FedMut: Generalized Federated Learning via Stochastic Mutation

Ming Hu[1], Yue Cao[1], Anran Li[1], Zhiming Li[1], Chengwei Liu[1], Tianlin Li[1], Mingsong Chen[2], Yang Liu[1]

[1]School of Computer Science and Engineering, Nanyang Technological University, Singapore
[2]MoE Engineering Research Center of SW/HW Co-Design Tech. and App., East China Normal University, China
ecnu_hm@163.com, {CAOY0033, anran.li, zhiming001}@ntu.edu.sg, lcwj3ntu@gmail.com, tianlin001@ntu.edu.sg, mschen@sei.ecnu.edu.cn, yangliu@ntu.edu.sg

Abstract

Although Federated Learning (FL) enables collaborative model training without sharing the raw data of clients, it encounters low-performance problems caused by various heterogeneous scenarios. Due to the limitation of dispatching the same global model to clients for local training, traditional Federated Average (FedAvg)-based FL models face the problem of easily getting stuck into a sharp solution, which results in training a low-performance global model. To address this problem, this paper presents a novel FL approach named FedMut, which mutates the global model according to the gradient change to generate several intermediate models for the next round of training. Each intermediate model will be dispatched to a client for local training. Eventually, the global model converges into a flat area within the range of mutated models and has a well-generalization compared with the global model trained by FedAvg. Experimental results on well-known datasets demonstrate the effectiveness of our FedMut approach in various data heterogeneity scenarios.

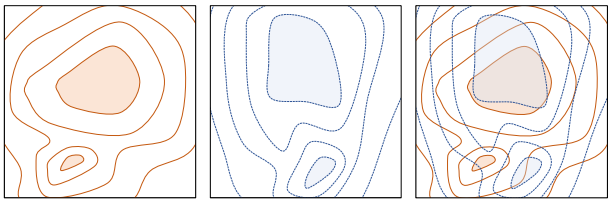
Introduction

As a well-known distribution machine learning paradigm, Federated Learning (FL) (McMahan et al. 2017; Li et al. 2021c; Hu et al. 2023b; Wang et al. 2023) has been widely used in various applications, such as Internet of Things (IoT) systems (Zhang et al. 2020b; Li et al. 2021a; Hu et al. 2023a; Jia et al. 2023; Cui et al. 2021), medical health (Rieke et al. 2020), and digital finance (Long et al. 2020). Traditional FL methods are based on the Federated Averaging (FedAvg) mechanism, which dispatches a global model to multiple local clients and aggregates their trained local models to update the global model. In this way, FL can enable collaborative machine learning without sharing data. Furthermore, to prevent privacy problems caused by gradient leakage, FedAvg adopts a secure aggregation mechanism (Bonawitz et al. 2017).

Although FL is promising in protecting the privacy, its performance is still seriously limited by data heterogeneity. Specifically, since the data on local clients are non-IID (Independent and Identically Distributed) (Huang et al. 2021), the optimization directions of local clients are divergent, resulting in the notorious "gradient divergence" problem (Karimireddy et al. 2020). Since the traditional FedAvg method only coarse-grained aggregates all the received local models to generate a global model, the conflict weights among local models cannot be wisely resolved. Thus, the global model trained by FedAvg may easily be worse-generalized. In other words, FL training may fall into a worse-generalized local optimum. To address this issue, state-of-the-art FL optimization methods attempt to use global variables (Karimireddy et al. 2020; Li et al. 2020), client grouping strategies (Fraboni et al. 2021; Chen et al. 2020), or knowledge distillation technologies (Lin et al. 2020; Zhu, Hong, and Zhou 2021). Although these methods can slightly improve the inference performance of the global model, they still have serious limitations, e.g., requirements of additional data or not compatible with the secure aggregation mechanism. Therefore, *how to improve the inference performance of secure FL without additional data has become an important challenge.*

