MultiSFL: Towards Accurate Split Federated Learning via Multi-Model Aggregation and Knowledge Replay

Zeke Xia[1*], Ming Hu[2*], Dengke Yan[1], Ruixuan Liu[1], Anran Li[3], Xiaofei Xie[2], Mingsong Chen[1†]

[1]MoE Engineering Research Center of SW/HW Co-Design Tech. and App., East China Normal University, China
[2]School of Computing and Information Systems, Singapore Management University, Singapore
[3]Department of Biomedical Informatics & Data Science, School of Medicine, Yale University
51255902180@stu.ecnu.edu.cn, hu.ming.work@gmail.com, {51265902053, 51255902059}@stu.ecnu.edu.cn, anran.li@ntu.edu.sg, xfxie@smu.edu.sg, mschen@sei.ecnu.edu.cn

Abstract

Although Split Federated Learning (SFL) effectively enables knowledge sharing among resource-constrained clients, it suffers from low training performance due to the neglect of data heterogeneity and catastrophic forgetting problems. To address these issues, we propose a novel SFL approach named MultiSFL, which adopts i) an effective multi-model aggregation mechanism to alleviate gradient divergence caused by heterogeneous data and ii) a novel knowledge replay strategy to deal with the catastrophic forgetting problem. MultiSFL adopts two servers (i.e., the fed server and main server) to maintain multiple branch models for local training and an aggregated master model for knowledge sharing among branch models. To mitigate catastrophic forgetting, the main server of MultiSFL selects multiple assistant devices for knowledge replay according to the training data distribution of each full branch model. Experimental results obtained from various non-IID and IID scenarios demonstrate that MultiSFL significantly outperforms conventional SFL methods by up to a 23.25% test accuracy improvement.

Introduction

Due to the merits of knowledge sharing among devices without compromising their data privacy, Federated Learning (FL) (McMahan et al. 2017; Li et al. 2023a; Wu et al. 2023b; Huang et al. 2024; Qi et al. 2024) becomes increasingly popular in the design of Artificial Intelligence of Things (AIoT) systems (Hu et al. 2023a; Wang et al. 2024a; Zhang et al. 2020). However, existing FL methods often suffer from the problem of low training performance when deployed in large-scale AIoT applications. This is mainly because the varying capabilities (e.g., memory, computing power) of heterogeneous AIoT devices may inevitably result in limited sizes of their hosting Deep Neural Network (DNN) models. According to the Wooden Bucket Theory, when dealing with resource-constrained AIoT devices, the weakest device determines the overall FL training performance. To address this issue, Split Federated Learning (SFL) (Thapa et al. 2022) has been investigated to accommodate various large-scale AIoT applications by enabling the co-training of large models across heterogeneous devices and cloud servers. Unlike conventional FL methods, SFL divides each participating DNN model into two parts, i.e., the client-side portion and the server-side portion, and places them on two cloud servers, i.e., the fed server and the main server, respectively. By decoupling the training efforts of client-side portions and server-side portions, the resource requirements imposed on clients are greatly alleviated. Meanwhile, by synchronizing the training between the client-side portions and server-side portions based on local data features, the test accuracy degradation caused by the trained models is negligible.

Although SFL is good at dealing with resource-constrained scenarios, it still suffers from the problem of ❶ data heterogeneity and ❷ catastrophic forgetting (Shao and Feng 2022). Specifically, due to the non-Independent and Identically Distributed (non-IID) (Sattler et al. 2020; Qi et al. 2023; Huang et al. 2023; Wu et al. 2023a) data among AIoT devices, SFL suffers from the "gradient divergence" problem (Karimireddy et al. 2020), which results in performance degradation of global models. Meanwhile, since the server only selects partial clients for model training in each SFL round, models tend to forget what they have learned in previous SFL training rounds, causing the notorious catastrophic forgetting problem (Shao and Feng 2022). To alleviate the impacts of such problems, various FL variants resort to knowledge distillation (Zhu, Hong, and Zhou 2021; Lin et al. 2020; Meng et al. 2024; Wang et al. 2023), client selection (Chen et al. 2020), global control variable (Li et al. 2020a; Karimireddy et al. 2020; Wang et al. 2024b) or multi-model paradigm (Hu et al. 2024c, 2023b, 2024b; Gao et al. 2024) mechanisms to mitigate the performance degradation problem caused by non-IID data, while many of them adopt model correction (Luo et al. 2023) and data augmentation (Xu et al. 2022) strategies to solve the catastrophic forgetting problem. However, most of them assume that the local training is performed based on a whole model, strongly limits their usage in SFL. Therefore, *how to address the gradient divergence and catastrophic forgetting problems is becoming an urgent issue for SFL.*

As a state-of-the-art FL training paradigm, the Multi-Model-based FL (MMFL) (Hu et al. 2024c, 2023b) is promising in dealing with the gradient divergence problem by adopting multiple homogeneous branch models rather

*These authors contributed equally.
†Corresponding author.

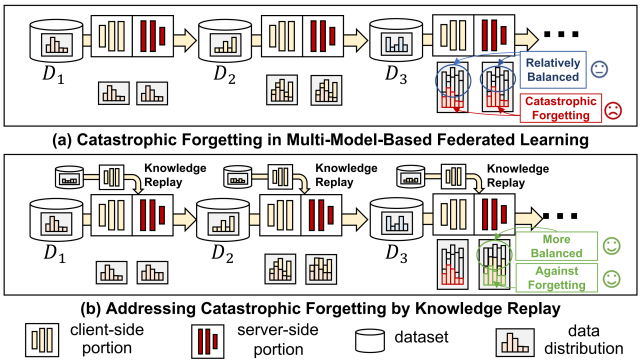


Figure 1: A motivating example of SFL.

than using a single global model for local training. Specifically, each branch model selects the least similar model as the collaborative model for aggregation. To alleviate the gradient divergence, the collaborative model is assigned a relatively low weight for aggregation. In addition, for each branch model, despite the fact that the distribution of data from each device is non-IID, the overall data distribution gradually converges towards balance through continuous training. Intuitively, inspired by multi-model-based FL, the main server and the fed server can maintain multiple portions and assign an index to match the corresponding server-side and client-side portions to achieve training of multiple branch models. Although the multi-model-based FL paradigm effectively alleviates the data heterogeneity problem, it still encounters the catastrophic forgetting problem. Figure 1(a) presents an example of the training process of a branch model. As shown in Figure 1(a), in non-IID scenarios, the branch model forgets the knowledge learned by D_1 after two training rounds and both D_2 and D_3 lack the data of the second category, inevitably resulting in a degradation of the branch model classification performance for the second category. Unlike conventional FL, in SFL, although the main server cannot access the full model, it can access training data labels, which can be used to identify which categories of knowledge are missing for each branch model. Intuitively, as shown in Figure 1(b), to alleviate catastrophic forgetting, the cloud server can request inactivated clients to upload a small number of missing category features to help train the specific server-side portion.

