FedSPU: Personalized Federated Learning for Resource-Constrained Devices with Stochastic Parameter Update

Ziru Niu[1], **Hai Dong**[1*], **A. K. Qin**[2]
[1]RMIT University, Melbourne, VIC 3000, Australia
[2]Swinburne University of Technology, Hawthorn, VIC 3122, Australia
ziru.niu@student.rmit.edu.au, hai.dong@rmit.edu.au, kqin@swin.edu.au

Abstract

Personalized Federated Learning (PFL) is widely employed in the Internet of Things (IoT) to handle high-volume, non-iid client data while ensuring data privacy. However, heterogeneous edge devices owned by clients may impose varying degrees of resource constraints, causing computation and communication bottlenecks for PFL. Federated Dropout has emerged as a popular strategy to address this challenge, wherein only a subset of the global model, i.e. a *sub-model*, is trained on a client's device, thereby reducing computation and communication overheads. Nevertheless, the dropout-based model-pruning strategy may introduce bias, particularly towards non-iid local data. When biased sub-models absorb highly divergent parameters from other clients, performance degradation becomes inevitable. In response, we propose federated learning with stochastic parameter update (FedSPU). Unlike dropout that tailors local models to small-size sub-models, FedSPU maintains the full model architecture on each device but randomly freezes a certain percentage of neurons in the local model during training while updating the remaining neurons. This approach ensures that a portion of the local model remains personalized, thereby enhancing the model's robustness against biased parameters from other clients. Experimental results demonstrate that FedSPU outperforms federated dropout by 4.45% on average in terms of accuracy. Furthermore, an introduced early stopping scheme leads to a significant reduction of the training time in FedSPU by $25\% \sim 71\%$ while maintaining high accuracy.

Introduction

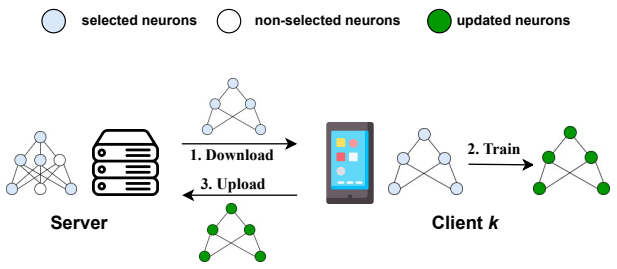
Federated Learning (FL) is a distributed machine learning paradigm that allows edge devices to collaboratively train a model without revealing private data (McMahan et al. 2017). However, the efficacy of FL in real-world IoT systems is usually impeded by both data and system heterogeneities of IoT clients. First, clients comprise edge devices from various geographical locations collecting data *that are naturally non-independent identical (non-iid)*. In such a scenario, a single global model struggles to generalize across all local datasets (Kulkarni, Kulkarni, and Pant 2020; Li et al. 2021). Second, IoT clients consist of physical devices with varying processor, memory, and bandwidth capabilities (Smith et al. 2017; Imteaj et al. 2022). Among these devices, some resource-constrained devices might be incapable of training the entire global model with a too complex structure.

To overcome the non-iid data problem, the Personalized Federated Learning (PFL) framework is introduced by empowering each client to maintain a unique local model tailored to its local data distribution. To address system heterogeneity, the technique of *federated dropout* (i.e. model pruning) is employed. Resource-constrained devices are allowed to train a *sub-model*, which is a subset of the global model. This approach reduces computation and communication overheads for training and transmitting the sub-model, aiding resource-constrained devices in overcoming computation and communication bottlenecks (Caldas et al. 2018; Horváth et al. 2021).

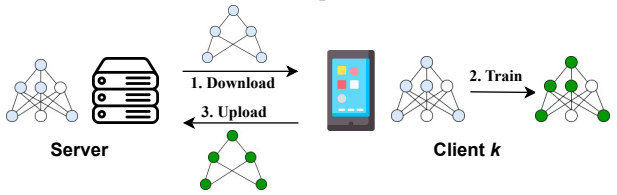
Various PFL frameworks have been developed to tackle the non-iid data challenge (Khodak, Balcan, and Talwalkar 2019; Fallah, Mokhtari, and Ozdaglar 2020; Arivazhagan et al. 2019; Sattler, Müller, and Samek 2021; Smith et al. 2017; Zhang et al. 2021; Marfoq et al. 2021; Wang et al. 2023; Dinh, Tran, and Nguyen 2020; Mansour et al. 2020; Deng, Kamani, and Mahdavi 2021; Zhang et al. 2023). Nevertheless, the inherent communication or computation bottleneck of resource-constrained edge devices is often overlooked in these frameworks. To tackle the computation and communication bottlenecks, *federated dropout* is applied (Caldas et al. 2018; Wen, Jeon, and Huang 2022; Horváth et al. 2021; Li et al. 2021; Jiang et al. 2022, 2023), where resource-constrained devices are allowed to train a subset of the global model, i.e. a *sub-model*. Compared with a full model, the computation and communication overheads for training and transmitting sub-models are reduced, facilitating resource-constrained devices to complete the training task within constraints. For example, in (Caldas et al. 2018; Wen, Jeon, and Huang 2022), the server randomly prunes neurons in the global model. In (Horváth et al. 2021), the server prunes the rightmost neurons in the global model. These works fail to meet the *personalization* requirement, as the server arbitrarily decides the architectures of local sub-models without considering the importance of neurons based on the *local data distributions* of clients.

On the other hand, (Jiang et al. 2023; Li et al. 2021; Jiang et al. 2022) let clients adaptively prune neurons to obtain the optimal architecture of the local sub-model. Each client first

*Corresponding author.



(a) Dropout.



(b) FedSPU.