From the perspective of geometric, due to the data heterogeneity, the loss landscapes (Hochreiter and Schmidhuber 1997) of different clients are slightly different, where a loss landscape shows the loss values of a target model with different parameter values on a specific dataset. Specifically, a model may achieve a low inference loss in a client, but may suffer from a higher inference loss in other clients. Therefore, to train a high-performance global model, it is important to optimize all the local models towards a more generalized direction. Many previous works (Hochreiter and Schmidhuber 1997; Stutz, Hein, and Schiele 2021; Foret et al. 2020) observed that the well-generalized solutions are located in flat areas rather than sharp areas of the loss landscapes. Typically, since the tasks of each local model are the same, the loss landscapes of each client are still similar. Intuitively, there is a greater probability that the flat optimal areas of different clients partially overlap compared to the sharp optimal areas. In other words, when the model converges into the overlapping region, it can achieve a high inference performance in the majority of clients.

Figure 1 illustrates the loss landscapes of two clients (i.e., Client 1 and Client 2) with heterogeneous data. Figure 1(a) shows the loss landscapes of Client 1 and Figure 1(b) shows that of Client 2. Each client has two optimal areas, i.e., the sharp one (located at the bottom of the figure) and the flat one (located at the top of the figure), respectively. Fig-



(a) Client 1 (b) Client 2 (c) Overlap

Figure 1: A motivating example of Loss Landscapes.

ure 1(c) presents the overlap of loss landscapes of the two clients. From Figure 1(c), we can observe that the loss landscapes of the two clients are similar. The two sharp optimal areas do not overlap, but the two flat optimal areas overlap partially. When a model converges into one of the sharp optimal areas of two clients, it will achieve low performance in the other client. On the contrary, when the model converges into the overlapping region of two flat optimal areas, it will achieve high performance in both clients. Intuitively, we can guide the FL training towards a flat optimal area to achieve a well-generalized global model.

Based on the above intuition, we present a novel FL approach named *FedMut*, which uses a heuristic mutation strategy to generate multiple models for local training to guide the aggregated global model towards a flat optimal area. Specifically, we mutate the aggregated gradients of the global model to generate multiple mutated models for local training. The generated models are located in the neighborhood of the global model. If the neighborhood is located in the same flat area, with the local training, the generated models are still optimized towards the optimal solution of this area. Otherwise, some local models are optimized towards another area, which can guide the aggregated global model to jump out of the current sharp area. Therefore, the global model will eventually converge into a flat area, where the flat area is larger than the neighborhood composed of mutation models. Additionally, since the mutation strategy of *FedMut* only uses the aggregated gradients of the global model rather than each uploaded local model, *FedMut* is compatible with the security aggregation mechanism. The main contributions of this paper are shown as follows:

- We present a novel FL approach *FedMut*, which uses multiple mutated models rather than the global model for local training to guide the global model to converge into a flat optimal solution.
- We present a mutation strategy to generate multiple mutated models by stochastic mutating the updated gradients of the global model.
- We conducted experiments using various models on three well-known datasets to demonstrate the effectiveness of our *FedMut* approach in both IID and non-IID scenarios.

Preliminaries

Problem Definition of Federated Learning

Typically, FL adopts a cloud-based architecture, which consists of a cloud server and multiple clients. Note that this paper only focuses on the scenario of homogeneous models, where the global model and all the local models have the same model structure.

Assume that the task of the global model and the local models of an FL system is to map an input space I to an output space O. Assume that the FL system has a cloud server S and N clients indicated by $\{c_1, c_2, ..., c_N\}$. Let each client i hold a local dataset, which is denoted by $D_i = \{d_{i,1}, d_{i,2}, \cdots, d_{i,n_i}\}$, where $d_{i,j} = (I_{i,j}, O_{i,j}) \in I \times O$. In the FL system, all involved clients collaboratively train a global model w_{glb} by uploading their trained local models to the cloud server. So far, standard FL methods aggregate local models based on FedAvg (McMahan et al. 2017) to generate the global model. The goal of a standard FL optimization problem is formulated as follows,