Inspired by the above example, we propose a new SFL approach named *MultiSFL* based on the MMFL-style training coupled with a novel knowledge-replay mechanism, which can effectively address the gradient divergence and catastrophic forgetting problems. Unlike existing SFL methods, MultiSFL uses the fed server and main server to maintain the server- and client-side portions of multiple branch models for SFL training. To achieve knowledge sharing among such branch models, each server aggregates all the models to generate a master model and aggregates the master model with each branch model, where the master model is assigned a small aggregation weight. To address catastrophic forgetting, the main server selects inactivated clients as assistant clients according to their data distributions, and requests sampling features from these assistant clients for the joint training of server-side branch models. To improve training performance and save communication overhead, our method adaptively adjusts the sampling proportion based on the critical learning period. This paper makes the following three major contributions:

- We design a novel MMFL-based SFL framework, which resorts to the random traversals of branch models to alleviate the gradient divergence problem.
- We propose an adaptive knowledge replay strategy that explores the most contributing knowledge for branch models to alleviate catastrophic forgetting.
- Comprehensive experiments on various well-known datasets and models show the superiority of MultiSFL over state-of-the-art (SOTA) FL methods for both IID and non-IID scenarios.

Preliminaries and Related Work

Split Federated Learning (SFL) (Thapa et al. 2022) combines the advantages of FL and Split Learning (SL). In SFL, the complete model is divided into two parts: the client-side model portion and the server-side model portion. Each client communicates with the main server and the fed server. In each SFL round, clients interact with the main server in parallel to execute the SL process. Subsequently, clients send their updated client-side portion to the fed server for aggregation. The fed server aggregates all client models and synchronizes the aggregated model with all clients. However, similar to FL, SFL still experiences the problem of low inference accuracy. The objective of SFL is to minimize the loss function over the collection of training data at N clients, i.e.,

$$\min_{w} F(w) = \sum_{k=1}^{N} \frac{|D_k|}{|D|} F_k(w), \tag{1}$$

where $w = w^c \oplus w^s$ is the combination of client-side model w^c and server-side model w^s, N is the number of clients that participate in local training, D_k is the dataset of the k^{th} client, $F_k(w) = \frac{1}{|D_k|}\sum_{j \in D_k} f_j(w)$ is the loss empirical objective over the data samples at client k.

Catastrophic Forgetting occurs specifically when the neural network is trained sequentially on multiple tasks. In this case, the optimal parameters for the current task might perform poorly on the objectives of previous tasks. Many algorithms in FL have been proposed to alleviate the forgetting issue. GradMA (Luo et al. 2023) uses historical gradient information on both the device and server sides and corrects the global gradient through quadratic planning, effectively improving the accuracy of the global model. FedReg (Xu et al. 2022) reduces knowledge forgetting by using generated pseudo data to regularize local training parameters and suppress potential conflicts with knowledge in local data introduced by pseudo data by using perturbed data. However, such methods cannot be applied to SFL due to the stringent requirements of the complete model. To our best knowledge, MultiSFL represents the first innovative approach that employs multi-model training and knowledge replay in SFL to enhance both model accuracy and training stability.

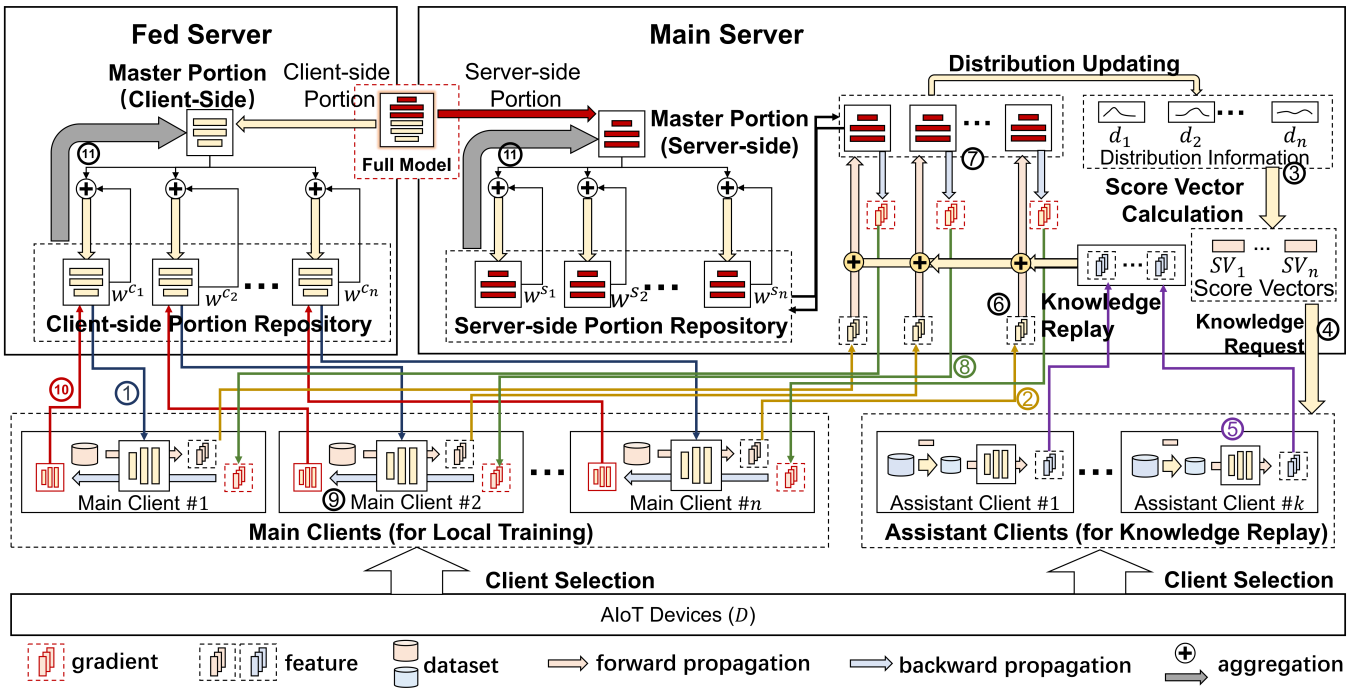


Figure 2: Framework and workflow of our approach.