Figure 1: In dropout (a), clients train sub-models with fewer parameters. In FedSPU (b), clients train full models with partial parameters frozen.

pre-trains an initialized model to evaluate the importance of each neuron. Then each client locally prunes the unimportant neurons and shares the remaining sub-model with the server hereafter. Nevertheless, evaluating neuron importance requires full-model training on the client side, which might be *expensive* or *prohibitive* for resource-constrained devices with a critical computation bottleneck. Furthermore, the adaptive model-pruning behaviors proposed in (Jiang et al. 2023; Li et al. 2021; Jiang et al. 2022) rely immensely on the local data. The non-iid local data distributions often lead to *highly unbalanced class distributions* across different clients (Puyol-Antón et al. 2021; Padurariu and Breaban 2019; Wang et al. 2021; Abeysekara, Dong, and Qin 2021). Consequently, the local dropout behavior can be *heavily biased* (Jiang et al. 2022), and the sub-model architecture among clients may *vary drastically*. In global communication, when a client absorbs parameters from other clients with inconsistent model architectures, the performance of the local model will be inevitably compromised.

To address the limitation of existing works, this paper introduces *Federated Learning with Stochastic Parameter Update (FedSPU)*, a consolidated PFL framework aimed at mitigating the issue of *local model personalization loss* while considering computation and communication bottlenecks in resource-constrained devices. It is observed that during global communication, a client's entire local sub-model is replaced by *biased parameters* of other clients, leading to local model personalization loss. Therefore, if we let a client share only a *partial model* with others, the adverse effect of other clients' biased parameters can be alleviated. Inspired by this, FedSPU *freezes* neurons instead of pruning them, as shown in Figure 1. Frozen neurons do not receive gradients during backpropagation and remain unaltered in subsequent updates. This approach eliminates computation overheads in backward propagation, enhancing *computational efficiency*. Moreover, FedSPU does not incur extra communication overheads compared with dropout, as only the parameters of the non-frozen neurons and the positions of these neurons are communicated between clients and the server, as depicted in Figure 1. Besides, compared with model parameters, the communication cost for sending the position indices of the non-frozen neurons is much smaller and usually ignorable (Li et al. 2021).

Unlike pruned neurons, frozen neurons persist within a local model, and still contribute to the model's final output. This design choice incurs slightly *higher computation costs* of forward propagation, but largely improves local personalization, as only a portion of a local model is replaced during communication as shown in Figure 1b. Moreover, the increased computation overhead of forward propagation is a minor problem, as in training, forward propagation constitutes a *significantly smaller* portion of the total computation overhead than backpropagation (Li et al. 2020; He and Sun 2015). Additionally, to alleviate the overall computation and communication costs, we consolidate FedSPU with an *early stopping* technique (Prechelt 2002; Niu et al. 2024). At each round, each client locally computes the training and testing errors, and compares them with the errors from the previous round. When the errors show no decrease, this client will cease training and no longer participate in FL. When all clients have halted training, FedSPU will terminate in advance to conserve computation and communication resources. We evaluate FedSPU on three typical deep learning datasets: EMNIST (Cohen et al. 2017), CIFAR10 (Krizhevsky et al. 2009) and Google Speech (Warden 2018), with five state-of-the-art dropout methods included for comparison: FjORD (Horváth et al. 2021), Hermes (Li et al. 2021), FedMP (Jiang et al. 2023), PruneFL (Jiang et al. 2022) and FedSelect (Tamirisa et al. 2024). Experiment results show that:

- FedSPU's unique approach of freezing neurons, rather than pruning, retains them within the local model, enhancing personalization. This design choice improves final model accuracy by an average of 4.45% over existing dropout methods.
- FedSPU reduces memory usage through neuron freezing rather than full-model training, avoiding memory-intensive dropout processes. This results in significant memory savings up to 54% at higher dropout rates.
- The integration of an early stopping mechanism allows FedSPU to reduce computation and communication costs significantly by 25%∼71%. Compared with existing methods, FedSPU exhibits better compatibility with early stopping with an average accuracy improvement of at least 5.11%.

Related Work

Personalized Federated Learning

To the best of our knowledge, existing PFL methods can be divided into three categories. **1) Fine-tuning:** Clients at first train a global model collaboratively, followed by indi-

vidual fine-tuning to adapt to local datasets. In (Arivazhagan et al. 2019), each client fine-tunes some layers of the global model to adapt to the local dataset. (Fallah, Mokhtari, and Ozdaglar 2020) proposes to find an initial global model that generalizes well to all clients through model-agnostic-meta-learning formulation, then clients fine-tune the model locally through just a few gradient steps. (Khodak, Balcan, and Talwalkar 2019) proposes to improve the global model's generalization with dynamic learning rate adaption based on the squared difference between local gradients, which significantly reduces the workload of local fine-tuning. (Sattler, Müller, and Samek 2021) fine-tunes the global model based on clusters, and clients from the same cluster share the same local model. **2) Personal Training:** Clients train personal models at the beginning. (Dinh, Tran, and Nguyen 2020) adds a regularization term to each local objective function that keeps local models from diverging too far from the global model to improve the global model's generalization. (Marfoq et al. 2021) assumes each local data distribution is a mixture of several unknown distributions, and optimizes each local model with the expectation-maximization algorithm. (Smith et al. 2017) speeds up the convergence of each local model by modeling local training as a primal-dual optimization problem. (Wang et al. 2023) lets each client maintain a personal head while training to improve the global model's generalization by aggregating local heads on the server side. (Zhang et al. 2021) proposes to train local models through first-order optimization, where the local objective function becomes the error subtraction between clients. **3) Hybrid:** Clients merge local and global models to foster mutual learning and maintain personalization. In (Deng, Kamani, and Mahdavi 2021; Mansour et al. 2020; Zhang et al. 2023), each client simultaneously trains the global model and its local model. The ultimate model for each client is a combination of the global and the local model.

Even though these works potently address the non-iid data problem in FL, they neglect the computation or communication bottleneck of resource-constrained IoT devices.

