SMoFi: Step-wise Momentum Fusion for Split Federated Learning on Heterogeneous Data

Mingkun Yang, Ran Zhu, Qing Wang, Jie Yang
Delft University of Technology
{m.yang-3, r.zhu-1, qing.wang, j.yang-3}@tudelft.nl

Abstract

Split Federated Learning is a system-efficient federated learning paradigm that leverages the rich computing resources at a central server to train model partitions. Data heterogeneity across silos, however, presents a major challenge undermining the convergence speed and accuracy of the global model. This paper introduces **Step-wise Momentum Fusion (SMoFi)**, an effective and lightweight framework that counteracts gradient divergence arising from data heterogeneity by synchronizing the momentum buffers across server-side optimizers. To control gradient divergence over the training process, we design a staleness-aware alignment mechanism that imposes constraints on gradient updates of the server-side submodel at each optimization step. Extensive validations on multiple real-world datasets show that SMoFi consistently improves global model accuracy (up to 7.1%) and convergence speed (up to 10.25×). Furthermore, SMoFi has a greater impact with more clients involved and deeper learning models, making it particularly suitable for model training in resource-constrained contexts.

Extended version — https://arxiv.org/abs/2511.09828

1 Introduction

The proliferation of mobile and sensing devices (Gubbi et al. 2013; Fortino and Trunfio 2014) has resulted in rich data at the edge, enabling a new range of Artificial Intelligence of Things applications (Wang et al. 2020b; Verbraeken et al. 2020). In this context, Federated Learning (FL) has been proposed as an important learning paradigm that exploits data generated at distributed edge devices while preserving data privacy (Konečnỳ et al. 2016; McMahan et al. 2017a; Bonawitz et al. 2019). On-device training of large models, however, remains a key challenge due to limited computing resources at the edge. To expedite on-device model training, prior work has explored collaborative model training, resorting to richer computing resources (Li et al. 2018; Eshratifar, Abrishami, and Pedram 2019; Wang et al. 2021). A promising approach is split learning (Gupta and Raskar 2018; Vepakomma et al. 2018b; Singh et al. 2019; Gao et al. 2020; Thapa et al. 2022), which divides a model into submodels trained separately on the edge and cloud, thereby offloading partial training overhead to the powerful server.

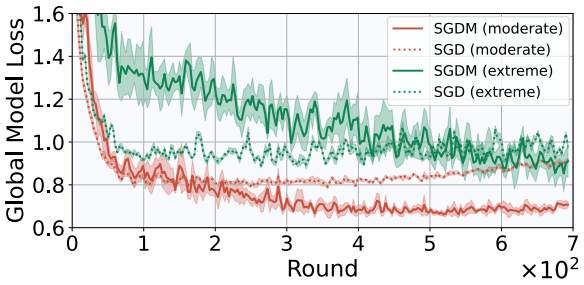

Figure 1: Momentum in local optimizers improves model performance on both moderate and extreme non-IID data in the long run, albeit slows down the learning.

The effectiveness of split FL, in terms of both model accuracy and convergence, is largely undermined by data heterogeneity–i.e., non-IID (non-identically and/or independently distributed) data silos–often presenting in real-world scenarios. Training on such non-IID data introduces inconsistency in model updates across both client-side and server-side submodels; such a divergence accumulates as the training proceeds over iterations. Consequently, aggregation of diverse model updates results in inferior accuracy and slower convergence of the global model compared with the IID setting. Existing methods to address such a challenge in FL either modify the loss function (Li et al. 2020; Li, He, and Song 2021; Gao et al. 2022) or impose robustness constraints on server aggregation (Hsu, Qi, and Brown 2019; Wang et al. 2020a; Reddi et al. 2020; Shi et al. 2025). While many of them are adaptable to split context, they overlook a unique attribute of split learning: the server directly controls the learning pace of multiple surrogate server-side models (each paired with a client-side submodel) that typically constitute the majority of the full model. This leads us to ask: *Can we impose constraints on model training in split FL by leveraging its inherent client-server interaction–enforced more tightly and synchronously than in conventional FL–without introducing additional overheads or privacy risk?*

In this paper, we propose SMoFi, a split FL framework that fuses momentum on a step basis to reduce weight inconsistency for learning with non-IID data. Momentum, as found in previous work, can improve FL in general to reach better model performance (Wang et al. 2019a); straightfor-

ward integration of stochastic gradient descent with momentum (SGDM) into local training of FL, however, slows down the learning as the local models converges towards their respective local optima better, making increasingly divergent updates, as shown in Figure 1. Such an effect is more significant when the data is more heterogeneous, e.g., green lines in the figure. In SMoFi, we propose to align optimization trajectories across server-side optimizers by synchronizing their momentum buffers in every step. As a result, SMoFi turns momentum from a slowing-down factor to a mechanism that speeds up model convergence while at the same time benefiting from the better performance brought about by momentum. By fusing momentum, SMoFi makes minimum changes to the existing split FL framework and thus can be plugged into existing FL methods. SMoFi is carefully designed to cope with a specific challenge of step-wise momentum fusion: certain devices with fewer batches finish the training earlier in the same round, not contributing to momentum alignment for model updates of the other devices. To maintain the constraint imposed by momentum alignment on the server-side models over the entire round, SMoFi fuses historical momentum with a staleness factor.

In summary, we make the following key contributions:

- We propose a novel split FL framework that improves the consistency of server-side submodel updates on non-IID data via step-wise momentum fusion.
- We introduce the staleness factor to counteract the diminishing momentum buffers synchronization to maintain the effectiveness of momentum fusion over the entire training process, especially as the number of local steps varies across participating clients.
- We conduct extensive evaluations and demonstrate that SMoFi significantly improves both the model accuracy (up to 7.1%) and convergence speed (up to 10.25×). Furthermore, our framework provides greater improvements when the number of clients increases and for larger models, making it particularly suitable for FL in resource-constrained contexts.

2 The SMoFi Framework

Our SMoFi is built upon the split FL framework to mitigate inconsistencies in weight updates caused by data heterogeneity, thereby facilitating more stable convergence toward the global optimum. Leveraging the modular structure of split FL, we introduce a step-wise momentum fusion as SMoFi's central design: at every SGD step, momentum buffers across all server-side solvers are synchronized. This alignment of optimization trajectories imposes constant constraints on inconsistent weight evolution during the training procedure. Our SMoFi also maintains a client-transparent architecture–requiring no changes or additional computation on the client side–thereby preserving the same privacy guarantees as existing frameworks such as SFLV1 and SFLV2 (Thapa et al. 2022). In the following, we present the SMoFi framework in detail, beginning with the general split FL structure and then presenting our proposed momentum alignment strategy.