Our MultiSFL Approach

Overview

Figure 2 illustrates the framework and workflow of our approach. Similarly to conventional SFL, MultiSFL consists of two servers (i.e., the fed server and the main server) and multiple AIoT devices. In MultiSFL, the full model is divided into two portions, i.e., the client-side portion and the server-side portion. As shown in Figure 2, to address the gradient divergence problem caused by non-IID data, MultiSFL adopts multiple branch models for local training. To achieve multi-model-based FL training, the fed-sever maintains a client-side portion repository to store client-side branch models, and the main server maintains a server-side portion repository to store server-side branch models. Note that a server-side and client-side portion corresponding to a branch model are assigned the same index in their repositories. To enable knowledge sharing among branch models, each server maintains a master model, which is aggregated by all the branch models in their repository.

To alleviate catastrophic forgetting, in each SFL training round, the main server calculates a score vector for each branch model, where the number of elements of the vector is equal to the number of classification categories. The value of each element is calculated according to the accumulated training data distribution, where the more recent training data distribution has a greater impact on the score vector calculation. The main server selects multiple inactivated clients as assistant clients and requests features of specific categories from assistant clients according to the score vectors to assist in the training of the server-side branch models.

As shown in Figure 2, the workflow of the training process for each intermediate model in MultiSFL includes eleven steps as follows: **Step 1 (Model Dispatching):** The fed server randomly selects n clients as main clients and dispatches the client-side branch models to main clients for local training. **Step 2 (Feature Uploading):** Each main client uses its data to perform the forward propagation process and uploads the output features of its client-side branch model to the main server. **Step 3 (Score Vector Calculation):** For each branch model, the main server calculates a score vector according to its current and historical training data distribution. **Step 4 (Knowledge Request):** Then the main server randomly selects k inactivated clients as assistant clients, dispatches a calculated score vector to an assistant client, and requests the fed server to dispatch client-side branch models to the corresponding assistant clients. **Step 5 (Knowledge Extraction):** Assistant clients select samples from their data according to the received score vector and use the selected samples to perform forward propagation. Then, they upload the output features to the main server. Note that if the main server does not collect enough features, it will repeat Steps 4-5 to request more knowledge. **Step 6 (Knowledge Replay):** The main server inputs the features collected from the main and assistant clients together into the corresponding server-side branch models for the forward propagation process and calculates the losses. **Step 7 (Server Model Backward):** The main server performs the backward propagation process to update the corresponding server-side branch models and obtains the gradient of the features uploaded by the main clients. **Step 8 (Gradients Dispatching):** The main server sends the gradient to the corresponding main client. **Step 9 (Client Model Back-**

ward): Each main client updates the client-side portion with the received gradient. **Step 10 (Model Uploading):** Each main client uploads its client-side portion to the fed server. **Step 11 (Model Repository Updating):** The fed server aggregates all client-side branch models to generate a new client-side master model and then updates each client-side branch model by aggregating it with the client-side master model. Similarly, the main server aggregates all the server-side branch models to generate a new server-side master model and then updates each server-side branch model by aggregating it with the new server-side master model.

Please note that during MultiSFL training, a client needs to send the labels, features, and sample quantities from clients to the server, which is exactly the same way as SFL. In other words, our approach is as secure as the conventional SFL, since it does not require clients to send additional data.

Implementation

Algorithm 1 details the implementation of our MultiSFL approach. Assume that there are n activated clients as the main clients participating in each SFL communication round. Line 1 initializes the client-side branch model repository W_c and its corresponding server-side branch model repository W_s. Line 2 initializes the model cumulative data distribution and sampling proportion. Lines 3-25 present the overall SFL training process. In line 4, we randomly select n clients as the main clients for local training. Lines 7-22 present the cooperative training process of each client-side branch model and its corresponding server-side model. In line 7, the main client $S[i]$ uses its local data to propagate forward and gain the intermediate feature f_c. Line 8 indicates that the main server updates the data distribution of $w_r^{s_i}$. In line 9, the main server calculates the score vector sv^i based on the historical data distribution. In line 10, the main server calculates the number of data to sample q^i for each data class based on sv^i. Lines 11-17 present the process of sampling features. Firstly, the main server initializes the total sample supply l' and the total sampled feature f_h (Line 11). The main server randomly selects an idle client dev_a that can satisfy or partially satisfy the current knowledge request as an assistant client, and the fed server sends the client-side model $w_r^{c_i}$ to dev_a (Line 13). After receiving the $w_r^{c_i}$, dev_a samples its local data and sends the feature of the sampled data using $w_r^{c_i}$ according to the current knowledge request $q^i - l'$ (Line 14). After receiving the feature f_a, the main server updates the total sample supply l' the total sampled feature f_h (Lines 15-16). In line 18, the main server puts f_c and f_h together to get f_s. In lines 19-20, the main server uses f_s to update the server-side branch model, and the main server sends the gradient of f_c back to the client $S[i]$, allowing the client $S[i]$ to update the client-side branch model $w_r^{c_i}$. Line 22 indicates that the fed server adjusts the sample proportion for the next round. In line 23, the fed server and the main server aggregate all client-side branch models and server-side branch models, respectively, to update w_{r+1}^c and w_{r+1}^s. In lines 24-26, the fed server and the main server perform aggregation between the models in the model repositories and the global model and update the model repositories.

Algorithm 1: Our MultiSFL approach

Input: i) R, maximum number of rounds; ii) C, client set;
Output: w^g, the global model
MultiSFL(R, C) **begin**

1: $(W_c, W_s) \leftarrow ([w_0^{c_1}, w_0^{c_2}, ..., w_0^{c_n}], [w_0^{s_1}, w_0^{s_2}, ..., w_0^{s_n}])$
2: $(L, p_0) \leftarrow ([L_1, L_2, ..., L_n], 0.01)$
3: **for** $r \leftarrow$ 0, 1, . . ., R-1 **do**
4: $S \leftarrow$Randomly select n devices from C
5: /* parallel for */
6: **for** $i \leftarrow$ 1, 2, . . ., n **do**
7: $(f_c, l_{S[i]}) \leftarrow w_r^{c_i}(D_{S[i]})$
8: $L_i[r] \leftarrow l_{S[i]}$
9: $sv^i \leftarrow$SVCaculate(L_i)
10: $q^i \leftarrow$KnowledgeRequest(sv^i, p_r)
11: $(l', f_h) \leftarrow (Zero(q^i), \{\})$;
12: **while** $l' < q^i$ **do**
13: $dev_a \leftarrow$Randomly select a device from $C - S$
14: $(f_a, l_a) \leftarrow$KnowledgeExtr($w_r^{c_i}$,D_{dev_a},$q^i - l'$)
15: $l' \leftarrow l' + l_a$
16: $f_h \leftarrow f_h \oplus f_a$
17: **end while**
18: $f_s \leftarrow f_c \oplus f_h$
19: $y \leftarrow w_r^{s_i}(f_s)$
20: $(w_r^{s_i}, w_r^{c_i}) \leftarrow (w_r^{s_i} - \eta\nabla(\ell(y)), w_r^{c_i} - \eta\nabla(\frac{\partial\ell(y)}{\partial f_c}))$
21: **end for**
22: $p_{r+1} \leftarrow \frac{FGN(r)-FGN(r-1)}{FGN(r-1)} \times p_r$
23: $(w_{r+1}^c, w_{r+1}^s) \leftarrow (\sum_{i=1}^n \frac{w_{r+1}^{c_i}}{n}, \sum_{i=1}^n \frac{w_{r+1}^{s_i}}{n})$
24: **for** $i \leftarrow$ 1, 2, . . ., n **do**
25: $(w_{r+1}^{c_i}, w_{r+1}^{s_i}) \leftarrow (\frac{w_{r+1}^{c_i} + \alpha w_{r+1}^c}{1+\alpha}, \frac{w_{r+1}^{s_i} + \alpha w_{r+1}^s}{1+\alpha})$
26: $(W_c[i], W_s[i]) \leftarrow (w_{r+1}^{c_i}, w_{r+1}^{s_i})$
27: **end for**
28: **end for**
29: $w^g \leftarrow w_R^c \oplus w_R^s$
30: **Return** w^g