$$\begin{aligned} &\min_{w} F(w) = \frac{1}{N}\sum_{i=1}^{N} f_i(w), \\ &s.t.,\ f_i(w) = \frac{1}{n_i}\sum_{j=1}^{n_i} l(w; d_{i,j}), \end{aligned} \tag{1}$$

where l denotes the loss functions of an individual sample (e.g., the cross-entropy loss), f_i denotes the loss functions of all the samples of client i, and F represents the loss function of the global model. Note that we focuse on the inference performance of the unified global model. The personalized and clustered FL problems are not considered in this paper.

Related Work

To enhance the traditional FL inference performance, many FL optimization methods have been presented. Specifically, these methods can be classified into three categories, i.e., global variable-based FL methods (Li et al. 2020; Karimireddy et al. 2020), device grouping-based FL methods (Fraboni et al. 2021; Chen et al. 2020; Li et al. 2021b), and knowledge distillation-based FL methods (Sattler et al. 2021; Lin et al. 2020; Zhu, Hong, and Zhou 2021).

The global variable-based FL methods typically use a global variable to guide local training towards a similar direction. For example, FedProx (Li et al. 2020) uses the squared distance between the local models and the global model as a proximal term to regularize the local loss functions, thus stabilizing the convergence of the model. SCAFFOLD (Karimireddy et al. 2020) generates a global control variable in each FL round and uses this global control variable to correct the optimization direction of local models in the local training process.

The device grouping-based FL methods attempt to group devices according to specific metrics and then wisely select devices from different groupings for local training. Typically, different from the consideration of other trustworthy properties (Huang et al. 2023; Zhang et al. 2020a; Li et al. 2023a, 2021d, 2023b; Zhang et al. 2023; Yue et al. 2023), it is difficult to directly obtain the data distributions of each client due to the consideration of privacy protection. Thus, the majority of device grouping-based methods use model similarity to the grouping metrics. For example, CluSamp (Fraboni et al. 2021) uses sample size or model

similarity as metrics for client grouping. FedCluster (Chen et al. 2020) groups clients into multiple groups that perform FL cyclically in each FL round.

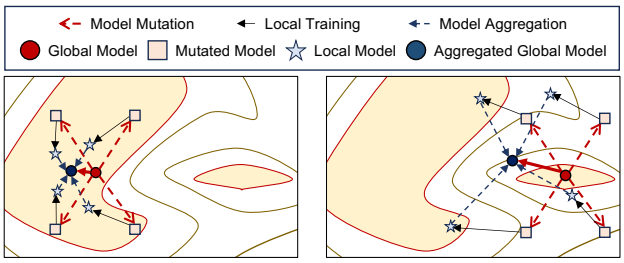
The knowledge distillation-based methods adopt the well-known knowledge distillation technology to enhance the inference performance of FL. Specifically, they use a well-performed "teacher model" to guide the training of "student models". For example, FedAUX (Sattler et al. 2021) performs data-dependent distillation using an auxiliary dataset to initialize the server model. FedDF (Lin et al. 2020) uses an ensemble model as the "teacher model" and unlabeled data for distillation to accelerate FL training. To address the problem of the requirement for additional datasets, FedGen (Zhu, Hong, and Zhou 2021) uses a built-in generator and a proxy dataset to achieve data-free distillation.

Although the above optimization FL methods can improve the inference performance of FL, they still have serious limitations. For global variable-based methods, they need additional communication overhead to dispatch global variables or additional computing overhead on clients to calculate the proximal term. For device grouping-based methods, they need to acquire all the local models, which results in FL incompatibility with the secure aggregation mechanism, thus causing potential privacy leakage risks. For knowledge distillation-based methods, they need additional computing overhead for knowledge distillation and additional datasets. To the best of our knowledge, *FedMut* is the first attempt to use a general mutation strategy to generate multiple mutated models for local training. By guiding all the mutated models to be optimized towards the same flat area, *FedMut* can train a well-generalized global model. Note that *FedMut* is compatible with the secure aggregation mechanism and does not require additional datasets or communication overhead.