Dropout in Federated Learning

In centralized machine learning, dropout is used as a regularization method to prevent a neural network from overfitting (Srivastava et al. 2014). Nowadays, as federated learning becomes popular in the IoT industry, dropout has also been applied to address the computation and communication bottlenecks of resource-constrained IoT devices. In dropout, clients are allowed to train and transmit a subset of the global model to reduce the computation and communication overheads. To extract a subset from the global model, i.e. a sub-model, Random Dropout (Caldas et al. 2018; Wen, Jeon, and Huang 2022) randomly prune neurons in the global model. FjORD (Horváth et al. 2021) continually prunes the rightmost neurons in a neural network. FedMP (Jiang et al. 2023), Hermes (Li et al. 2021) and PruneFL (Jiang et al. 2022) let clients adaptively prune the unimportant neurons, and the importance of a neuron is the *l1*-norm, *l2*-norm of parameters, and *l2*-norm of gradient respectively. FedSelect (Tamirisa et al. 2024) lets all clients extract a tiny sub-model first by pruning the unimportant neurons with the least gradient norms, and gradually expand the width of the sub-model during training.

Random Dropout and FjORD represent *global dropout* methods and do not meet the personalization requirement, as the server arbitrarily prunes neurons without considering clients' non-iid data. On the other hand, *local dropout* methods such as FedMP, Hermes, PruneFL and FedSelect may be hindered by the bias of unbalanced local datasets.

Compared with existing works, FedSPU comprehensively meets the personalization requirement and addresses the computation and communication bottlenecks of resource-constrained devices, meanwhile overcoming data imbalance.

FedSPU: Federated Learning with Stochastic Parameter Update

Problem Setting

Given a set $\mathbb{C} = \{1, 2, ..., N\}$ of clients with local datasets $\{D_1, D_2, ..., D_N\}$ and local models $\{w_1, w_2, ..., w_N\}$. The goal of a PFL framework is to determine the optimal set of local models $\{w_1^*, w_2^*, ..., w_N^*\}$ such that:

$$w_1^*, ..., w_N^* \triangleq \underset{w_1,...,w_N}{\arg\min} \frac{1}{N} \sum_{k=1}^{N} F_k(w_k) \tag{1}$$

where w_k^* is the optimal model for client k ($1 \leq k \leq N$), and F_k is the objective function of client k. F_k is equivalent to the empirical risk over k's local dataset D_k. That is:

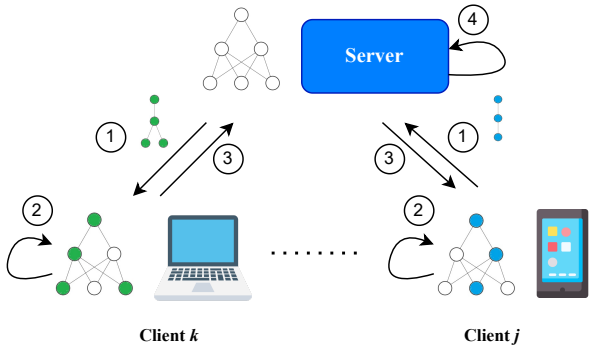
$$F_k(w_k) = \frac{1}{n_k} \sum_{i=1}^{n_k} \mathcal{L}((\boldsymbol{x}_i, y_i), w_k) \tag{2}$$

where n_k is the size of dataset D_k and $\mathcal{L}$ is the loss function of model w_k over the $i-$th sample $(\boldsymbol{x}_i, y_i)$.

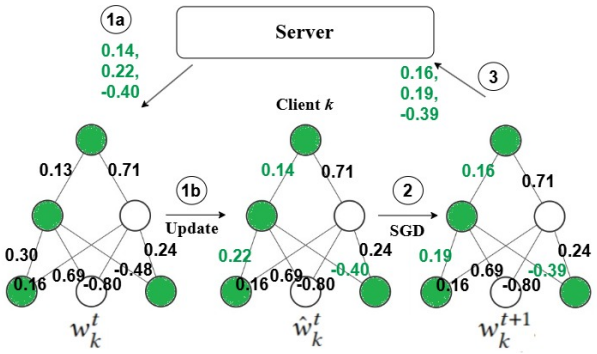
Algorithm 1: FedSPU

Require: maximum global iteration T, clients $\mathbb{C} = \{1, ..., N\}$, initial global model w^0.

Server broadcasts w_0 to all clients.
For round $t = 1, 2, ..., T$:
Server executes:
  randomly sample a subset of clients $\mathbb{C}_t \subset \mathbb{C}$.
  $\forall k \in \mathbb{C}_t$:
    randomly sample $A_k(w^t)$ based on p_k.
    send $A_k(w^t)$ to k.
Each client $k \in \mathbb{C}_t$ **in parallel does**:
  merge $A_k(w^t)$ into w_k^t to get $\hat{w}_k^t$. ▷ see Fig. 2b
  local SGD: $w_k^{t+1} = \hat{w}_k^t - \eta \nabla \bar{F}_k(\hat{w}_k^t)$. ▷ see Eq. (3)
  send $A_k(w_k^{t+1})$ to the server. ▷ see Fig. 2b
Server executes:
  $\forall k \in \mathbb{C}_t$:
    receive $A_k(w_k^{t+1})$ from k.
  Aggregate all $A_k(w_k^{t+1})$ to get w^{t+1}. ▷ see Fig. 3
return $w_1, w_2, ..., w_N$.



(a) Overview of FedSPU.



(b) The local training procedure of client k in FedSPU.

Figure 2: Demonstration of the FedSPU framework.

Solution Overview

A comprehensive procedure of FedSPU is presented in Algorithm 1, with an overview presented in Figure 2. As shown in Figure 2a, at round t, the server randomly selects a set of participating clients $\mathbb{C}_t$ and executes steps ①∼④.

①. For every participating client k, the server selects a set of active neurons from the global model, and sends the neurons' parameters $A_k(w^t)$ to k. Specifically, in each layer, random p_k of the neurons are selected, where $p_k \in (0, 1]$ is the ratio of active neurons. The value of p_k depends on the system characteristic of the client k, with more powerful k's device (e.g. base station, data silo) having larger p_k. The active neurons are selected randomly to ensure uniform parameter updates. Locally, client k updates the local model w_k^t with the received $A_k(w^t)$ to obtain an intermediate model $\hat{w}_k^t$ as shown in Figure 2b.