Knowledge Replay Strategy

To solve catastrophic forgetting, MultiSFL adopts a knowledge replay strategy, which requests inactivated clients to upload features of categories that are less distributed in recent training data for each full branch model and uses these features to train the corresponding server-side branch models. As shown in Algorithm 1, our knowledge replay strategy consists of three key processes, i.e., score vector calculation, knowledge request, and knowledge extraction.

Score Vector Calculation (SVCalculate(·)). This process evaluates the knowledge of each category learned by full branch model w^i, where the number of elements in the vector equals the number of categories, and the model trained by more data of a specific category is assigned a larger score value for the corresponding category. To mitigate catastrophic forgetting, current training data is weighted more heavily for score vector calculation, while historical data is weighted less. For each branch model w_r^i, we calculate the score value sv^i as follows:

$$sv^i = \frac{\sum_{j=0}^{r} \gamma^{r-j} L_i[j]}{\sum_{j=0}^{r} \gamma^{r-j}}, \tag{2}$$

where γ is the decay factor smaller than 0, and l_j is the local

data distribution of w_r^i in the r-th round of training. Note that since our approach performs knowledge replay only on the server-side branch model w^{s_i}, there is no need to consider knowledge replay when calculating sv^i.

Knowledge Request (KnowledgeRequest(·)). This process calculates the number of data samples of each category that need to be selected to replay knowledge according to the score vector. We first calculate the average score value of the score vector. The main server prefers to select the data of a category when its corresponding score value is larger than the average score. For each data category c, we calculate its priority value $prior_c^i$ as follows:

$$prior_c^i = \max(0, mean(sv^i) - sv^i[c]). \tag{3}$$

Note that the less accumulated data of data class c, the higher its priority. Using the priority values, we calculate the number of sampling data for the data category c as follows:

$$q_c^i = \frac{|D_i| \times p_r \times prior_c^i}{\sum_c prior_c^i}, \tag{4}$$

where p_r is the sampling proportion of round r, and $|D_i|$ is the local data size of the main client selected for model w_r^i.

Knowledge Extraction (KnowledgeExtr(·)). After receiving $w_r^{c_i}$, the assistant client randomly selects its local data for forward propagation according to the calculated number of sampling data and uploads the intermediate features to the main server. In addition, MultiSFL repeats the knowledge extraction process until the request calculated by the Knowledge Request process is fulfilled. Please note that the communication overhead caused by transmitting intermediate features in SFL is much greater than that caused by the transmitting model. Therefore, the additional communication overhead caused by sending the model to assistant clients is negligible.

Adaptive Adjustment of Sampling Proportion

In our knowledge replay strategy, the sampling proportion p_r is a key parameter that significantly affects the performance of MultiSFL. Specifically, with an increasing sampling proportion p_r, the main server requests more data to train the server-side branch model, which can substantially improve the accuracy of each branch model, especially in non-IID scenarios. However, a larger value of p_r inevitably results in a significant increase in communication overhead. To balance the accuracy of the model and communication overhead, we propose a dynamic sampling proportion adjustment mechanism. Previous work (Yan, Wang, and Li 2022) observed that when the curvature of the loss landscape at a particular point w is large, model training is in a Critical Learning Period (CLP) (Yan, Wang, and Li 2022). Inspired by this observation, we prefer to select more data in the CLP and select fewer data at rather training rounds. We use the Federated Gradient Norm (FGN) to approximate the curvature of the loss landscape at a particular point w during training (Yan et al. 2023). The FGN of round r can be defined as follows:

$$FGN(r) = \frac{\sum_{i=1}^{n} -\eta \left\| g(w_r^i, \xi) \right\|^2}{n}, \tag{5}$$

where $g(w_r^i, \xi)$ denotes the gradient of the loss function evaluated on ξ. Based on FGN, we adjust the sampling proportion p_{t+1} for the next round as follows:

$$p_{r+1} = \frac{FGN(r) - FGN(r-1)}{FGN(r-1)} \times p_r. \tag{6}$$

Convergence Analysis

Similarly to (Luo et al. 2023; Li et al. 2020b; Ma et al. 2021), our analysis relies on the following assumptions:

Assumption 1. *f_i is L-smooth satisfying $f_i(w) \leq f_i(w') + (w-w')^T \nabla f_i(w') + \frac{L}{2}||w-w'||_2^2$, where $i \in \{1, 2, \cdots, N\}$.*

Assumption 2. *f_i is μ-convex satisfying $f_i(w) \geq f_i(w') + (w-w')^T \nabla f_i(w') + \frac{\mu}{2}||w-w'||_2^2$, where $i \in \{1, 2, \cdots, N\}$ and $\mu \geq 0$.*

Assumption 3. *The variance of stochastic gradients is upper bounded by σ^2 and the expectation of squared norm of stochastic gradients is upper bounded by G^2, i.e., $\mathbb{E}||\nabla f_i(w;\xi) - \nabla f_i(w)||^2 \leq \sigma^2$, $\mathbb{E}||\nabla f_i(w;\xi)||^2 \leq G^2$, where ξ is a data batch of the i^{th} client in the t^{th} SFL round.*

According to our aggregation strategy, we have:

Lemma 1. *Let $w_r^i = \alpha v_r^i + (1-\alpha)\overline{v}_r$, $\alpha \in [0,1]$, and $\overline{w}_r = \sum_{i=1}^{N} w_r^i$. We have*

$$||\overline{w}_r - w^\star||^2 \leq \frac{1}{N}\sum_{i=1}^{N}||w_r^i - w^\star||^2 \leq \frac{1}{N}\sum_{i=1}^{N}||v_r^i - w^\star||^2,$$

where $w^\star$ is the optimal parameters for the global loss function $F(\cdot)$.