Our FedMut Approach

Motivation

Inspired by the findings of various empirical studies (Hochreiter and Schmidhuber 1997; Stutz, Hein, and Schiele 2021; Cha et al. 2021; Foret et al. 2020) that well-generalized solutions are located in a flat area rather than a sharp one, our goal is to guide FL training towards a flat area to train a well-generalized global model. Intuitively, if a global model is located in a flat area, the models located in its neighborhoods are still in the same flat area with a higher probability. On the contrary, if a model is located in a sharp area, the models located in its neighborhoods may be in the other area. Based on this intuition, we mutate the global model to generate multiple models in its neighborhoods for local training. When all the mutated models are located in the same area, we can obtain a well-generalized global model, whose neighborhoods are flat. Note that, since all the mutated models are located in the same flat area, when local training, all these models are optimized towards the optimal solution of such area, and the aggregated global model is still in such flat area. Therefore, the global model eventually converges into such a flat area. On the contrary, traditional FedAvg-based methods that use the global model for training are easily stuck in a sharp area.



(a) Mutation in Flat Area (b) Mutation in Sharp Area

Figure 2: A motivating example of mutation.

Figure 2 illustrates an example of our motivation. The two subfigures show the same loss landscape of the global model. Here, the landscape has two optimal areas, i.e., a flat area on the left of each subfigure and a sharp area on the right of each subfigure, respectively. Figure 2(a) presents the example that the global model is located in a flat area, and Figure 2(b) presents the example that the global model is located in a sharp area. In each subfigure, the red dot denotes the initial global model, the red squares denote the mutated models, the blue stars present the trained local models, and the blue dot indicates the aggregated global model. We assume that four clients are involved in local training in each FL round. As seen in Figure 2(a), the cloud server generates four mutated models through the model mutation process. Since the neighborhood composed of mutation models is smaller than the flat area, where the initial global model is located, all the mutated models are still located in this flat area. Through local training, all the mutated models are optimized towards the center of the flat area, and the aggregated global model is consequently updated to the center of the flat area. As shown in Figure 2(b), since the neighborhood composed of mutation models is larger than the flat area, the three mutated models are located in the other area rather than the sharp area, where the initial global model is located. Through local training, three mutated models are optimized towards the flat area, and only one mutated model is optimized sharp area. Consequently, the aggregated global model moves towards the flat area. Therefore, by mutating the global model to multiple mutated models in neighborhoods and using such mutated models for local training, we can eventually obtain a well-generalized global model, which is located in a flat area larger than the neighborhoods composed of all the mutated models.

Overview

Figure 3 presents the framework of our FedMut approach, which consists of a cloud server and N involved clients. In each FL round, the cloud server randomly selects K activated clients for local training. Note that since real-work FL systems are difficult to ensure that all the involved clients participate in local training of each FL round, typically the number of activated clients is smaller than that of involved clients. As shown in Figure 3, each FedMut training round