2.1 Collaborative Training in Split FL

In split FL, a central server with rich computing resources collaborates with a set of clients $\mathcal{J}$, each possessing local data $\mathcal{D}_j$ ($j \in \mathcal{J}$), to collaboratively train a task model $f^{\mathcal{W}}$. Given the index of cut layer L, the model is split into the client-side model $f^{\mathcal{W}_c}$ and the server-side model $f^{\mathcal{W}_s}$: $f^{\mathcal{W}}(\cdot) = f^{\mathcal{W}_s}(f^{\mathcal{W}_c}(\cdot))$, 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 $d = d_c + d_s$. For simplicity, we assume a fixed cut layer across all clients and communication rounds, and 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). split FL aims to find the optimal $\mathcal{W}^*$ as in FL

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

wherein, the global objective function $\mathcal{L}(\mathcal{W})$ is the weighted sum of local objectives $\{\mathcal{L}_{\mathcal{D}_j}(\mathcal{W})\}_{j \in \mathcal{J}}$, with weights satisfying $\sum_{j \in \mathcal{J}} p_j = 1$, e.g., the fraction of local samples. The optimal global model parameters $\mathcal{W}^*$ are approached by having each client optimize its local objective and then aggregating these local model parameters, iterating over the N communication rounds.

In round $n \in [N]$ ($[N] = \{1, \cdots, N\}$), considering the communication bandwidth and client availability, the central server randomly selects a subset of devices $\mathcal{J}^n \subseteq \mathcal{J}$ to perform SGD updating the $\mathcal{W}_c$ and $\mathcal{W}_s$ over the $\{T_j\}_{j \in \mathcal{J}^n}$ local steps. The number of steps $T_j = E\lfloor \frac{|\mathcal{D}_j|}{B} \rfloor$ ($\lfloor \cdot \rfloor$ is the floor function) depending on local epochs E, mini-batch size B, and the size of local samples $\mathcal{D}_j$ that varies across clients.

One-step SGD. All the participating clients perform one step of SGD in parallel. The j-th client bootstraps the local SGD by propagating forward on a randomly selected sample batch. It then offloads activations–also called *smashed data* at the layer L–to the server. We denote activations as $\mathbf{A}_j^{(n,\tau)} = \{f^{\mathcal{W}_c}(\mathbf{x})\}_{\mathbf{x} \in \mathcal{B}_j^\tau}$ where $\tau \in [T_j]$ is the SGD step index and $\mathcal{B}_j^\tau \subseteq \mathcal{D}_j$ is the sample batch. The server proceeds with forward propagation $\hat{\mathbf{Y}} = \{f^{\mathcal{W}_s}(a)\}_{a \in \mathbf{A}_j^{(n,\tau)}}$, followed by gradient descent on the surrogate server-side model $\mathcal{W}_{s,j}^{(n,\tau+1)} = \mathcal{W}_{s,j}^{(n,\tau)} - \eta \nabla \mathcal{L}_{\mathcal{B}_j^\tau}(\mathcal{W}_{s,j}^{(n,\tau)})$ with the learning rate η. Specifically, the stochastic gradients on the server-side model are $\nabla \mathcal{L}_{\mathcal{B}_j^\tau}(\mathcal{W}_{s,j}^{(n,\tau)}) = \frac{1}{|\mathcal{B}_j^\tau|} \sum_{\mathbf{x} \in \mathcal{B}_j^\tau} \nabla l(\mathbf{x}; \mathcal{W}_{s,j}^{(n,\tau)})$, and $l(\cdot)$ is the loss function, e.g., cross-entropy loss between $\hat{\mathbf{Y}}$ and labels shared from clients[1]. The server sends the gradients on the cut layer back to client j, which then backpropagates through the local model following the chain rule: $\mathcal{W}_{c,j}^{(n,\tau+1)} = \mathcal{W}_{c,j}^{(n,\tau)} - \eta \nabla \mathcal{L}_{\mathcal{B}_j^\tau}(\mathcal{W}_{c,j}^{(n,\tau)})$.

One-step SGDM. When using SGDM (Polyak 1964; Liu, Gao, and Yin 2020) as the optimizer, the updating rule for

[1]Real-world split FL implementation partitions the full model into three parts: predictions $\hat{\mathbf{Y}}$ from the top-submodel remain local; thus loss calculation is performed by clients without label sharing.

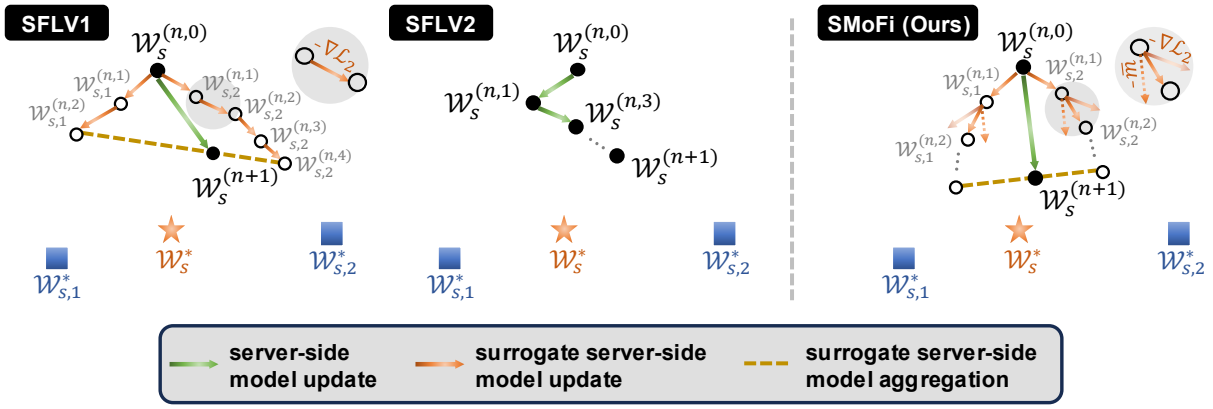

Figure 2: Comparison of server-side model updates $\mathcal{W}_s^{(n,0)} \mapsto \mathcal{W}_s^{n+1}$ in our SMoFi, and the state-of-the-art SFV1 and SFV2: In SFV1, the server updates surrogate server-side models $\mathcal{W}_{s,j}^{(n,\tau)}$ in parallel, and periodically aggregates them–e.g., after each local epoch as illustrated; In SFV2, the server sequentially interacts with clients to update the server-side model; SMoFi is akin to the SFV1 where the server updates surrogate models in parallel while introduces momentum alignment at each step τ by synchronizing the momentum buffers $\bar{m}$ across the server-side solvers. Such alignment helps the aggregated model converge toward the global optimum $\mathcal{W}_s^*$, rather than local optima $\mathcal{W}_{s,1}^*$ or $\mathcal{W}_{s,2}^*$.

the server-side submodel is reformulated as

$$m_{s,j}^{(n,\tau+1)} = \beta m_{s,j}^{(n,\tau)} + \nabla \mathcal{L}_{\mathcal{B}_j^\tau}(\mathcal{W}_{s,j}^{(n,\tau)}), \tag{2}$$

$$\mathcal{W}_{s,j}^{(n,\tau+1)} = \mathcal{W}_{s,j}^{(n,\tau)} - \eta m_{s,j}^{(n,\tau+1)}, \tag{3}$$

where β is the momentum coefficient, and $m_{s,j}^{(n,\tau)}$ is the momentum buffer retaining the gradients in the history steps. Similarly, each client updates the local submodel based on the gradients $\nabla \mathcal{L}_{\mathcal{B}_j^\tau}(\mathcal{W}_{c,j}^{(n,\tau)})$ and momentum buffer $m_{c,j}^{(n,\tau)}$.