②. Client k updates model $\hat{w}_k^t$ using stochastic gradient descent (SGD) to get a new model w_k^{t+1} following Equation (3):

$$w_k^{t+1} = \hat{w}_k^t - \eta \nabla \bar{F}_k(\hat{w}_k^t) \tag{3}$$

where η is the learning rate and $\nabla \bar{F}_k(\hat{w}_k^t)$ is the gradient of F_k with respect to only the active parameters. That is, for all elements $\{\hat{w}_{k,1}^t, ..., \hat{w}_{k,m}^t\}$ in $\hat{w}_k^t$, we have:

$$\nabla \bar{F}_k(\hat{w}_{k,i}^t) = \begin{cases} \nabla F_k(\hat{w}_{k,i}^t), & \hat{w}_{k,i}^t \in A_k(w^t) \\ 0, & \text{otherwise.} \end{cases}, 1 \le i \le m \tag{4}$$

In this step, only the active parameters are updated as shown in Figure 2b.

③. Client k uploads the updated active parameters $A_k(w_k^{t+1})$ to the server.

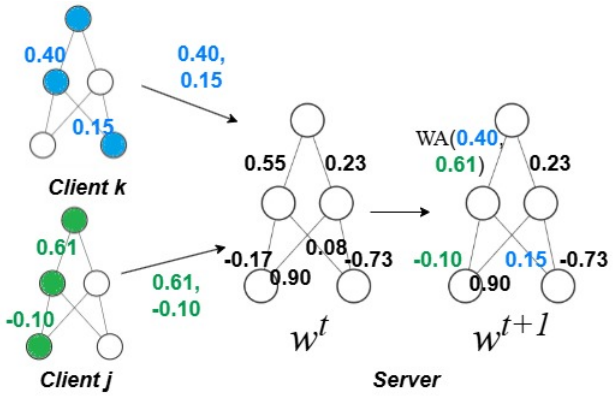


Figure 3: The aggregation scheme in FedSPU.

④. The server aggregates all updated parameters and updates the global model. In this step, FedSPU applies a standard aggregation scheme commonly used in existing dropout methods, where only the active parameters get aggregated and updated (Caldas et al. 2018; Horváth et al. 2021; Li et al. 2021). Figure 3 shows a simple example of how the aggregation scheme works, where "WA" stands for weighted average.

Full Local Model Preserves Personalization

FedSPU freezes neurons instead of pruning them to preserve the integrity of the local model architecture, thereby preserving the personalization of local models. For clarity, Figure 4 shows a comparison between a local sub-model and a local full model. As shown in the left-hand side of Figure 4, in global communication, when receiving other clients' biased parameters from the server, the entire local sub-model is replaced, resulting in a loss of personalization. Conversely, as illustrated in the right-hand side of Figure 4, for a local full model, only partial parameters are replaced, while the remainder remains personalized. This limits the adverse effect of biased parameters from other clients, enabling the local model to maintain performance on the local dataset.

Enhancing FedSPU with Early Stopping Strategy

Since FedSPU slightly increases the computation overhead, it requires more computation resources (e.g., energy (Imteaj et al. 2022), time (He and Sun 2015)) for training. This may pose a challenge for resource-constrained devices. To address this concern, it is expected to reduce the training time of FedSPU without sacrificing accuracy (Niu et al. 2024). Motivated by this, we enhance FedSPU with the **Early Stopping (ES)** technique (Prechelt 2002) to prevent clients from unnecessary training to avoid the substantial consumption of computation and communication resources. At round t, after training, each client k computes $\mathcal{L}_t$ following Equation (5):

$$\mathcal{L}_t = \lambda \mathcal{L}_{train} + (1 - \lambda)\mathcal{L}_{test}. \tag{5}$$

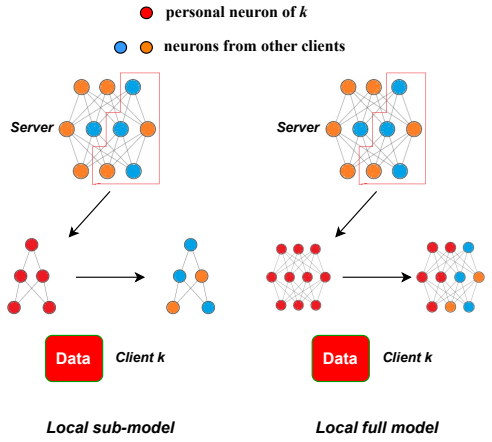


Figure 4: Comparison between replacing all parameters of a local sub-model and replacing a portion of the parameters for a local full model during global communication.

$\mathcal{L}_{train}$ is the training error of current round t, $\mathcal{L}_{test}$ is the testing error of w_k^t on k's validation set. $\lambda \in (0,1)$ is the train-test split factor of the local dataset D_k. When the loss $\mathcal{L}_t$ is non-decreasing, i.e. $\mathcal{L}_t > \mathcal{L}_{t-1}$, client k will stop training and no longer participate in FL due to resource concerns. If all clients have stopped training before the maximum global iteration T, FedSPU will terminate prematurely. The enhanced FedSPU framework with early stopping is detailed in Algorithm 2.

Convergence Analysis

This section analyzes the convergence of each local model in FedSPU. We first make some common assumptions following existing works (Li et al. 2021; Jiang et al. 2023):

Assumption 1. *Every local objective function F_k is $L-$smooth ($L > 0$). That is, $\forall w_1, w_2$, we have:*

$$F_k(w_2) - F_k(w_1) \leq \langle \nabla F_k(w_1), w_2 - w_1 \rangle + \frac{L}{2}\|w_2 - w_1\|^2.$$

Assumption 2. *The divergence between local gradients with and without incorporating the parameter received from the server is bounded:*

$$\exists\, Q > 0, \forall\, k, t,\ \frac{\mathbb{E}(\|\nabla F_k(w_k^t)\|^2)}{\mathbb{E}(\|\nabla F_k(\hat{w}_k^t)\|^2)} \leq Q.$$

Assumption 3. *The divergence between the local parameters with and without incorporating the parameter received from the server is bounded:*

$$\exists\, \sigma > 0, \forall\, k, t,\ \mathbb{E}(\|\hat{w}_k^t - w_k^t\|^2) \leq \sigma^2.$$

Additionally, with respect to the gradient $\nabla \bar{F}_k$, we derive the following lemmas:

Lemma 1. $\forall\, w_k,\ \mathbb{E}(\|\bar{F}_k(w_k)\|^2) = p_k^2\, \|F_k(w_k)\|^2$.