Theorem 1. *Assume that the server performs model aggregation after E rounds of SGD,i.e., E rounds of SGD are performed in each SFL round, and the whole training consists of r rounds. Let $t = r \times E$ be the current number of SGD rounds, and $\eta_t = \frac{2}{\mu(t+\lambda)}$ be the learning rate. We have:*

$$\mathbb{E}[F(\overline{w}_t)] - F^\star \leq \frac{L}{2\mu(t+\lambda)}\left[\frac{4B}{\mu} + \frac{\mu(\lambda+1)}{2}\Delta_1\right], \tag{7}$$

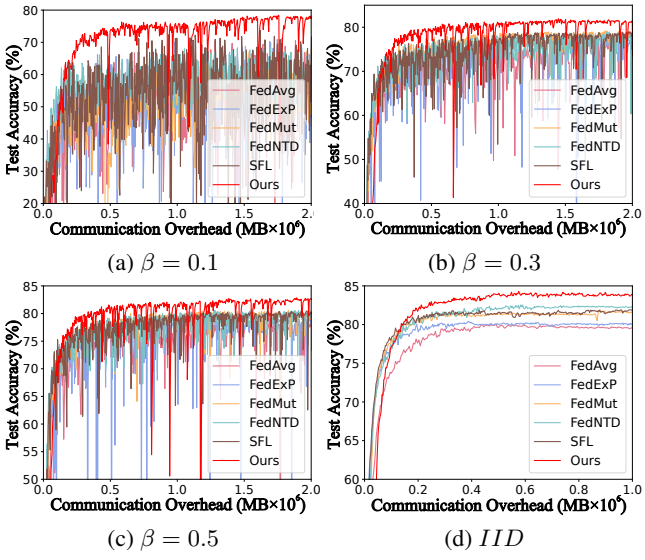
where $B = 10L\Gamma + 4(E-1)^2G^2$.

Experimental Results

To evaluate the effectiveness of our approach, we implemented MultiSFL using the PyTorch framework (Paszke et al. 2019) and conducted a comparative analysis with classical FL, i.e., FedAvg (McMahan et al. 2017) and SFL (Thapa et al. 2022). Since no relevant work in the SFL field has been published at top conferences, we modified three FL baseline methods aimed at solving data heterogeneity (i.e., FedNTD (Lee et al. 2022), FedExP (Jhunjhunwala et al. 2023), and FedMut (Hu et al. 2024a)) to achieve their SFL versions for a fair comparison. For all methods, we adopted an SGD optimizer with a fixed learning rate of 0.01 and a momentum of 0.5 and set the batch size to 50. For our method, we set α to 0.1 and γ to 0.5. All experimental results were obtained from an Ubuntu workstation equipped with an Intel i9 CPU, 64GB of memory, and an NVIDIA RTX 4090 GPU.

Model	Dataset	Hetero. Settings	Test Accuracy (%)					
			FedAvg	SFL	FedMut	FedNTD	FedExP	MultiSFL (Ours)
MobileNetV2	CIFAR-10	$\beta = 0.1$	41.20 ± 3.34	40.74 ± 3.72	38.05 ± 3.60	50.92 ± 2.83	35.63 ± 3.10	$\mathbf{64.45 \pm 0.33}$
		$\beta = 0.3$	58.58 ± 2.20	54.57 ± 2.51	51.48 ± 1.35	64.45 ± 1.19	53.32 ± 2.37	$\mathbf{69.23 \pm 0.56}$
		$\beta = 0.5$	59.41 ± 0.83	58.69 ± 0.88	61.89 ± 0.51	62.98 ± 0.47	55.87 ± 0.80	$\mathbf{70.44 \pm 0.53}$
		IID	64.75 ± 0.11	63.80 ± 0.18	66.54 ± 0.15	66.99 ± 0.28	62.09 ± 0.09	$\mathbf{75.87 \pm 0.16}$
	CIFAR-100	$\beta = 0.1$	31.38 ± 0.97	31.07 ± 1.28	35.18 ± 1.07	36.53 ± 0.30	33.18 ± 1.49	$\mathbf{45.66 \pm 0.27}$
		$\beta = 0.3$	39.63 ± 0.60	39.37 ± 0.55	39.03 ± 0.57	43.85 ± 0.49	40.66 ± 1.15	$\mathbf{45.36 \pm 0.61}$
		$\beta = 0.5$	38.85 ± 0.30	40.26 ± 0.60	41.55 ± 0.42	42.65 ± 0.28	33.18 ± 1.49	$\mathbf{46.34 \pm 0.67}$
		IID	42.17 ± 0.11	40.63 ± 0.22	44.32 ± 0.21	42.57 ± 0.17	41.70 ± 0.23	$\mathbf{52.53 \pm 0.13}$
	FEMNIST	-	81.34 ± 0.40	80.92 ± 0.35	81.64 ± 0.45	81.32 ± 0.14	80.65 ± 0.47	$\mathbf{82.63 \pm 0.28}$
ResNet18	CIFAR-10	$\beta = 0.1$	48.01 ± 2.73	45.08 ± 3.64	53.84 ± 3.49	51.17 ± 2.27	47.31 ± 2.86	$\mathbf{66.72 \pm 0.48}$
		$\beta = 0.3$	59.99 ± 0.37	60.13 ± 1.01	67.46 ± 0.23	62.83 ± 0.26	59.93 ± 0.33	$\mathbf{69.69 \pm 0.20}$
		$\beta = 0.5$	62.70 ± 0.33	63.00 ± 0.34	67.84 ± 0.76	64.96 ± 0.24	62.80 ± 0.24	$\mathbf{70.54 \pm 0.21}$
		IID	64.79 ± 0.14	64.51 ± 0.22	70.59 ± 0.11	69.10 ± 0.13	64.36 ± 0.13	$\mathbf{73.20 \pm 0.11}$
	CIFAR-100	$\beta = 0.1$	35.23 ± 0.37	35.71 ± 0.52	36.77 ± 0.33	41.46 ± 0.59	35.81 ± 0.57	$\mathbf{47.17 \pm 0.19}$
		$\beta = 0.3$	41.24 ± 0.19	41.30 ± 0.22	45.80 ± 0.24	47.31 ± 0.18	41.54 ± 0.22	$\mathbf{52.00 \pm 0.13}$
		$\beta = 0.5$	42.82 ± 0.19	43.08 ± 0.13	45.90 ± 0.45	48.74 ± 0.15	42.59 ± 0.24	$\mathbf{54.17 \pm 0.15}$
		IID	43.01 ± 0.19	43.10 ± 0.22	48.01 ± 0.09	49.91 ± 0.85	43.10 ± 0.15	$\mathbf{56.06 \pm 0.16}$
	FEMNIST	-	83.01 ± 0.33	83.40 ± 0.24	83.33 ± 0.29	84.18 ± 0.18	83.23 ± 0.25	$\mathbf{84.58 \pm 0.17}$
VGG16	CIFAR-10	$\beta = 0.1$	66.26 ± 4.55	65.86 ± 2.87	65.86 ± 2.52	64.95 ± 0.68	67.47 ± 4.52	$\mathbf{78.33 \pm 0.68}$
		$\beta = 0.3$	77.23 ± 0.26	78.83 ± 0.13	79.78 ± 0.15	78.32 ± 0.75	77.69 ± 0.22	$\mathbf{82.03 \pm 0.08}$
		$\beta = 0.5$	78.47 ± 0.09	80.35 ± 0.12	80.91 ± 0.15	80.14 ± 0.41	79.57 ± 0.28	$\mathbf{82.94 \pm 0.09}$
		IID	79.92 ± 0.07	81.90 ± 0.06	81.78 ± 0.13	82.36 ± 0.11	80.30 ± 0.03	$\mathbf{84.12 \pm 0.08}$
	CIFAR-100	$\beta = 0.1$	47.70 ± 1.63	50.41 ± 0.50	50.20 ± 0.43	49.81 ± 0.81	49.13 ± 0.84	$\mathbf{53.14 \pm 0.30}$
		$\beta = 0.3$	53.93 ± 0.44	56.95 ± 0.18	57.08 ± 0.38	57.62 ± 0.23	54.78 ± 0.69	$\mathbf{60.15 \pm 0.32}$
		$\beta = 0.5$	55.00 ± 0.59	57.52 ± 0.62	57.85 ± 0.44	59.24 ± 0.35	56.42 ± 0.40	$\mathbf{61.23 \pm 0.19}$
		IID	57.26 ± 0.07	60.10 ± 0.14	58.10 ± 0.39	62.19 ± 0.08	58.02 ± 0.10	$\mathbf{66.05 \pm 0.18}$
	FEMNIST	-	82.24 ± 0.57	84.35 ± 0.29	83.58 ± 0.30	83.56 ± 0.42	71.06 ± 1.03	$\mathbf{84.86 \pm 0.25}$
DenseNet161	TinyImageNet	$\beta = 0.01$	21.64 ± 0.34	21.59 ± 0.47	20.06 ± 0.35	21.86 ± 0.20	20.83 ± 0.31	$\mathbf{34.86 \pm 0.21}$
		$\beta = 0.05$	33.65 ± 0.42	33.64 ± 0.33	31.56 ± 0.43	33.61 ± 0.27	29.93 ± 0.38	$\mathbf{39.32 \pm 0.21}$
		$\beta = 0.1$	35.71 ± 0.27	36.17 ± 0.26	33.92 ± 0.42	35.93 ± 0.17	32.25 ± 0.17	$\mathbf{40.52 \pm 0.15}$
		IID	38.53 ± 0.11	39.61 ± 0.18	36.80 ± 0.17	37.52 ± 0.08	34.27 ± 0.14	$\mathbf{43.00 \pm 0.11}$