2.2 SMoFi Design

Benefiting from our client-transparent design in SMoFi, we can focus on the server-side optimization. Broadly speaking, there are two main strategies for updating the server-side submodel in split FL: 1) *Parallel updating*, where the server updates the $|\mathcal{J}^n|$ surrogate submodels in parallel and periodically synchronizes the submodels by weighted averaging, as in SFLV1; and 2) *Sequential updating*, where the server retains a single submodel and update it by sequentially training with clients in the randomized order, as in SFLV2. As illustrated in Figure 2, the updates of surrogate server-side models in SFLV1 are independent across local steps. Even when using SGDM as the optimizer, each surrogate converges toward a diverged local optimum, as it minimizes objective $\{\mathcal{L}_{\mathcal{B}_j^\tau}(\cdot)\}_{j\in\mathcal{J}^n}$ inconsistent across clients due to heterogeneous data silos and varying local steps $\{T_j\}_{j\in\mathcal{J}^n}$ (Wang et al. 2020a). Besides, the sequential training in SFLV2 introduces severe latency in practical deployment. Considering both efficiency and efficacy, SMoFi adopts parallel updates of the surrogate server-side submodels, while imposing consistency constraints through momentum alignment to mitigate divergent convergence trajectories during collaborative training in split FL.

Momentum Alignment. In the *Parallel updating* framework, the server maintains $|\mathcal{J}^n|$ optimizers where each optimizer has a momentum buffer $m_{s,j}^{(n,\tau)}$ tracking the accumulated gradients of the j-th server-side model. Under objective inconsistency, these momentum buffers $\{m_{s,j}^{(n,\tau)}\}_{j\in\mathcal{J}^n}$ gradually diverge as training proceeds. The momentum buffers then further negatively affect gradient descent, causing the models across clients to converge in increasingly divergent directions. To address this, SMoFi aligns the momentum buffers across the server-side optimizers, and Equation 2 is then reformulated as

$$m_{s,j}^{(n,\tau+1)} = \beta \bar{m}_s^{(n,\tau)} + \nabla \mathcal{L}_{\mathcal{B}_j^\tau}(\mathcal{W}_{s,j}^{(n,\tau)}), \tag{4}$$

where $\bar{m}_s^{(n,\tau)}$ is the aligned momentum, synchronized after all server-side solvers perform gradient descent at each local step. In this way, each server-side model is updated based on stochastic gradients over its local mini-batch samples, combined with a unified momentum term (**line 7** in Algorithm 1).

SMoFi synchronizes the momentum buffer by weighted averaging the momentum in all server-side solvers at each local step. However, as training progresses, the number of active server-side solvers contributing to this average decreases because some surrogate models complete their training earlier than others. Specifically, the set of active clients at the τ-th step, denoted by $\mathcal{J}^{(n,\tau)} \subseteq \mathcal{J}^n$, includes only those clients for which $T_j \geqslant \tau$. Given that T_j varies across the clients due to non-IID local data, the size of the active set $|\mathcal{J}^{(n,\tau)}|$ tends to decrease over local steps. It leads to fewer solver momentum being averaged as training progresses, thereby diminishing the strength of the constraint from momentum alignment.

Staleness Factor. To maintain the effectiveness of constraint throughout all $max\{T_j\}_{j\in\mathcal{J}^n}$ local steps, SMoFi introduces a mechanism that leverages $\mathcal{H}^{(n)}$, a record of the momentum buffers $m_{s,j}^{(n,\tau+1)}$ at the final step of the client, i.e., when $\tau = T_j, \forall j \in \mathcal{J}^{(n,\tau)}$. At each local step, SMoFi aligns the

Algorithm 1: The Step-wise Momentum Fusion (SMoFi) in the n-th round

Input: Selected clients $\mathcal{J}^n$; Cut layer L; Numbers of local steps $\{T_j\}_{j\in\mathcal{J}^n}$; Current global model weights $\mathcal{W}^{(n-1)}$; Learning rate η; Momentum coefficient β.

Output: $\{\mathcal{W}_{s,j}^{(n,T_j)}\}_{j\in\mathcal{J}^n}$

$\{\mathcal{W}_{s,j}^{(n,0)}\}_{j\in\mathcal{J}^n} \leftarrow \mathcal{W}_{L:-1}^{(n-1)}$, $\mathcal{H}^n \leftarrow \emptyset$, $\bar{m}_s^{(n,0)} \leftarrow \mathbf{0}$

Server Executes:

for *step* $\tau = 0, 1, \cdots, max\{T_j\}_{j\in\mathcal{J}^n} - 1$ **do**

  for $j \in \mathcal{J}^{(n,\tau)}$ ***in parallel*** **do**

    `// Server-side Backpropagation`

    $\mathbf{A}_j^{(n,\tau)} \leftarrow$Collecting local activation

    $\nabla\mathcal{L}_{\mathcal{B}_j^\tau}(\mathcal{W}_{s,j}^{(n,\tau)}) \leftarrow$Stochastic gradients on $\mathbf{A}_j^{(n,\tau)}$

    $m_{s,j}^{(n,\tau+1)} \leftarrow \beta\,\bar{m}_s^{(n,\tau)} + \nabla\mathcal{L}_{\mathcal{B}_j^\tau}(\mathcal{W}_{s,j}^{(n,\tau)})$

    $\mathcal{W}_{s,j}^{(n,\tau+1)} \leftarrow \mathcal{W}_{s,j}^{(n,\tau)} - \eta m_{s,j}^{(n,\tau+1)}$

    $\nabla\mathcal{L}_{\mathcal{B}_j^\tau}(\mathcal{W}_{L\,s,j}^{(n,\tau)}) \leftarrow$Sending back to client j