Lemma 2. $\forall w_k, \langle \nabla F_k(w_k), \nabla \bar{F}_k(w_k) \rangle = \|\nabla \bar{F}_k(w_k)\|^2$.

Based on the assumptions and lemmas, Theorem 1 holds:

Algorithm 2: FedSPU with Early Stopping (FedSPU + ES)

Require: maximum global iteration T, clients $\mathbb{C} = \{1, ..., N\}$, initial global model w^0.

Server broadcasts w_0 to all clients.
For round $t = 1, 2, ..., T$:
Server executes:
  randomly sample a subset of clients $\mathbb{C}_t \subset \mathbb{C}$.
  $\forall k \in \mathbb{C}_t$:
    randomly sample $A_k(w^t)$ based on p_k.
    send $A_k(w^t)$ to k.
Each client $k \in \mathbb{C}_t$ **in parallel does**:
  merge $A_k(w^t)$ into w_k^t to get $\hat{w}_k^t$. ▷ see Fig. 2b
  local SGD: $w_k^{t+1} = \hat{w}_k^t - \eta \nabla \bar{F}_k(\hat{w}_k^t)$. ▷ see Eq. (3)
  compute $\mathcal{L}_t$. ▷ see Eq. (5)
  If $\mathcal{L}_t > \mathcal{L}_{t-1}$:
    status $\leftarrow$ *stopped*
  Else:
    status $\leftarrow$ *on*
  send $A_k(w_k^{t+1})$ and *status* to the server.
Server executes:
  $\forall k \in \mathbb{C}_t$:
    receive $A_k(w_k^{t+1})$, *status* from k.
    If *status* == *stopped*:
      remove k from $\mathbb{C}$.
  Aggregate all $A_k(w_k^{t+1})$ to get w^{t+1}. ▷ see Fig. 3
  If $\mathbb{C} == \varnothing$:
    TERMINATE.
return $w_1, w_2, ..., w_N$.

Theorem 1. *When the learning rate η satisfies $\eta < \frac{1+\sqrt{1-\frac{QL}{p_k^2}}}{L}$, every local model w_k will at least reach a ϵ-critical point w_k^ϵ (i.e. $\|\nabla F_k(w_k^\epsilon)\| \leq \epsilon$) in $O(\frac{w_k^0 - w_k^\epsilon}{\epsilon\eta})$ rounds, with $\epsilon = \sqrt{\frac{(L+1)Q\sigma^2}{(2\eta - L\eta^2)p_k^2 + Q}}$.*

According to Theorem 1, in FedSPU, each client's local objective function will converge to a relatively low value, given that the learning rate is small enough. This means that every client's personal model will eventually acquire favorable performance on the local dataset even if the objective function is not necessarily convex. The proofs can be found in the Appendix.

Experiment

Experiment Setup

Datasets and models. We evaluate FedSPU on three real-world datasets that are very commonly used in the state-of-the-art, including: **Extended MNIST (EMNIST)** contains 814,255 images of human-written digits/characters from 62 categories (numbers 0-9 and 52 upper/lower-case English letters). Each sample is a black-and-white-based image with 28×28 pixels (Cohen et al. 2017). **CIFAR10** contains 50,000 images of real-world objects across 10 categories. Each sample is an RGB-based colorful image with 32×32 pixels (Krizhevsky et al. 2009). **Google Speech** is an audio dataset containing 101,012 audio commands from more

Dataset	PruneFL	FjORD	Hermes	FedMP	FedSelect	FedSPU
EMNIST	67.34 ± 0.9	7.66 ± 0.4	69.09 ± 0.6	67.42 ± 0.9	66.26 ± 1.7	**73.42**± 0.4
CIFAR10	36.65 ± 2.0	24.95 ± 2.2	40.52 ± 4.7	33.48 ± 1.5	47.83 ± 1.1	**51.81**± 1.5
Google Speech	21.5 ± 2.0	11.20 ± 7.0	32.03 ± 3.0	21.08 ± 1.7	34.06 ± 2.1	**39.1**± 2.8

Table 1: Mean final test accuracy (%) across three Dirichlet distributions with parameters 0.1, 0.5 and 1.0 (**without ES**).

Dataset	PruneFL	FjORD	Hermes	FedMP	FedSelect	FedSPU
EMNIST	62.8 ± 3.8	0.08 ± 0.3	68.83 ± 1.2	63.0 ± 2.7	62.3 ± 4.9	**73.31**± 0.2
CIFAR10	31.2 ± 2.1	21.2 ± 3.2	30.6 ± 3.3	26.9 ± 0.5	37.85 ± 0.3	**42.66**± 1.5
Google Speech	16.7 ± 2.0	10.68 ± 10.2	29.66 ± 2.0	16.6 ± 2.2	19.2 ± 2.6	**35.7**± 1.6

Table 2: Mean test accuracy (%) across three Dirichlet distributions with parameters 0.1, 0.5 and 1.0 (**with ES**).

than 2,000 speakers. Each sample is a human-spoken word belonging to one of the 35 categories (Warden 2018). For EMNIST and Google Speech, a convolutional neural network (CNN) with two convolutional layers and one fully-connected layer is used, following the setting of (Horváth et al. 2021). For CIFAR10, a CNN with two convolutional layers and three fully-connected layers is used, following the setting of (Li et al. 2021).