Table 1: Comparison of test accuracy.



(a) $\beta = 0.1$ (b) $\beta = 0.3$

(c) $\beta = 0.5$ (d) IID

Figure 3: Comparison of communication overhead on CIFAR-10 using VGG-16.

Experimental Settings

We compared MultiSFL with all baselines on four well-known datasets, i.e., CIFAR-10, CIFAR-100 (Krizhevsky 2009), FEMNIST (Caldas et al. 2018), and TinyImageNet (Deng et al. 2009). To mimic the non-IID distributions of device data for CIFAR-10, CIFAR-100, and TinyImageNet, we employed the Dirichlet distribution $Dir(\beta)$ (Hsu, Qi, and Brown 2019), where the smaller values of β indicate greater data heterogeneity. Note that since the dataset FEMNIST itself is naturally non-IID distributed, we did not apply the Dirichlet distribution to the FEMNIST dataset. To show the pervasiveness of our approach, we considered four widely used DNN models, i.e., MobileNetV2 (Sandler et al. 2018), ResNet-18 (He et al. 2016), VGG-16 (Simonyan and Zisserman 2014) and DenseNet-161 (Huang et al. 2017), which have different structures and depths. We simulated 100 clients and assumed that only 10% of devices (i.e., $n = 10$) were selected as main clients in the local training of one FL communication round.

Performance Comparison

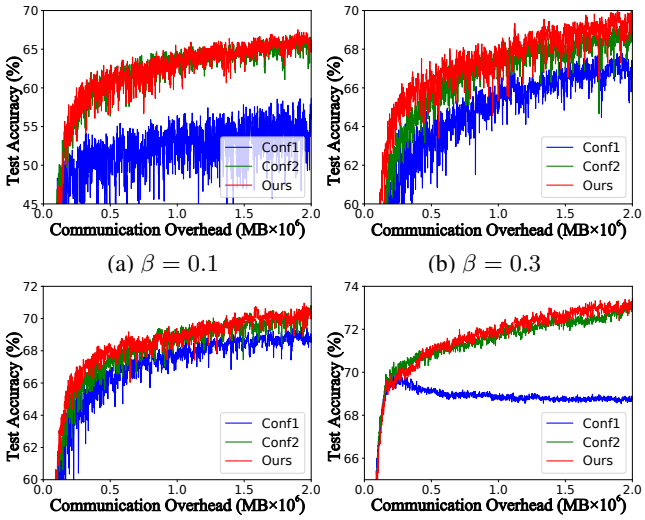
Comparison of Communication Overhead. Figure 3 compares the learning trends of all baselines, where the horizontal axes denote communication overhead along the training processes. We can observe that although MultiSFL incurs additional communication overhead caused by sampling features and assistant client models, our approach still achieves the highest accuracy for the same communication overhead. Meanwhile, we can find that the learning curves of our method become much more stable than other methods when β decreases, meaning higher data heterogeneity.

Comparison of Accuracy. Table 1 compares the test accuracy between MultiSFL and all the baselines in different settings involving different models, datasets, and data distributions. From this table, we can see that MultiSFL al-

ways achieves the best performance in all cases. Especially for the combination of the CIFAR-10 dataset and the MobileNetV2 model, MultiSFL outperforms SFL by 23.71% when $\beta = 0.1$. We can observe that as β decreases, the test accuracy gap of all baselines between IID and non-IID cases increases significantly. In other words, although other baselines have improved compared to FedAvg and SFL, they suffer the same degree of accuracy degradation as SFL and FedAvg in non-IID scenarios. Since our approach effectively mitigates the problem of catastrophic forgetting, such gaps are much smaller than those obtained by other baseline methods. In other words, MultiSFL leads to the smallest accuracy degradation between IID and non-IID scenarios, showing its superiority in robustness against various non-IID data distributions.