    `// Historical Momentum Update`

    if $\tau = |T_j|$ **then**

      $\mathcal{H}^n \leftarrow$ Recording $m_{s,j}^{(n,\tau+1)}$

  `// Momentum Alignment`

  $\bar{m}_s^{(n,\tau+1)} \leftarrow$ Updating by Equation 5

momentum buffer for the next step (**line 12** in Algorithm 1) by averaging both the current momentum of optimizers and the historical state stored in $\mathcal{H}^{(n)}$:

$$\bar{m}_s^{(n,\tau+1)} = \frac{\sum_{j\in\mathcal{J}^{(n,\tau)}} m_{s,j}^{(n,\tau+1)} + \sum_{j\in\mathcal{H}^n} s_\alpha(\tau) m_{s,j}^{(n,|T_j|+1)}}{|\mathcal{J}^{(n,\tau)}|+|\mathcal{H}^n|}, \tag{5}$$

where s_α is the staleness of the historical momentum and $|\mathcal{J}^{(n,\tau)}|+|\mathcal{H}^n| = |\mathcal{J}^n|$, $\forall\tau \in [max\{T_j\}_{j\in\mathcal{J}^n}]$. The momentums in the current step are equally important to the momentum alignment as gradients are calculated on the same size of mini-batch across surrogate server-side solvers. For the historical ones, we employ a polynomial staleness factor (Xie, Koyejo, and Gupta 2019), satisfying

$$s_\alpha = (\tau - |T_j| + 1)^\alpha, \alpha < 0. \tag{6}$$

In this way, the number of momentum buffers that contribute to the synchronization remains constant $|\mathcal{J}^{(n)}|$ at each step.

The central server[2] first aggregates both client-side and server-side submodels at the end of the communication round: $\bar{\mathcal{W}}^n = \sum_{j\in\mathcal{J}^n} p_j \mathcal{W}_j^n$ where $\mathcal{W}_j^n = [\mathcal{W}_{c,j}^{(n,T_j)}, \mathcal{W}_{s,j}^{(n,T_j)}]$. Similar to work (Hsu, Qi, and Brown 2019), we update the global model with momentum: $\mathcal{W}^n = \mathcal{W}^{n-1} - m_g^n$. The global momentum buffer is updated following $m_g^n = \beta_g m_g^{n-1} + \mathcal{W}^{n-1} - \bar{\mathcal{W}}^n$ and β_g is the global momentum coefficient.

[2]In SFLV1/SFLV2, a fed server is used to aggregate the client-side submodels for the client-side global model. For simplicity, we let the central server performs the operations of the fed server.

3 Convergence Analysis

In this section, we provide the convergence analysis of SMoFi under the practical partial client participation. We start by stating assumptions commonly adopted in prior work (Acar et al. 2021; Rodio et al. 2023).

Assumption 3.1. (L-Smooth Objectives) The local objective $\mathcal{L}_j = \mathcal{L}_{\mathcal{D}_j}, \forall j \in \mathcal{J}$ is L-smooth ($L > 0$), i.e., $\forall\mathcal{W}, \mathcal{W}'$, it satisfies

$$\mathcal{L}_j(\mathcal{W}) \leq \mathcal{L}_j(\mathcal{W}') + \langle\nabla\mathcal{L}_j(\mathcal{W}'), \mathcal{W}-\mathcal{W}'\rangle + \frac{L}{2}\|\mathcal{W}-\mathcal{W}'\|_2^2. \tag{7}$$

Assumption 3.2. (μ-Strongly Convex Objectives) The local objectives $\mathcal{L}_{\mathcal{D}_1}, \cdots, \mathcal{L}_{\mathcal{D}_{|\mathcal{J}|}}$ are all convex, i.e., $\forall\mathcal{W}, \mathcal{W}'$, it satisfies

$$\mathcal{L}_j(\mathcal{W}) \geq \mathcal{L}_j(\mathcal{W}') + \langle\nabla\mathcal{L}_j(\mathcal{W}'), \mathcal{W}-\mathcal{W}'\rangle + \frac{\mu}{2}\|\mathcal{W}-\mathcal{W}'\|_2^2. \tag{8}$$

Assumption 3.3. (Unbiased Gradient and Bounded Variance) For mini-batch $\mathcal{B}_j^\tau$ uniformly sampled at random from local data of j-th client $\mathcal{D}_j$, the resulting stochastic gradient is unbiased to the gradient entire local dataset, that is, $\mathbb{E}_{\mathcal{B}_j^\tau\sim\mathcal{D}_j}[\nabla\mathcal{L}_{\mathcal{B}_j^\tau}(\mathcal{W})] = \nabla\mathcal{L}_j(\mathcal{W})$. Also, the variance of the stochastic gradient is bounded, i.e., $\forall\tau, j \in \mathcal{J}$, there exists σ satisfying

$$\mathbb{E}_{\mathcal{B}_j^\tau\sim\mathcal{D}_j}[\|\nabla\mathcal{L}_{\mathcal{B}_j^\tau}(\mathcal{W}) - \nabla\mathcal{L}_j(\mathcal{W})\|_2^2] \leq \sigma^2. \tag{9}$$

Assumption 3.4. (Bounded Gradients) The stochastic gradient is bounded; i.e., $\forall\tau, j \in \mathcal{J}$ there exists G satisfying $\mathbb{E}_{\mathcal{B}_j^\tau\sim\mathcal{D}_j}[\|\nabla\mathcal{L}_{\mathcal{B}_j^\tau}(\mathcal{W})\|_2^2] \leq G^2$.

The work (Han et al. 2024) offers convergence guarantees for both SFLV1 and SFLV2. The convergence analysis of SMoFi follows the idea of this work, as SMoFi introduces modifications to the SFLV1 workflow. Based on Assumptions 3.1- 3.4, we provide the convergence guarantee for SMoFi in Theorem 3.5.

Theorem 3.5. *Under the Assumptions 3.1, 3.2, 3.3, and 3.4, SMoFi has the similar convergence guarantees with SFLV1 with the momentum SGD as the optimization solver. Given the predefined communication rounds N, client participation rate θ, and a small enough learning rate $\eta^n = \frac{4}{\mu(\gamma+n)}$, the error between the global model at N-th round and the global optimum is bounded by*

$$\mathbb{E}[\mathcal{L}(\mathcal{W}^N)] - \mathcal{L}(\mathcal{W}^*) \leq \mathcal{O}(\frac{A}{(\gamma+N)}) + \mathcal{O}(\frac{B}{(\gamma+N)}) + \mathcal{O}(\frac{C}{(\gamma+N)}). \tag{10}$$

The A, B, C, and γ in the error bound follows $A = |\mathcal{J}|\sum_{j\in\mathcal{J}} p_j^2(2\sigma^2 + (1+\frac{1}{\theta})G^2)$, $B = \sum_{j\in\mathcal{J}} p_j(2\sigma^2 + G^2)$, $C = \|\mathcal{W}^0 - \mathcal{W}^*\|$, and $\gamma = 8L/\mu - 1$.

It indicates that the convergence bound of SMoFi achieves an order of $\mathcal{O}(1/N)$.