We conduct three runs on each dataset. For each run, data are allocated to clients unevenly following the settings of (Acar et al. 2021; Luo et al. 2021), following a *Dirichlet distribution* with parameter α. We tune the value of α with 0.1, 0.5 and 1.0 to create three different distributions for each run. We split each client's dataset into a training set and a testing set with the split factor $\lambda = 0.7$.

Baselines. We compare FedSPU with five typical methods: **FjORD (Horváth et al. 2021):** The server prunes neurons in a fixed right-to-left order. **FedSelect (Tamirisa et al. 2024):** All clients start with a small sub-model and gradually expand it. **FedMP (Jiang et al. 2023):** Each client k locally prunes neurons to create a personal sub-model. In each layer, $1 - p_k$ of the neurons with the least importance scores are pruned. The importance of a neuron is defined as the *l1*-norm of the parameters. **Hermes (Li et al. 2021):** Similar to FedMP, each client k locally prunes the $1 - p_k$ least important neurons in each layer. The importance of a neuron is defined as the *l2*-norm of the parameters. **PruneFL (Jiang et al. 2022):** Similarly, each client k locally prunes the $1-p_k$ least important neurons in each layer. The importance of a neuron is defined as the *l2*-norm of the neuron's gradient.

Parameter settings and system implementation. The maximum global iteration is set to $T = 500$ with a total of $M = 100$ clients. The number of active clients per round is set to 10, and each client has five local training epochs (Horváth et al. 2021). For ES, if the number of non-stopped clients is less than 10, then all non-stopped clients will be selected. The learning rate is set to 2e-4, 5e-4 and 0.1 respectively for EMNIST, Google Speech and CIFAR10. The batch size is set to 16 for EMNIST and Google Speech, and 128 for CIFAR10. The experiment is implemented with Pytorch 2.0.0 and the Flower framework (Beutel et al. 2022). The server runs on a desktop computer and clients run on NVIDIA Jetson Nano Developer Kits with one 128-core Maxwell GPU and 4GB 64-bit memory. For the emulation of system heterogeneity and resource constraints, we divide the clients into 5 uniform clusters following (Horváth et al. 2021). Clients of the same cluster share the same value of p_k. The values of p_k for the five clusters are 0.2, 0.4, 0.6, 0.8 and 1.0 respectively. For FedSelect, the initial and final values of p_k are set to 0.25 and 0.5 (Tamirisa et al. 2024).

Experiment Results

Accuracy. FedSPU obtains higher final accuracy than dropout as Table 1 shows. On average, FedSPU improves the final test accuracy by 4.45% compared with the best results of dropout (Hermes). These results prove the usefulness of local full models in preserving personalization.

Computation and communication overheads. To assess computation overhead, we focus on the wall-clock training time rather than floating point operations out of practical concern (Bonawitz et al. 2019; He and Sun 2015; Li et al. 2022). As shown by Table 3, the additional computation overhead caused by FedSPU is minor. Among all cases, the training time of FedSPU is less than $1.11\times$ that of the fastest baseline. Moreover, FedSPU does not incur extra communication overhead compared with dropout. As shown in Table 3, there is very little difference between the size of the transmitted parameters in FedSPU and dropout.

Effect of the early stopping strategy. With early stopping (ES), the number of training rounds of FedSPU is reduced by $25\% \sim 71\%$ as shown in Table 4. Correspondingly, the total computation and communication overheads are also reduced as shown in Table 3.

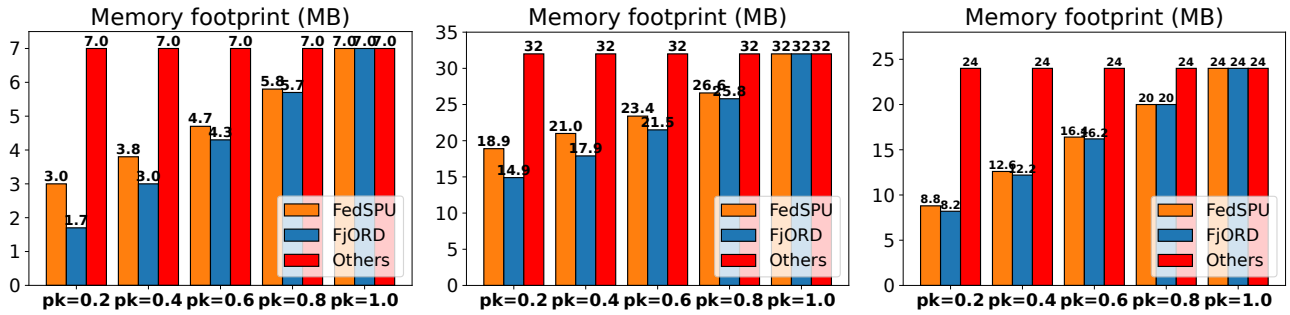
As shown in Tables 1 and 4, for EMNIST and Google Speech, ES effectively alleviates the computation/communication cost by $25\% \sim 59\%$ in FedSPU with a marginal accuracy sacrifice of 0.11% and 3.4%. For CIFAR10, the ES strategy becomes more aggressive, reducing the cost by $57\% \sim 71\%$ with 9.2% of accuracy loss. Despite this, the final accuracy of FedSPU+ES is still higher than that of dropout in most cases as shown in Tables 1 and 2.

For fairness, we also evaluate the baselines' performance with ES. As shown in Table 2, FedSPU consistently obtains the highest accuracy, demonstrating better compatibility with the ES mechanism. Overall, FedSPU improves the mean accuracy by at least 5.11% in the presence of ES.

Memory footprint. We calculate the memory footprint by (Pfeiffer, Khalili, and Henkel 2023), where the total

Total time of local training (in hours)							
Dataset	PruneFL	FjORD	Hermes	FedMP	FedSelect	FedSPU	FedSPU+ES
EMNIST	8.11	7.85	8.57	7.30	7.68	7.82	**4.0**
CIFAR10	24.69	25.03	25.18	25.5	25.06	25.05	**8.9**
Google Speech	8.29	7.83	8.09	8.16	8.03	8.71	**5.97**
Total size of parameter transmission (in GB)							
EMNIST	11.69	11.72	11.73	11.66	20.6	11.71	**7.23**
CIFAR10	18.20	18.17	18.2	18.43	20.7	18.04	**6.37**
Google Speech	4.39	4.36	4.36	4.36	8.39	4.36	**2.99**

Table 3: Comparison of the total training time (hours) and the size of transmitted parameters for $T = 500$ rounds.