Ablation Study

To demonstrate the effectiveness of our proposed mechanisms in MultiSFL, we investigated two variants of MultiSFL: i) *Conf1* without considering knowledge replay; and ii) *Conf2* with both knowledge replay and fixed feature sampling proportion. Note that *Conf2* has the same overall feature sampling proportion as MultiSFL, and *Conf1*, *Conf2*, and MultiSFL have the same total communication overhead. We conducted experiments on CIFAR-10 using the ResNet-18 model considering non-IID and IID settings. Based on the results of Figure 4, we can find that MultiSFL can always achieve the highest test accuracy, demonstrating the effectiveness of our proposed mechanisms.



(a) $\beta = 0.1$ (b) $\beta = 0.3$

(c) $\beta = 0.5$ (d) IID

Figure 4: Learning curves for different configurations.

From Figure 4, we can find that compared with *Conf1*, both *Conf2* and MultiSFL based on knowledge replay can achieve more stable learning curves and better test accuracy. Things become even more significant when dealing with scenarios with higher heterogeneity, revealing the effectiveness of our proposed knowledge replay method. Compared with *Conf2*, our proposed sampling proportion adaptive adjustment strategy speeds up the training process of MultiSFL, achieving slightly better inference performance and demonstrating the effectiveness of our proposed adaptive adjustment of sampling proportion.

Discussions

Time Complexity. Assume that a complete model has w parameters and the training dataset contains various data with c types. Moreover, assume that there are m clients in total, and in each communication round, n of them are selected as the main clients to participate in the local training. Please note that, in our approach, the additional overhead is mainly caused by two operations, i.e., Adaptive Adjustment of Sampling Proportion (AASP) and Knowledge Replay Strategy (KRS), and the time complexity of AASP is O(w). As shown in Algorithm 1, we can figure out that the time complexity of KRS is O($n(c+cm)$). Note that since the time cost of the model training on the server is much higher than that of AASP and KRS, the additional time cost caused by our approach is negligible.

Limitations. In our method, the segmentation positions are the same for all models. In other words, our approach cannot work on heterogeneous models (Jia et al. 2024; Chen et al. 2024). In the future, we will explore a strategy to adaptively adjust the splitting position of the model to accelerate model training and improve global model accuracy. In addition, security and robustness properties, e.g., backdoor attack (Yang et al. 2023) and fairness (Li et al. 2023b,c), are still interesting topics for future work.

Conclusion

This paper proposed a novel SFL framework named MultiSFL, aiming to address the notorious catastrophic forgetting problem when dealing with non-IID scenarios. By continuously conducting local training on clients without being interrupted by the global model for aggregation, the training process of models in MultiSFL approximately traverses all the client data, thus mitigating the impact of non-IID distributions of client data. Meanwhile, by applying our proposed sampling strategy for assistant clients and knowledge replay mechanism, the overall training process of MultiSFL can be accelerated, and the underlying catastrophic forgetting symptom can be greatly suppressed. In addition to the theoretical convergence analysis of MultiSFL, we conducted comprehensive experiments to demonstrate the superiority of MultiSFL in improving the overall test accuracy.

Acknowledgements

This work was supported by Natural Science Foundation of China (62272170), Shanghai Trusted Industry Internet Software Collaborative Innovation Center, the National Research Foundation, Singapore, and the Cyber Security Agency under its National Cybersecurity R&D Programme (NCRP25-P04-TAICeN). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Cyber Security Agency of Singapore.

References

Caldas, S.; Duddu, S. M. K.; Wu, P.; Li, T.; Konečnỳ, J.; McMahan, H. B.; Smith, V.; and Talwalkar, A. 2018. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*.

Chen, C.; Chen, Z.; Zhou, Y.; and Kailkhura, B. 2020. Fedcluster: Boosting the convergence of federated learning via cluster-cycling. In *Proceedings of IEEE International Conference on Big Data*, 5017–5026.

Chen, Z.; Jia, C.; Hu, M.; Xie, X.; Li, A.; and Chen, M. 2024. FlexFL: Heterogeneous Federated Learning via APoZ-Guided Flexible Pruning in Uncertain Scenarios. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 43(11): 4069–4080.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Li, F.-F. 2009. Imagenet: A large-scale hierarchical image database. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 248–255.

Gao, F.; Hu, M.; Xie, Z.; Shi, P.; Xie, X.; Yi, G.; and Wang, H. 2024. NebulaFL: Effective Asynchronous Federated Learning for JointCloud Computing. *arXiv preprint arXiv:2412.04868*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of 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*.

Hu, M.; Cao, E.; Huang, H.; Zhang, M.; Chen, X.; and Chen, M. 2023a. AIoTML: A Unified Modeling Language for AIoT-Based Cyber-Physical Systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(11): 3545–3558.

Hu, M.; Cao, Y.; Li, A.; Li, Z.; Liu, C.; Li, T.; Chen, M.; and Liu, Y. 2024a. FedMut: Generalized Federated Learning via Stochastic Mutation. In *Proceedings of AAAI Conference on Artificial Intelligence*, 12528–12537.

Hu, M.; Xia, Z.; Yan, D.; Yue, Z.; Xia, J.; Huang, Y.; Liu, Y.; and Chen, M. 2023b. GitFL: Uncertainty-Aware Real-Time Asynchronous Federated Learning using Version Control. In *Proceedings of IEEE Real-Time Systems Symposium*, 145–157.

Hu, M.; Yue, Z.; Xie, X.; Chen, C.; Huang, Y.; Wei, X.; Lian, X.; Liu, Y.; and Chen, M. 2024b. Is aggregation the only choice? federated learning via layer-wise model recombination. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 1096–1107.

Hu, M.; Zhou, P.; Yue, Z.; Ling, Z.; Huang, Y.; Li, A.; Liu, Y.; Lian, X.; and Chen, M. 2024c. FedCross: Towards Accurate Federated Learning via Multi-Model Cross-Aggregation. In *Proceedings of IEEE International Conference on Data Engineering*, 2137–2150.

Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 4700–4708.

Huang, W.; Ye, M.; Shi, Z.; and Du, B. 2024. Generalizable Heterogeneous Federated Cross-Correlation and Instance Similarity Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46: 2.

Huang, W.; Ye, M.; Shi, Z.; Li, H.; and Du, B. 2023. Rethinking federated learning with domain shift: A prototype view. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 16312–16322.

Jhunjhunwala, D.; et al. 2023. FedExP: Speeding Up Federated Averaging via Extrapolation. In *Proceedings of International Conference on Learning Representations*.