Setup	CIFAR-10/DIR$_{100}$(0.2)		CIFAR-100/DIR$_{100}$(0.2)		Tiny-ImageNet/DIR$_{200}$(0.2)	
Methods	**Acc. (%)**	**R/R↑**	**Acc. (%)**	**R/R↑**	**Acc. (%)**	**R/R↑**
FedAvg (McMahan et al. 2017b)	77.16±0.11	258/1.00×	48.10±0.36	183/1.00×	33.43±0.12	161/1.00×
+ FedAvgM (Hsu, Qi, and Brown 2019)	79.19±0.09	190/1.36×	50.28±0.26	126/1.45×	33.58±0.34	57/2.82×
+ SlowMo (Wang et al. 2019b)	76.54±0.06	177/1.46×	50.96±0.23	125/1.46×	33.82±0.29	44/3.66×
+ FedNAG (Yang et al. 2022)	78.24±0.43	170/1.52×	48.30±1.06	198/0.92×	30.94±0.44	335/0.48×
+ SMoFi	**81.82±0.61**	**56/4.61×**	**53.83±0.79**	**64/2.86×**	**39.73±0.05**	**16/10.06×**
FedProx (Li et al. 2020)	77.38±0.01	167/1.00×	48.67±0.06	175/1.00×	34.86±0.89	120/1.00×
+ FedAvgM (Hsu, Qi, and Brown 2019)	79.26±0.65	207/0.81×	50.45±0.13	111/1.58×	34.25±0.20	50/2.40×
+ SlowMo (Wang et al. 2019b)	76.63±0.09	210/0.80×	51.44±0.65	122/1.43×	33.67±0.22	85/1.41×
+ FedNAG (Yang et al. 2022)	77.59±0.80	200/0.84×	48.95±0.04	169/1.04×	31.20±0.21	284/0.42×
+ SMoFi	**81.99±0.37**	**54/3.09×**	**54.03±0.31**	**71/2.46×**	**40.79±0.13**	**13/9.23×**
FedNAR (Li et al. 2023)	77.21±0.06	255/1.00×	48.02±0.31	183/1.00×	33.37±0.34	164/1.00×
+ FedAvgM (Hsu, Qi, and Brown 2019)	79.21±0.47	190/1.34×	50.80±0.23	120/1.53×	33.63±0.76	105/1.56×
+ SlowMo (Wang et al. 2019b)	76.71±0.20	199/1.28×	51.94±0.18	123/1.49×	33.57±0.48	161/1.02×
+ FedNAG (Yang et al. 2022)	77.94±0.53	199/1.28×	48.51±0.24	175/1.05×	34.17±0.27	63/2.60×
+ SMoFi	**81.65±0.65**	**46/5.54×**	**53.72±0.42**	**67/2.73×**	**40.47±0.11**	**16/10.25×**

Table 1: Performance comparison between SMoFi and momentum-based counterparts across three baseline methods and three benchmark datasets. Methods denoted with + represent baselines combined with SMoFi or its counterparts. We report the average and standard deviation of Top-1 accuracy, the number of communication rounds (R) required to reach the target accuracy (i.e., 90% of the best global model accuracy by FedAvg), and the corresponding convergence speedup (R↑). All results are averaged over three trials, with bold font indicating the best performance for each setup.

Setup	CIFAR-10/DIR$_{100}$(0.2)			CIFAR-100/DIR$_{100}$(0.2)			Tiny-ImageNet/DIR$_{200}$(0.2)		
Methods	**Acc. (%)**	**R**	**T (h)**	**Acc. (%)**	**R**	**T (h)**	**Acc. (%)**	**R**	**T (h)**
SFLV1 ($\bar{\tau}=1$) (Thapa et al. 2022)	68.10±0.57	>1000	>551.62	38.43±0.06	>600	>172.66	21.81±0.98	>400	>578.95
SFLV1 ($\bar{\tau}=E$) (Thapa et al. 2022)	77.84±0.17	69	28.62	46.68±0.21	**40**	**10.23**	35.47±0.12	44	58.57
SFLV2 (Thapa et al. 2022)	79.42±0.04	278	144.50	53.64±0.51	143	42.58	34.72±0.95	310	527.48
MergeSFL (Liao et al. 2024)	79.47±0.09	76	**15.84**	50.16±0.20	53	11.22	34.74±0.55	118	152.25
SMoFi	**81.82±0.61**	**56**	29.02	**53.83±0.79**	64	18.46	**39.73±0.05**	**16**	**23.02**

Table 2: Performance comparison between SMoFi and split FL methods across three benchmark datasets. We report the average and standard deviation of Top-1 accuracy, along with the number of communication rounds (R) and wall-clock time (T) required to reach the target accuracy (i.e., 90% of the best global model accuracy by FedAvg).

4 Experimental Results

4.1 Experimental Setups

Datasets and Models. We experiment with three widely used image benchmarks: CIFAR10, CIFAR100 (Krizhevsky, Hinton et al. 2009), and Tiny-ImageNet (Le and Yang 2015). We implement the commonly used task model for each dataset: ResNet-18 (He et al. 2016) for CIFAR10 and CIFAR100; ResNet-34 for Tiny-ImageNet. To further validate the robustness of SMoFi in different task models, we also explore various models including VGG, MobileNetV2 (Sandler et al. 2018), and DenseNet (Huang et al. 2017). In the n-th communication round, the server randomly selects 20% clients $\mathcal{J}^n \subseteq \mathcal{J}$ for participation.

Heterogeneous Clients Setup. We simulate *data heterogeneity* in line with previous work (Hsu, Qi, and Brown 2019; Li et al. 2022), where the j-th client possesses data in the distribution $\boldsymbol{q_j} \in \mathbb{R}^c$ (c is the number of classes). We sample $\boldsymbol{q_j}$ from a Dirichlet distribution $\mathrm{Dir}_{|\mathcal{J}|}(\gamma)$ with a tunable concentration parameter $\gamma > 0$ that controls the level of heterogeneity. A smaller γ indicates a more heterogeneous distribution setting. Specifically, we set $\mathrm{Dir}_{100}(0.2)$ for CIFAR10 and CIFAR100, and $\mathrm{Dir}_{200}(0.2)$ for Tiny-ImageNet. We also simulate the *system heterogeneity* by varying computing power and communication bandwidth across clients, to assess the wall-clock time efficiency of various split FL frameworks. The heterogeneity profiles are from the public dataset AI benchmark (Ignatov et al. 2019) and MobiPerf (Huang et al. 2011).

Baselines. The baselines chosen for comparison are categorized into momentum-based methods and split FL methods. We compare SMoFi against momentum-based counterparts on three baselines: 1) the vanilla FL framework *FedAvg* (McMahan et al. 2017b); 2) *FedProx* (Li et al. 2020), adding a proximal term into the local objective function; and 3) *FedNAR* (Li et al. 2023) with self-adjusted weight decay. Building upon each baseline, we integrate three momentum-based methods including: 1) *FedAvgM* (Hsu, Qi, and Brown 2019), applying server momentum during global updates; 2) *SlowMo* (Wang et al. 2019b), periodically synchronizing and updating the local momentum across clients; and 3) *FedNAG* (Yang et al. 2022), implementing Nesterov Accelerated Gradient (NAG) (Sutskever et al. 2013; Bengio, Boulanger-Lewandowski, and Pascanu 2013) for local training with periodic momentum aggregation. Additionally, we include three split FL frameworks for evaluation: SFLV1, SFLV2 (Thapa et al. 2022), and MergeSFL (Liao et al.

