|^2-\frac{k(k-1)}{M(M-1)}\sum_i\|\mathbf{y}_i\|^2=\frac{k(M-k)}{M-1}\cdot\frac{1}{M}\sum_i\|\mathbf{y}_i\|^2. \quad (51)$$

The inequality follows from $\frac{M-k}{M-1}\le1$. □

Lemma D.9 (Within-round drift of SFLV2)**.** *Under the conditions of Theorem 2.4, SFLV2 satisfies Equation 43 with the SFLV2 row of Table 5.*

Proof. For $k'<mT$, let $j_{k'}$ and $\tau_{k'}$ denote the client and its local step in the $(k'+1)$-th interaction. This interaction evaluates $\mathbf{g}_{s,j_{k'}}^{(n,\tau_{k'})}$ at $\mathcal{W}_{j_{k'}}^{(n,\tau_{k'})}$, so

$$\tilde{\mathcal{W}}_s^{(n,k)}-\mathcal{W}_s^{n-1}=-\frac{\eta_s}{m}\sum_{k'=0}^{k-1}\mathbf{g}_{s,j_{k'}}^{(n,\tau_{k'})}. \quad (52)$$

For each client j, we split its gradient at $\mathcal{W}^{n-1}$ into the global gradient, the sampling error, and the deviation of client j from the cohort average:

$$\nabla_{\mathcal{W}_s}\mathfrak{L}_{\mathcal{D}_j}(\mathcal{W}^{n-1})=\nabla_{\mathcal{W}_s}\mathfrak{L}(\mathcal{W}^{n-1})+\Big(\frac{1}{m}\sum_{i\in\mathcal{J}^n}\nabla_{\mathcal{W}_s}\mathfrak{L}_{\mathcal{D}_i}(\mathcal{W}^{n-1})-\nabla_{\mathcal{W}_s}\mathfrak{L}(\mathcal{W}^{n-1})\Big)$$
$$+\Big(\nabla_{\mathcal{W}_s}\mathfrak{L}_{\mathcal{D}_j}(\mathcal{W}^{n-1})-\frac{1}{m}\sum_{i\in\mathcal{J}^n}\nabla_{\mathcal{W}_s}\mathfrak{L}_{\mathcal{D}_i}(\mathcal{W}^{n-1})\Big). \quad (53)$$

The displacement then splits into five parts:

(i) the stochastic noise, whose second moment is at most $k\sigma_s^2$ by Lemma D.3;

(ii) the change of local gradients. By the Cauchy–Schwarz inequality, its second moment is at most k times the sum of its k squared terms;

(iii) k times the sampling error, whose second moment is at most $\frac{k^2\chi\epsilon_s^2}{m}$ by Equation 25;

(iv) k times the global gradient $\nabla_{\mathcal{W}_s}\mathfrak{L}(\mathcal{W}^{n-1})$;

(v) the sum of the deviations of the first k interacting clients from the cohort average.

Part (v) is absent in SA-SFL and reflects the order in which clients are served. The deviations sum to zero over the multiset of all mT interactions, in which each client of $\mathcal{J}^n$ appears T times. The interaction order is a uniformly random permutation of this multiset, independent of $\mathcal{W}^{n-1}$ and $\mathcal{J}^n$. Lemma D.8 with $M = mT$ then gives

$$\mathbb{E}\Big\|\sum_{k'=0}^{k-1}\Big(\nabla_{\mathcal{W}_s}\mathfrak{L}_{\mathcal{D}_{j_{k'}}}(\mathcal{W}^{n-1}) - \frac{1}{m}\sum_{i\in\mathcal{J}^n}\nabla_{\mathcal{W}_s}\mathfrak{L}_{\mathcal{D}_i}(\mathcal{W}^{n-1})\Big)\Big\|^2 \\ \le k\cdot\frac{1}{m}\sum_{j\in\mathcal{J}^n}\|\nabla_{\mathcal{W}_s}\mathfrak{L}_{\mathcal{D}_j}(\mathcal{W}^{n-1}) - \nabla_{\mathcal{W}_s}\mathfrak{L}(\mathcal{W}^{n-1})\|^2. \tag{54}$$

Here we also use that the cohort average minimizes the sum of squared distances. By the server-block version of Equation 23, the expectation of the right-hand side is at most $k\epsilon_s^2$. Using $\|\sum_{r=1}^5 x_r\|^2 \le 5\sum_r\|x_r\|^2$, we obtain

$$\mathbb{E}\|\tilde{\mathcal{W}}_s^{(n,k)} - \mathcal{W}_s^{n-1}\|^2 \\ \le \frac{5\eta_s^2}{m^2}\Big[k\sigma_s^2 + k\epsilon_s^2 + k^2\Big(\frac{\chi\epsilon_s^2}{m} + \|\nabla_{\mathcal{W}_s}\mathfrak{L}(\mathcal{W}^{n-1})\|^2\Big) \\ + k\sum_{k'<k}\mathbb{E}\|\nabla_{\mathcal{W}_s}\mathfrak{L}_{\mathcal{D}_{j_{k'}}}(\mathcal{W}_{j_{k'}}^{(n,\tau_{k'})}) - \nabla_{\mathcal{W}_s}\mathfrak{L}_{\mathcal{D}_{j_{k'}}}(\mathcal{W}^{n-1})\|^2\Big]. \tag{55}$$