(a) EMNIST. (b) CIFAR10. (c) Google Speech.

Figure 5: Comparison of memory footprint (MB) with different p_k. "Others" stand for FedMP, Hermes, PruneFL and FedSelect.

Dataset	Rounds	
	FedSPU	FedSPU+ES
EMNIST ($\alpha = 0.1$)	500	207
EMNIST ($\alpha = 0.5$)	500	374
EMNIST ($\alpha = 1.0$)	500	346
CIFAR10 ($\alpha = 0.1$)	500	171
CIFAR10 ($\alpha = 0.5$)	500	216
CIFAR10 ($\alpha = 1.0$)	500	147
Google Speech ($\alpha = 0.1$)	500	306
Google Speech ($\alpha = 0.5$)	500	358
Google Speech ($\alpha = 1.0$)	500	365

Table 4: Comparison of final accuracy and estimated computation and communication cost (combined) between FedSPU and FedSPU with early stopping.

memory footprint equals the accumulated size of weights, gradients, and activations stored in the model. As Figure 5 shows, FedSPU significantly reduces the memory footprint compared with FedSelect, PruneFL, FedMP and Hermes ("Others" in Figure 5) which require full-model training, achieving average 54%, 44%, 31% and 18% reduction with $p_k = 0.2, 0.4, 0.6, 0.8$ respectively.

Conclusion

We propose FedSPU, a novel personalized federated learning approach with stochastic parameter update. FedSPU preserves the global model architecture on each edge device, randomly freezing portions of the local model based on device capacity, training the remaining segments with local data, and subsequently updating the model based solely on the trained segments. This methodology ensures that a segment of the local model remains personalized, thereby mitigating the adverse effects of biased parameters from other clients. We also propose to combine FedSPU with early stopping to mitigate the training iterations, which further reduces the overall computation and communication costs while maintaining high accuracy. In the future, we plan to explore the similarities of local clients in a privacy-preserving way, leveraging techniques such as learning vector quantization (Qin and Suganthan 2005) and graph matching (Gong et al. 2016) to guide the model freezing process and enhance local model training. Furthermore, we intend to extend FedSPU to traditional FL problems, and enhance the generalization capability of the global model.

Acknowledgements

This work is funded by the Australian Research Council under Grant No. DP220101823, DP200102611, and LP180100114.

References

Abeysekara, P.; Dong, H.; and Qin, A. K. 2021. Data-Driven Trust Prediction in Mobile Edge Computing-Based IoT Systems. *IEEE Transactions on Services Computing*.

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*.

Arivazhagan, M. G.; Aggarwal, V.; Singh, A. K.; and Choudhary, S. 2019. Federated Learning with Personalization Layers. arXiv:1912.00818.

Beutel, D. J.; Topal, T.; Mathur, A.; Qiu, X.; Fernandez-Marques, J.; Gao, Y.; Sani, L.; Li, K. H.; Parcollet, T.; de Gusmão, P. P. B.; and Lane, N. D. 2022. Flower: A Friendly Federated Learning Research Framework. arXiv:2007.14390.

Bonawitz, K.; Eichner, H.; Grieskamp, W.; Huba, D.; Ingerman, A.; Ivanov, V.; Kiddon, C.; Konečný, J.; Mazzocchi, S.; McMahan, B.; Van Overveldt, T.; Petrou, D.; Ramage, D.; and Roselander, J. 2019. Towards Federated Learning at Scale: System Design. In Talwalkar, A.; Smith, V.; and Zaharia, M., eds., *Proceedings of Machine Learning and Systems*, volume 1, 374–388.

Caldas, S.; Konečny, J.; McMahan, H. B.; and Talwalkar, A. 2018. Expanding the Reach of Federated Learning by Reducing Client Resource Requirements. In *NeurIPS Workshop on Federated Learning for Data Privacy and Confidentiality*.

Cohen, G.; Afshar, S.; Tapson, J.; and van Schaik, A. 2017. EMNIST: an extension of MNIST to handwritten letters. *arXiv preprint arXiv:1702.05373*.

Deng, Y.; Kamani, M. M.; and Mahdavi, M. 2021. Adaptive Personalized Federated Learning.

Dinh, C. T.; Tran, N. H.; and Nguyen, T. D. 2020. Personalized Federated Learning with Moreau Envelopes. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20. Red Hook, NY, USA: Curran Associates Inc. ISBN 9781713829546.

Fallah, A.; Mokhtari, A.; and Ozdaglar, A. 2020. Personalized Federated Learning: A Meta-Learning Approach. arXiv:2002.07948.

Gong, M.; Wu, Y.; Cai, Q.; Ma, W.; Qin, A. K.; Wang, Z.; and Jiao, L. 2016. Discrete particle swarm optimization for high-order graph matching. *Information Sciences*, 328: 158–171.

He, K.; and Sun, J. 2015. Convolutional neural networks at constrained time cost. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5353–5360.

Horváth, S.; Laskaridis, S.; Almeida, M.; Leontiadis, I.; Venieris, S.; and Lane, N. 2021. FjORD: Fair and Accurate Federated Learning under heterogeneous targets with Ordered Dropout. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*, volume 34, 12876–12889. Curran Associates, Inc.

Imteaj, A.; Thakker, U.; Wang, S.; Li, J.; and Amini, M. H. 2022. A Survey on Federated Learning for Resource-Constrained IoT Devices. *IEEE Internet of Things Journal*, 9(1): 1–24.