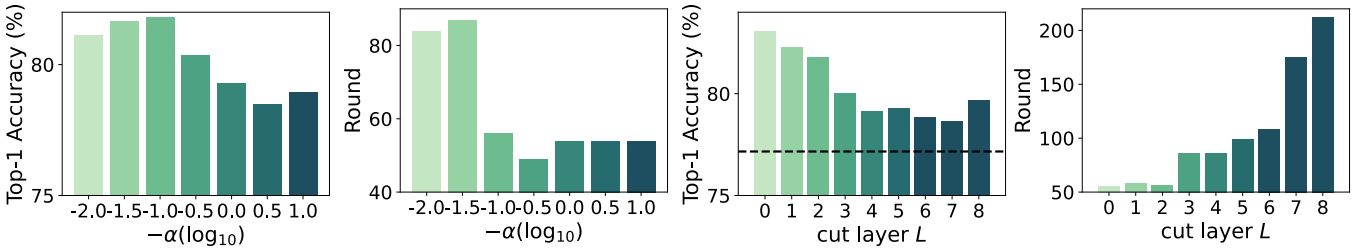

Figure 3: Sensitivity study of SMoFi under CIFAR10: (left two) accuracy and convergence under varying staleness factor α; (right two) performance under different cut layers L. The dashed line represents the accuracy of FedAvg under the same setting.

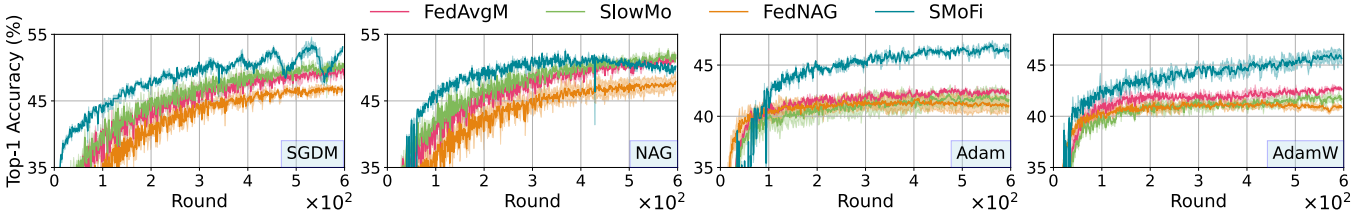

Figure 4: Learning curves of *FedAvg* integrated with SMoFi and its counterparts on the CIFAR100 using ResNet-18 under $\mathrm{Dir}_{100}(0.2)$ distribution. From left to right, we investigate different optimizers: SGD with momentum (SGDM), Nesterov Accelerated Gradient (NAG), Adaptive Moment Estimation (Adam), and Adam with decoupled weight decay (AdamW).

2024). For SFLV1, we investigate two server-side aggregation frequencies: after every training step ($\bar{\tau} = 1$) and after each local epoch ($\bar{\tau} = E$). Note that the performance of SFLV1 is equivalent to FedAvg when aggregating surrogate server-side models at each communication round.

For SMoFi, the staleness factor α is fixed at -0.1 across all settings, while β_g varies by task: 0.3 for CIFAR10, and 0.5 for CIFAR100 and Tiny-ImageNet. To ensure a fair comparison, we also fine-tune the hyperparameters for all baselines and counterpart methods.

Metrics. We run all the methods under each setup three times and report the average *Top-1 accuracy* within 1000, 600, and 400 communication rounds for CIFAR10, CIFAR100, and Tiny-ImageNet, respectively. To evaluate the convergence speed, we calculate the *round-to-accuracy* (**R**) performance, defined as the number of communication rounds required for the global model to reach the target accuracy–90% of the best performance achieved by FedAvg–across all settings. We also report the *time-to-accuracy* performance (**T**) from a system efficiency perspective when evaluating the split FL frameworks.

4.2 Performance Evaluation

SMoFi Effectiveness. SMoFi can be easily integrated into other split FL frameworks as a plug-in approach. SMoFi improves baseline performance by: 1) speeding up the global model convergence, thereby reducing the overall latency, and 2) further improving the global model performance.

Table 1 compares original baselines (i.e., FedAvg, FedProx, and FedNAR) and the baselines combined with SMoFi and momentum-based counterpart methods (denoted with **+**) on three datasets. In each setup, the full model in SMoFi is split at the shallow layers, with the server-side model holding the majority of the task model and the client-side model restricted to the bottom few layers. For instance, in the CIFAR10 task with ResNet-18, we fix the cut layer at $L = 2$ for all participating clients, where each client trains a small portion of the model comprising the input block and two residual blocks, while the server-side model includes the remaining 6 residual blocks and the output block. Such a model splitting strategy aligns with practical SFL deployment, where the server typically has significantly greater computational resources than edge devices (i.e., clients), allowing more training tasks to be allocated to the central server for better training efficiency gains.

Experimental results show that SMoFi consistently improves the performance of baselines across all benchmarks. The improvements are bigger in the complex classification tasks, particularly in complex tasks such as Tiny-ImageNet (200 classes) using the ResNet-34 model, making it highly suitable for scenarios favoring split training over conventional FL (Singh et al. 2019). Compared to the three momentum-based counterparts, SMoFi guarantees objective consistency in higher frequency by step-wise momentum alignment during model training, thereby further improving the accuracy of the global model. Moreover, SMoFi speeds up the convergence of the global model to the target accuracy by a large margin compared to baselines integrated with the counterpart methods. Unlike FedProx, which requires clients to report the local weight information–in addition to the activations of the cut layer–to the server when applied in a split training framework, SMoFi maintains the same level of privacy guarantee as SplitFed (Thapa et al. 2022) in terms of its client-side transparency without extra data reporting.

Table 2 shows the results for split FL frameworks with a constant cut layer at $L = 2$ for fair comparison. Performance for SFLV1 is sensitive to the aggregation frequency at the server side: step-wise aggregation underperforms, failing to

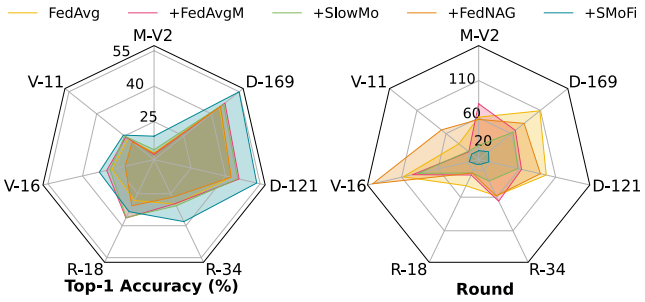

Figure 5: Robustness analysis on the Tiny-ImageNet dataset under $\mathrm{Dir}_{200}(0.2)$ distribution with various task-specific models: VGG (V), MobileNet (M), ResNet (R), and DenseNet (D). We report the best global model performance (left) and round-to-accuracy performance (right), within 150 communication rounds.

achieve the target accuracy within the given communication rounds, whereas epoch-wise aggregation yields better performance. SFLV2 outperforms SFLV1 in most cases, consistent with findings in work (Han et al. 2024), albeit at the cost of increased latency due to the sequential interaction between the server and clients. The MergeSFL offers fast convergence speed in terms of temporal space, even though it requires more communication rounds compared with our SMoFi, due to the adaptive batch size depending on device capabilities. However, SMoFi consistently provides superior performance in the long run.