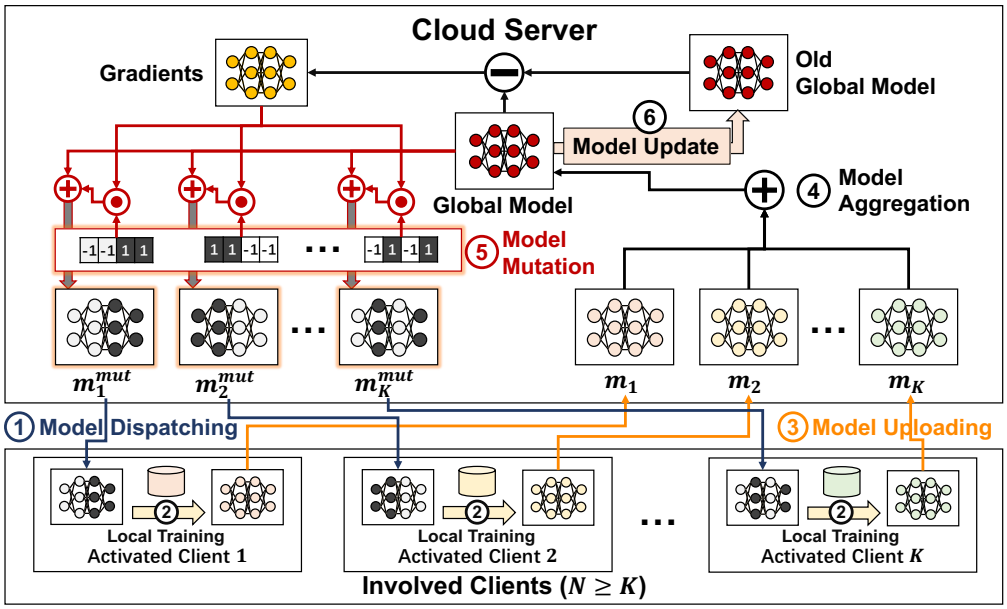


Figure 3: Framework of FedMut.

consists of six steps, i.e., model dispatching, local training, model uploading, model aggregation, model mutation, and model updating, respectively. The details of the six steps are as follows. ❶ **Step 1 (Model Dispatching):** The cloud server dispatches K mutated global models to K activated clients. Note that each activated client is dispatched only one mutated model. ❷ **Step 2 (Local Training):** Each activated client trains the received mutated global model using their raw data. ❸ **Step 3 (Model Uploading):** Each activated client uploads its trained local model to the cloud server. ❹ **Step 4 (Model Aggregation):** Similar to conventional FL methods, when receiving local models from all the activated clients, the cloud server aggregates these local models to generate a new global model. ❺ **Step 5 (Model Mutation):** The cloud server mutates the aggregated global model to generate K mutated models according to the gradients of the global model. Specifically, the gradients of the global model are the weight changes of the global model relative to the old global model from the previous round. Our FedMut approach performs a layer-wise mutation strategy, which stochastically generates a one-dimensional matrix whose element values are only -1 and 1 and whose length is the number of model layers (denotes L). According to the matrix, for the i^{th} layer of the global model, the cloud server adds the weights of the layer to the value of the corresponding gradients for the value of the i^{th} element equal to 1 and subtracts the weights of the corresponding layer to the value of the corresponding gradients for the value of the i^{th} element equal to -1. Note that the cloud server will generate K different mutation matrices to generate K mutated models. These mutated models will be dispatched for local training in the next round. ❻ **Step 6 (Model Updating):** The cloud server replaces the old global model with the aggregated global model in the current round.

After a certain number of FL training rounds, the cloud server can obtain a well-generalized global model, which will be deployed to AIoT devices for inference tasks. Note that in our approach, all the mutated models are only used for local training rather than model deployment.

Algorithm 1: Implementation of FedMut

Input:
i) $round$, # of training rounds; ii) C, the set of involved clients; iii) K, # of activated clients.
Output:
i) w_{glb}, the trained global model
FedMut($round$,C,K)

1: $w_{glb}^0 \leftarrow w_0$ // initialize global model
2: $w_1^{mut}, ..., w_K^{mut} \leftarrow w_{glb}^0$ // initialize mutated models
3: **for** r = 1, ..., $round$ **do**
4: $L_c \leftarrow$ Randomly select K clients from C
/*parallel for block*/
5: **for** i = 1, ..., K **do**
6: $w_i^r \leftarrow$*LocalTraining*$(w_i^{mut}, L_c[i])$
7: **end for**
8: $w_{glb}^r \leftarrow$*ModelAggregation*$(w_1^r, ..., w_K^r)$
9: $[w_1^{mut}, ..., w_K^{mut}] \leftarrow$*ModelMutation*$(w_{glb}^r, w_{glb}^{r-1}, K)$
10: **end for**
11: **return** w_{glb}^r