Jia, C.; Hu, M.; Chen, Z.; Yang, Y.; Xie, X.; Liu, Y.; and Chen, M. 2024. AdaptiveFL: Adaptive heterogeneous federated learning for resource-constrained AIoT systems. In *Proceedings of ACM/IEEE Design Automation Conference*, 1–6.

Karimireddy, S. P.; Kale, S.; Mohri, M.; Reddi, S.; Stich, S.; and Suresh, A. T. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *Proceedings of International Conference on Machine Learning*, 5132–5143.

Krizhevsky, A. 2009. Learning multiple layers of features from tiny images. *https://www.cs.toronto.edu/$\sim$ $kriz/cifar.html$*.

Lee, G.; Jeong, M.; Shin, Y.; Bae, S.; and Yun, S.-Y. 2022. Preservation of the global knowledge by not-true distillation in federated learning. In Proceedings of Advances in Neural Information Processing Systems*, 38461–38474.*

Li, A.; Liu, R.; Hu, M.; Tuan, L. A.; and Yu, H. 2023a. Towards interpretable federated learning. arXiv preprint arXiv:2302.13473.

Li, T.; Guo, Q.; Liu, A.; Du, M.; Li, Z.; and Liu, Y. 2023b. FAIRER: fairness as decision rationale alignment. In Proceedings of International Conference on Machine Learning*, 19471–19489.*

Li, T.; Li, Z.; Li, A.; Du, M.; Liu, A.; Guo, Q.; Meng, G.; and Liu, Y. 2023c. Fairness via Group Contribution Matching. In Proceedings of International Joint Conference on Artificial Intelligence*, 436–445.*

Li, T.; Sahu, A. K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; and Smith, V. 2020a. Federated optimization in heterogeneous networks. In Proceedings of Machine Learning and Systems*, 429–450.*

Li, X.; Huang, K.; Yang, W.; Wang, S.; and Zhang, Z. 2020b. On the Convergence of FedAvg on Non-IID Data. In Proceedings of International Conference on Learning Representations (ICLR).

Lin, T.; Kong, L.; Stich, S. U.; and Jaggi, M. 2020. Ensemble distillation for robust model fusion in federated learning. In Proceedings of Advances in Neural Information Processing Systems*, volume 33, 2351–2363.*

Luo, K.; Li, X.; Lan, Y.; and Gao, M. 2023. GradMA: A Gradient-Memory-based Accelerated Federated Learning with Alleviated Catastrophic Forgetting. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3708–3717.*

Ma, Q.; Xu, Y.; Xu, H.; Jiang, Z.; Huang, L.; and Huang, H. 2021. FedSA: A Semi-Asynchronous Federated Learning Mechanism in Heterogeneous Edge Computing. IEEE Journal of Selected Areas in Communications*, 39(12): 3654–3672.*

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

Meng, L.; Qi, Z.; Wu, L.; Du, X.; Li, Z.; Cui, L.; and Meng, X. 2024. Improving Global Generalization and Local Personalization for Federated Learning. IEEE Transactions on Neural Networks and Learning Systems*.*

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In Proceedings of Advances in Neural Information Processing Systems*, 8024–8035.*

Qi, Z.; Meng, L.; Chen, Z.; Hu, H.; Lin, H.; and Meng, X. 2023. Cross-Silo Prototypical Calibration for Federated Learning with Non-IID Data. In Proceedings of ACM International Conference on Multimedia*, 3099–3107.*

Qi, Z.; Meng, L.; He, W.; Zhang, R.; Wang, Y.; Qi, X.; and Meng, X. 2024. Cross-Training with Multi-View Knowledge Fusion for Heterogenous Federated Learning. arXiv preprint arXiv:2405.20046*.*

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 4510–4520.*

Sattler, F.; Wiedemann, S.; Müller, K.-R.; and Samek, W. 2020. Robust and Communication-Efficient Federated Learning From Non-i.i.d. Data. IEEE Transactions on Neural Networks and Learning Systems*, 31(9): 3400–3413.*

Shao, C.; and Feng, Y. 2022. Overcoming Catastrophic Forgetting beyond Continual Learning: Balanced Training for Neural Machine Translation. In Proceedings of Annual Meeting of the Association for Computational Linguistics*, 2023–2036.*

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556*.*

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

Wang, H.; Jia, Y.; Zhang, M.; Hu, Q.; Ren, H.; Sun, P.; Wen, Y.; and Zhang, T. 2024a. FedDSE: Distribution-aware Submodel Extraction for Federated Learning over Resource-constrained Devices. In Proceedings of ACM on Web Conference*, 2902–2913.*

Wang, H.; Li, Y.; Xu, W.; Li, R.; Zhan, Y.; and Zeng, Z. 2023. Dafkd: Domain-aware federated knowledge distillation. In Proceedings of IEEE/CVF conference on computer vision and pattern recognition*, 20412–20421.*

Wang, H.; Zheng, P.; Han, X.; Xu, W.; Li, R.; and Zhang, T. 2024b. FedNLR: Federated Learning with Neuron-wise Learning Rates. In Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 3069–3080.*

Wu, F.; Guo, S.; Qu, Z.; He, S.; Liu, Z.; and Gao, J. 2023a. Anchor sampling for federated learning with partial client participation. In Proceedings of International Conference on Machine Learning*, 37379–37416.*

Wu, F.; Guo, S.; Wang, H.; Zhang, H.; Qu, Z.; Zhang, J.; and Liu, Z. 2023b. From deterioration to acceleration: A calibration approach to rehabilitating step asynchronism in federated optimization. IEEE Transactions on Parallel and Distributed Systems*, 34(5): 1548–1559.*

Xu, C.; Hong, Z.; Huang, M.; and Jiang, T. 2022. Acceleration of federated learning with alleviated forgetting in local training. arXiv preprint arXiv:2203.02645*.*

Yan, G.; Wang, H.; and Li, J. 2022. Seizing critical learning periods in federated learning. In Proceedings of AAAI Conference on Artificial Intelligence*, volume 36, 8788–8796.*

Yan, G.; Wang, H.; Yuan, X.; and Li, J. 2023. CriticalFL: A Critical Learning Periods Augmented Client Selection Framework for Efficient Federated Learning. In Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2898–2907.*

Yang, Y.; Hu, M.; Cao, Y.; Xia, J.; Huang, Y.; Liu, Y.; and Chen, M. 2023. Protect federated learning against backdoor attacks via data-free trigger generation. arXiv preprint arXiv:2308.11333*.*

Zhang, X.; Hu, M.; Xia, J.; Wei, T.; Chen, M.; and Hu, S. 2020. Efficient federated learning for cloud-based AIoT applications. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(11): 2211–2223.*

Zhu, Z.; Hong, J.; and Zhou, J. 2021. Data-free knowledge distillation for heterogeneous federated learning. In Proc. of International Conference on Machine Learning*, 12878–12889.*