Sensitivity Study. In Figure 3, we investigate the sensitivity of SMoFi to the staleness factor α (ranging from -0.01 to -10) and the cut layer L across the 8 residual blocks of ResNet-18. From the results, a smaller α, such as $\alpha = -1$, assigns lower weights to historical momentums during alignment, allowing the global model to converge faster, but can lead to suboptimal model performance. To trade off the model accuracy and convergence, we take $\alpha = -0.1$ as the default setting for all experiments on SMoFi. Moreover, performance gains from SMoFi are consistent across all cut layers L, with more significant benefits when the model is split at shallower layers (i.e., a smaller L). This aligns with typical split FL deployments, where a powerful server holds the majority of the model training task.

Robustness Analysis. We evaluate the robustness of momentum alignment in SMoFi from two perspectives: performance across different optimizers and model architectures. In Figure 4, SMoFi not only converges faster and yields higher accuracy than its counterparts under the SGDM optimizer (i.e., as used in Table 1 and Table 2), but also consistently outperforms them when using NAG (Sutskever et al. 2013), Adam (Kingma 2014), and AdamW (Loshchilov 2017). Note that, for local optimizers like Adam or AdamW, we periodically align both the first- and second-moment estimates on the server side. Figure 5 further investigates the robustness of SMoFi on Tiny-ImageNet across various model architectures. We report both the best global accuracy within 150 communication rounds and the round-to-accuracy performance, where the target accuracy is defined as the performance of FedAvg using the corresponding task model. Results show that SMoFi consistently improves both accuracy and convergence across all 7 types of models. The benefits of momentum alignment are more obvious for deeper or more complex model architectures, making SMoFi particularly suitable for split training scenarios involving resource-constrained clients and a powerful central server.

5 Related Work

Split Federated Learning. The concept of split learning was first introduced in works (Gupta and Raskar 2018; Vepakomma et al. 2018a) to split neural layers into two parts and assign them to the devices (with data resources) and the server (with supercomputing resources). SplitFed (Thapa et al. 2022) takes this a step further by integrating it into the FL framework Current research in split FL primarily aims to address two key challenges: reducing training latency and mitigating the risk of privacy leakage. CPSL (Wu et al. 2023) first partitions devices into several clusters. Training across the clusters follows the same sequential way as SL, while training devices within the cluster in parallel; FedGKT (He, Annavaram, and Avestimehr 2020) deploys a compact CNN (composing a lightweight feature extractor and a classifier) on device and the majority of the large model on the server; Works (Vepakomma et al. 2019; Abuadbba et al. 2020; Pasquini, Ateniese, and Bernaschi 2021) focus on reducing the privacy leakage in split learning and defending against adversarial attacks.

Data Heterogeneity. Existing methods for addressing the challenges of non-IID data silos can be broadly categorized into three types: 1) loss function modification such as works (Li and Zhan 2021; Gao et al. 2022; Li et al. 2020; Li, He, and Song 2021) reducing the inconsistency across clients by adding the penalty term into the local objective; 2) robustness aggregation by re-weighting the local updates (Wang et al. 2020a), alternatively, taking aggregation as optimization problem and applying various optimizer (Reddi et al. 2020); 3) adaptive hyperparameter setting, for instance, learning rate and weight decay of each local SGD solver (Li et al. 2023), or selection rate in each communication round (Balakrishnan et al. 2022). Among these approaches, works (Hsu, Qi, and Brown 2019; Wang et al. 2019b; Yang et al. 2022) apply momentum-based updating in either local training or central aggregation.

6 Conclusion

In this paper, we revisit the use of momentum to improve the performance of split FL on non-IID data silos. We propose SMoFi, a simple yet effective split FL framework that aligns the momentum of server-side solvers at each learning step. By leveraging the inherent client-server interaction in split FL, SMoFi imposes gradient-based constraints to mitigate training divergence. Experimental results show that SMoFi significantly improves both convergence speed and accuracy of the global model. Moreover, SMoFi requires zero modifications on clients, making it fully client-transparent–without additional communication overhead or privacy risk–thus offering a practical solution for real-world deployment.

Acknowledgments

This work is partly funded by the EU's Horizon Europe HarmonicAI project under the HORIZON-MSCA-2022-SE-01 scheme with grant agreement number 101131117. This work is also supported by the ICAI GENIUS lab of the research program ROBUST (project number KICH3.LTP.20.006), partly funded by the Dutch Research Council (NWO). We also thank the support provided by Samenwerkende Universitaire RekenFaciliteiten (SURF) with their Snellius infrastructure.

References

Abuadbba, S.; Kim, K.; Kim, M.; Thapa, C.; Camtepe, S. A.; Gao, Y.; Kim, H.; and Nepal, S. 2020. Can we use split learning on 1d cnn models for privacy preserving training? In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, 305–318.

Acar, D. A. E.; Zhao, Y.; Matas, R.; Mattina, M.; Whatmough, P.; and Saligrama, V. 2021. Federated Learning Based on Dynamic Regularization. In *International Conference on Learning Representations*.

Balakrishnan, R.; Li, T.; Zhou, T.; Himayat, N.; Smith, V.; and Bilmes, J. 2022. Diverse client selection for federated learning via submodular maximization. In *International Conference on Learning Representations*.

Bengio, Y.; Boulanger-Lewandowski, N.; and Pascanu, R. 2013. Advances in optimizing recurrent networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, 8624–8628. IEEE.

Bonawitz, K.; Eichner, H.; Grieskamp, W.; Huba, D.; Ingerman, A.; Ivanov, V.; Kiddon, C.; Konečnỳ, J.; Mazzocchi, S.; McMahan, B.; et al. 2019. Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1: 374–388.

Eshratifar, A. E.; Abrishami, M. S.; and Pedram, M. 2019. JointDNN: An efficient training and inference engine for intelligent mobile cloud computing services. *IEEE Transactions on Mobile Computing*, 20(2): 565–576.

Fortino, G.; and Trunfio, P. 2014. *Internet of things based on smart objects: Technology, middleware and applications*. Springer.

Gao, L.; Fu, H.; Li, L.; Chen, Y.; Xu, M.; and Xu, C.-Z. 2022. Feddc: Federated learning with non-iid data via local drift decoupling and correction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10112–10121.