Implementation

Algorithm 1 shows the implementation of our FedMut approach. Line 1 initializes the global model. Line 2 initializes all the mutated models using the initial global model. This is because, in the first round, the cloud server cannot obtain

the gradients of the global model and thus cannot perform the model mutation process. Lines 3-10 present the training process of each FL round. In Line 4, the cloud server randomly selects K clients from the set of involved clients C as activated clients for local training. Lines 5-7 present the local training process. The cloud server parallel dispatches the K mutated models to K selected clients. Specifically, in Line 6, the cloud server dispatches the mutated model w_i^{mut} to the i^{th} selected client $L_c[i]$ and the client $L_c[i]$ performs the function *LocalTraining(·)* to train w_i^{mut} using its raw data. After local training, the client $L_c[i]$ uploads the trained model w_i^r to the server. Line 8 presents the model aggregation process. In Line 9, the function *ModelMutation(·)* mutates the global model w_{glb}^r according to the old global model w_{glb}^{r-1} generated in the last round to generate K mutated models for the local training of the next round.

Model Mutation To guide the global model to converge into a flatter solution, FedMut mutates the global model based on its gradients to generate multiple mutated models located in the neighborhood of the global model for local training. Our model mutation strategy mutates the model in units of layers. Specifically, we randomly add or subtract each layer of the global model to its corresponding gradient value to generate a mutated model. Assume that the global model $w_{glb} = \{layer_1, layer_2, ..., layer_L\}$ has L layers, where $layer_i$ denotes the weights of the i^{th} layer of w_{glb} and $g_{glb} = \{lg_1, lg_2, ..., lg_L\}$ denotes the gradients of w_{glb}, where lg_i denotes the weights of the i^{th} layer of g_{glb}. For the i^{th} mutated model, we first randomly generate a mutation weight list $L_{mut}^i = [v_1^i, v_2^i, ..., v_L^i]$, where $v_1^i, v_2^i, ..., v_L^i \in \{-1, 1\}$ and then generate a mutated model as follows.

$$w_i^{mut} = \{layer_1 + \alpha v_1^i \cdot lg_1, ..., layer_L + \alpha v_L^i \cdot lg_L\}, \quad (2)$$

where α is a hyperparameter, which decides the distance of the mutated model with the global model. In other words, a higher value of α indicates a greater distance between the mutated model and the global model. Furthermore, a greater distance can guide the global model to converge into a flatter area. However, a too-large distance may make the global model difficult to converge. This is because there may not exist a flat enough area larger than the neighborhood composed of mutated models. To ensure that the mutation models can be evenly distributed in the neighborhood of the global model, we set a constraint on the mutation process as follows.

$$w_{glb} = \frac{1}{K} \sum_{i=1}^{K} w_i^{mut}, \quad (3)$$

where w_i^{mut} denotes the i^{th} mutated model. Specifically, for the mutation values of a layer j, we have

$$\sum_{i=1}^{K} v_j^i = 0. \quad (4)$$

To accommodate the situation where the number of active clients K is odd, we set $w_K^{mut} = w_{glb}$ when $K\%2 = 1$.

Algorithm 2: Implementation of Model Mutation

Input:
i) w_{glb}, the global model; ii) w'_{glb}, the old global model; iii) K, # of activated clients.
Output:
i) L_{mut}, the list of mutated models
ModelMutation(w_{glb},w'_{glb},K)