Each pair (j,τ) corresponds to exactly one interaction. The server-side part of the drift is therefore $\frac{1}{m}\sum_{k=0}^{mT-1}\mathbb{E}\|\tilde{\mathcal{W}}_s^{(n,k)} - \mathcal{W}_s^{n-1}\|^2$. Likewise, the sum over k of the gradient-change terms equals $\sum_{\tau=0}^{T-1}\sum_{j\in\mathcal{J}^n}\mathbb{E}\|\nabla_{\mathcal{W}_s}\mathfrak{L}_{\mathcal{D}_j}(\mathcal{W}_j^{(n,\tau)}) - \nabla_{\mathcal{W}_s}\mathfrak{L}_{\mathcal{D}_j}(\mathcal{W}^{n-1})\|^2$. Using $\sum_{k<mT}k \le \frac{(mT)^2}{2}$, $\sum_{k<mT}k^2 \le \frac{(mT)^3}{3}$, and $\sum_k k\sum_{k'<k}a_{k'} \le \frac{(mT)^2}{2}\sum_k a_k$, the server-side part is at most

$$5\eta_s^2\Big[\frac{T^2}{2}\cdot\frac{\sigma_s^2+\epsilon_s^2}{m} + \frac{T^3}{3}\Big(\frac{\chi\epsilon_s^2}{m} + \|\nabla_{\mathcal{W}_s}\mathfrak{L}(\mathcal{W}^{n-1})\|^2\Big)\Big] \\ + \frac{5}{2}\eta_s^2T^2\sum_{\tau=0}^{T-1}\frac{1}{m}\sum_{j\in\mathcal{J}^n}\mathbb{E}\|\nabla_{\mathcal{W}_s}\mathfrak{L}_{\mathcal{D}_j}(\mathcal{W}_j^{(n,\tau)}) - \nabla_{\mathcal{W}_s}\mathfrak{L}_{\mathcal{D}_j}(\mathcal{W}^{n-1})\|^2. \tag{56}$$

Compared with SA-SFL, part (v) adds $\eta_s^2\epsilon_s^2/m$ to Σ_σ.

For the client side, we sum Equation 44 over τ, using $\sum_\tau\tau\sum_{\tau'<\tau}a_{\tau'} \le \frac{T^2}{2}\sum_\tau a_\tau$. This bounds it by $4\eta_c^2\big[\frac{T^2}{2}\sigma_c^2 + \frac{T^3}{3}(\epsilon_c^2 + \|\nabla_{\mathcal{W}_c}\mathfrak{L}(\mathcal{W}^{n-1})\|^2)\big]$ plus the client-side gradient-change term with coefficient $2\eta_c^2T^2$. We add the two sides, relax the client-side constants to those of the server side, and combine the gradient-change terms by Assumption 2.1 as in the proof of Lemma D.4. This gives

$$\sum_{\tau=0}^{T-1}\Delta_\tau \le \frac{5}{2}T^2\,\Sigma_\sigma + \frac{5}{3}T^3\Big(\Sigma_\epsilon + \eta_c^2\|\nabla_{\mathcal{W}_c}\mathfrak{L}(\mathcal{W}^{n-1})\|^2 + \eta_s^2\|\nabla_{\mathcal{W}_s}\mathfrak{L}(\mathcal{W}^{n-1})\|^2\Big) \\ + \frac{5}{2}\eta_{\max}^2S^2T^2\sum_{\tau=0}^{T-1}\Delta_\tau, \tag{57}$$

and we conclude as in the proof of Lemma D.7. □

Under identical effective (η_c,η_s), the local-drift bound of SA-SFL is no larger than those of SFLV1($\bar{\tau}$) and SFLV2. SFLV1 accumulates drift because each server replica follows one client's data between averaging points; SFLV2 avoids replicas but introduces an order-dependent term through sequential updates. SA-SFL instead evaluates all participating-client gradients at the same server-side model and averages them at every step. Consequently, its server-side heterogeneity appears only through client-sampling error, which vanishes under full participation. With identical mini-batches, SA-SFL also follows the same trajectory as SFLV1($\bar{\tau}=1$), but needs only one shared server-side submodel.