Gao, Y.; Kim, M.; Abuadbba, S.; Kim, Y.; Thapa, C.; Kim, K.; Camtepe, S. A.; Kim, H.; and Nepal, S. 2020. End-to-end evaluation of federated learning and split learning for internet of things. *arXiv preprint arXiv:2003.13376*.

Gubbi, J.; Buyya, R.; Marusic, S.; and Palaniswami, M. 2013. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7): 1645–1660.

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

Han, P.; Huang, C.; Tian, G.; Tang, M.; and Liu, X. 2024. Convergence analysis of split federated learning on heterogeneous data. *arXiv preprint arXiv:2402.15166*.

He, C.; Annavaram, M.; and Avestimehr, S. 2020. Group knowledge transfer: Federated learning of large cnns at the edge. *Advances in Neural Information Processing Systems*, 33: 14068–14080.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hsu, T.-M. H.; Qi, H.; and Brown, M. 2019. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*.

Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4700–4708.

Huang, J.; Chen, C.; Pei, Y.; Wang, Z.; Qian, Z.; Qian, F.; Tiwana, B.; Xu, Q.; Mao, Z.; Zhang, M.; et al. 2011. Mobiperf: Mobile network measurement system. *Technical Report. University of Michigan and Microsoft Research*.

Ignatov, A.; Timofte, R.; Kulik, A.; Yang, S.; Wang, K.; Baum, F.; Wu, M.; Xu, L.; and Van Gool, L. 2019. Ai benchmark: All about deep learning on smartphones in 2019. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 3617–3635. IEEE.

Kingma, D. P. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Konečnỳ, J.; McMahan, H. B.; Yu, F. X.; Richtárik, P.; Suresh, A. T.; and Bacon, D. 2016. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

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

Li, J.; Gao, H.; Lv, T.; and Lu, Y. 2018. Deep reinforcement learning based computation offloading and resource allocation for MEC. In *2018 IEEE wireless communications and networking conference (WCNC)*, 1–6. IEEE.

Li, J.; Li, A.; Tian, C.; Ho, Q.; Xing, E.; and Wang, H. 2023. FedNAR: federated optimization with normalized annealing regularization. *Advances in Neural Information Processing Systems*, 36: 74753–74763.

Li, Q.; Diao, Y.; Chen, Q.; and He, B. 2022. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th international conference on data engineering (ICDE)*, 965–978. IEEE.

Li, Q.; He, B.; and Song, D. 2021. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10713–10722.

Li, T.; Sahu, A. K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; and Smith, V. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2: 429–450.

Li, X.-C.; and Zhan, D.-C. 2021. Fedrs: Federated learning with restricted softmax for label distribution non-iid data. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 995–1005.

Liao, Y.; Xu, Y.; Xu, H.; Wang, L.; Yao, Z.; and Qiao, C. 2024. Mergesfl: Split federated learning with feature merging and batch size regulation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, 2054–2067. IEEE.

Liu, Y.; Gao, Y.; and Yin, W. 2020. An improved analysis of stochastic gradient descent with momentum. *Advances in Neural Information Processing Systems*, 33: 18261–18271.

Loshchilov, I. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017a. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017b. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, volume 54, 1273–1282. PMLR.

Pasquini, D.; Ateniese, G.; and Bernaschi, M. 2021. Unleashing the tiger: Inference attacks on split learning. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2113–2129.

Polyak, B. T. 1964. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5): 1–17.

Reddi, S.; Charles, Z.; Zaheer, M.; Garrett, Z.; Rush, K.; Konečnỳ, J.; Kumar, S.; and McMahan, H. B. 2020. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*.

Rodio, A.; Faticanti, F.; Marfoq, O.; Neglia, G.; and Leonardi, E. 2023. Federated learning under heterogeneous and correlated client availability. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*, 1–10. IEEE.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4510–4520.

Shi, C.; Li, J.; Zhao, H.; Guo, D.; and Chang, Y. 2025. FedLWS: Federated Learning with Adaptive Layer-wise Weight Shrinking. *arXiv preprint arXiv:2503.15111*.

Singh, A.; Vepakomma, P.; Gupta, O.; and Raskar, R. 2019. Detailed comparison of communication efficiency of split learning and federated learning. *arXiv preprint arXiv:1909.09145*.

Sutskever, I.; Martens, J.; Dahl, G.; and Hinton, G. 2013. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, 1139–1147. PMLR.

Thapa, C.; Arachchige, P. C. M.; Camtepe, S.; and Sun, L. 2022. Splitfed: When federated learning meets split learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 8485–8493.

Vepakomma, P.; Gupta, O.; Dubey, A.; and Raskar, R. 2019. Reducing leakage in distributed deep learning for sensitive health data. *arXiv preprint arXiv:1812.00564*, 2.

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

Vepakomma, P.; Swedish, T.; Raskar, R.; Gupta, O.; and Dubey, A. 2018b. No peek: A survey of private distributed deep learning. *arXiv preprint arXiv:1812.03288*.

Verbraeken, J.; Wolting, M.; Katzy, J.; Kloppenburg, J.; Verbelen, T.; and Rellermeyer, J. S. 2020. A survey on distributed machine learning. *Acm computing surveys (csur)*, 53(2): 1–33.

Wang, J.; Liu, Q.; Liang, H.; Joshi, G.; and Poor, H. V. 2020a. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33: 7611–7623.

Wang, J.; Tantia, V.; Ballas, N.; and Rabbat, M. 2019a. SlowMo: Improving Communication-Efficient Distributed SGD with Slow Momentum. In *International Conference on Learning Representations*.

Wang, J.; Tantia, V.; Ballas, N.; and Rabbat, M. 2019b. Slowmo: Improving communication-efficient distributed sgd with slow momentum. *arXiv preprint arXiv:1910.00643*.

Wang, S.; Zhang, X.; Uchiyama, H.; and Matsuda, H. 2021. HiveMind: Towards cellular native machine learning model splitting. *IEEE Journal on Selected Areas in Communications*, 40(2): 626–640.

Wang, X.; Han, Y.; Leung, V. C.; Niyato, D.; Yan, X.; and Chen, X. 2020b. Convergence of edge computing and deep learning: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 22(2): 869–904.

Wu, W.; Li, M.; Qu, K.; Zhou, C.; Shen, X.; Zhuang, W.; Li, X.; and Shi, W. 2023. Split learning over wireless networks: Parallel design and resource management. *IEEE Journal on Selected Areas in Communications*, 41(4): 1051–1066.

Xie, C.; Koyejo, S.; and Gupta, I. 2019. Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934*.

Yang, Z.; Bao, W.; Yuan, D.; Tran, N. H.; and Zomaya, A. Y. 2022. Federated learning with nesterov accelerated gradient. *IEEE Transactions on Parallel and Distributed Systems*, 33(12): 4863–4873.