Jiang, Y.; Wang, S.; Valls, V.; Ko, B. J.; Lee, W.-H.; Leung, K. K.; and Tassiulas, L. 2022. Model pruning enables efficient federated learning on edge devices. *IEEE Transactions on Neural Networks and Learning Systems*.

Jiang, Z.; Xu, Y.; Xu, H.; Wang, Z.; Liu, J.; Chen, Q.; and Qiao, C. 2023. Computation and Communication Efficient Federated Learning With Adaptive Model Pruning. *IEEE Transactions on Mobile Computing*, 1–18.

Khodak, M.; Balcan, M.-F. F.; and Talwalkar, A. S. 2019. Adaptive gradient-based meta-learning methods. *Advances in Neural Information Processing Systems*, 32.

Krizhevsky, A.; et al. 2009. Learning multiple layers of features from tiny images. *University of Toronto*.

Kulkarni, V.; Kulkarni, M.; and Pant, A. 2020. Survey of Personalization Techniques for Federated Learning. arXiv:2003.08673.

Li, A.; Sun, J.; Li, P.; Pu, Y.; Li, H.; and Chen, Y. 2021. Hermes: An Efficient Federated Learning Framework for Heterogeneous Mobile Clients. In *Proceedings of ACM MobiCom*, 420–437. New York, NY, USA: Association for Computing Machinery. ISBN 9781450383424.

Li, C.; Zeng, X.; Zhang, M.; and Cao, Z. 2022. PyramidFL: A Fine-Grained Client Selection Framework for Efficient Federated Learning. In *Proceedings of ACM MobiCom*, 158–171. New York, NY, USA: Association for Computing Machinery. ISBN 9781450391818.

Li, S.; Zhao, Y.; Varma, R.; Salpekar, O.; Noordhuis, P.; Li, T.; Paszke, A.; Smith, J.; Vaughan, B.; Damania, P.; and Chintala, S. 2020. PyTorch Distributed: Experiences on Accelerating Data Parallel Training. *Proc. VLDB Endow.*, 13: 3005–3018.

Luo, M.; Chen, F.; Hu, D.; Zhang, Y.; Liang, J.; and Feng, J. 2021. No Fear of Heterogeneity: Classifier Calibration for Federated Learning with Non-IID Data. In Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*.

Mansour, Y.; Mohri, M.; Ro, J.; and Suresh, A. T. 2020. Three Approaches for Personalization with Applications to Federated Learning. arXiv:2002.10619.

Marfoq, O.; Neglia, G.; Bellet, A.; Kameni, L.; and Vidal, R. 2021. Federated Multi-Task Learning under a Mixture of Distributions. In Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and Arcas, B. A. y. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Artificial Intelligence and Statistics (AISTAS)*, 1273–1282.

Niu, Z.; Dong, H.; Qin, A. K.; and Gu, T. 2024. FLrce: Resource-Efficient Federated Learning with Early-Stopping Strategy. arXiv:2310.09789.

Padurariu, C.; and Breaban, M. E. 2019. Dealing with Data Imbalance in Text Classification. *Procedia Computer Science*, 159: 736–745. Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 23rd International Conference KES2019.

Pfeiffer, K.; Khalili, R.; and Henkel, J. 2023. Aggregating Capacity in FL through Successive Layer Training for Computationally-Constrained Devices. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Prechelt, L. 2002. Early Stopping - But When? In *Neural Networks: Tricks of the trade*, 55–69. Springer.

Puyol-Antón, E.; Ruijsink, B.; Piechnik, S. K.; Neubauer, S.; Petersen, S. E.; Razavi, R.; and King, A. P. 2021. Fairness in Cardiac MR Image Analysis: An Investigation of Bias Due to Data Imbalance in Deep Learning Based Segmentation. In de Bruijne, M.; Cattin, P. C.; Cotin, S.; Padoy, N.; Speidel, S.; Zheng, Y.; and Essert, C., eds., *Medical Image Computing and Computer Assisted Intervention – MICCAI 2021*, 413–423. Cham: Springer International Publishing. ISBN 978-3-030-87199-4.

Qin, A. K.; and Suganthan, P. N. 2005. Initialization insensitive LVQ algorithm based on cost-function adaptation. *Pattern Recognition*, 38(5): 773–776.

Sattler, F.; Müller, K.-R.; and Samek, W. 2021. Clustered Federated Learning: Model-Agnostic Distributed Multitask Optimization Under Privacy Constraints. *IEEE Transactions on Neural Networks and Learning Systems*, 32(8): 3710–3722.

Smith, V.; Chiang, C.-K.; Sanjabi, M.; and Talwalkar, A. S. 2017. Federated multi-task learning. *Advances in neural information processing systems*, 30.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56): 1929–1958.

Tamirisa, R.; Xie, C.; Bao, W.; Zhou, A.; Arel, R.; and Shamsian, A. 2024. FedSelect: Personalized Federated Learning with Customized Selection of Parameters for Fine-Tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 23985–23994.

Wang, L.; Han, M.; Li, X.; Zhang, N.; and Cheng, H. 2021. Review of Classification Methods on Unbalanced Data Sets. *IEEE Access*, 9: 64606–64628.

Wang, Y.; Xu, H.; Ali, W.; Li, M.; Zhou, X.; and Shao, J. 2023. FedFTHA: A Fine-Tuning and Head Aggregation Method in Federated Learning. *IEEE Internet of Things Journal*, 10(14): 12749–12762.

Warden, P. 2018. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *arXiv preprint arXiv:1804.03209*.

Wen, D.; Jeon, K.-J.; and Huang, K. 2022. Federated Dropout – A Simple Approach for Enabling Federated Learning on Resource Constrained Devices. arXiv:2109.15258.

Zhang, J.; Hua, Y.; Wang, H.; Song, T.; Xue, Z.; Ma, R.; and Guan, H. 2023. FedALA: Adaptive Local Aggregation for Personalized Federated Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37: 11237–11244.

Zhang, M.; Sapra, K.; Fidler, S.; Yeung, S.; and Alvarez, J. M. 2021. Personalized Federated Learning with First Order Model Optimization. In *International Conference on Learning Representations*.