1: $g_{glb} \leftarrow w_{glb} - w'_{glb}$
2: $L_{mut} \leftarrow []$
3: **if** $K\%2 = 1$ **then**
4: $\quad L_{mut} \leftarrow L_{mut} \oplus [w_{glb}]$
5: **end if**
6: $L \leftarrow$ the number of layers of w_{glb}
7: $Lv_1, Lv_2, ..., Lv_L \leftarrow [\underbrace{-1, -1, ... -1}_{\lfloor \frac{K}{2} \rfloor}, \underbrace{1, 1, ...1}_{\lfloor \frac{K}{2} \rfloor}]$
8: **for** $i \leftarrow 1, ..., L$ **do**
9: $\quad Lv_i \leftarrow$ Randomly shuffle the elements in Lv_i
10: $\quad [v_i^1, v_i^2, ..., v_i^{2\lfloor \frac{K}{2} \rfloor}] \leftarrow Lv_i$
11: **end for**
12: **for** $i \leftarrow 1, ..., 2\lfloor \frac{K}{2} \rfloor$ **do**
13: $\quad w_{mut} \leftarrow w_{glb}$
14: $\quad$ **for** $j \leftarrow 1, ..., L$ **do**
15: $\quad\quad w_{mut}.layer_j \leftarrow w_{glb}.layer_j + \alpha v_j^i \cdot g_{glb}.lg_j$
16: $\quad$ **end for**
17: $\quad L_{mut} \leftarrow L_{mut} \oplus [w_{mut}]$
18: **end for**
19: **return** L_{mut}

Algorithm 2 presents the implementation of the model mutation. Line 1 calculates the gradients of the global model using the old global model. Line 2 initializes the list of mutated models. Lines 3-5 add the global model to the mutated model list when K is odd. Line 6 calculates the layer number of the global model. In Line 2, the cloud server initializes the L mutation value lists (i.e., Lv_1, Lv_2, ..., Lv_L) for each layer. Here, each list has $2\lfloor \frac{K}{2} \rfloor$ elements where $\lfloor \cdot \rfloor$ denotes the round-down operation, and half of the elements are assigned to -1 and the other half of the elements are assigned to 1. Lines 8-11 randomly shuffle each mutation value list to generate mutation values for the model mutation. Lines 12-18 perform the model mutation process. Line 15 generates the weights of each layer of each mutated model according to the corresponding mutation value. Line 17 adds the mutated model to the list of mutation models L_{mut}. Finally, Line 19 returns L_{mut}.

Dynamic Preference Mutation

Typically, since the model is far away from any optimal solution, the inference capacity of the global model is poor in the early stage of training. At this stage, our mutation strategy will focus on mutating the model in the direction of the gradients. In other words, we shift the neighborhood in the direction of the gradients. Specifically, we set the mutation value $v \in \{1, -1 + \beta_t\}$ rather than $\{1, -1\}$, where β_t is a hyperparameter whose value will increase with the number

of training rounds. The calculation of β_t is as follows.

$$\beta_t = max(\beta_0(1 - \frac{t}{T_b}), 0), \tag{5}$$

where T_b is the bound of the early training stage. In this way, we can guide the mutated model towards the direction of the gradients, thus accelerating the FL training.

Convergence Analysis

Similar to FedAvg, the global model of FedMut is aggregated from all the trained local models. Let t denote the t^{th} SGD iteration on the local client and each local training contains E SGD iterations, $\bar{w}$ indicates the aggregated model of all the local models. According to our mutation strategy, FedMut satisfies the following property. For $i \in 1, 2, ..., K$, we have

$$||w_{i,n}^{mut} - w_{glb}^{n}||^2 = \alpha^2 ||w_{glb}^{n} - w_{glb}^{n-1}||^2, \tag{6}$$

where $w_{i,n}^{mut}$ denotes the i^{th} mutated model in the n round. Inspired by (Li et al. 2019), we make the following four assumptions on the loss functions of local clients (i.e., f_1, f_2, ..., f_K) as follows.

Assumption 1. For $i \in \{1, 2, \cdots, K\}$, f_i is L-smooth, where $f_i(v) \leq f_i(w) + (v - w)^T \nabla f_i(w) + \frac{L}{2}||v - w||_2^2$.

Assumption 2. For $i \in \{1, 2, \cdots, K\}$, f_i is μ-strongly convex, where $f_i(v) \geq f_i(w) + (v - w)^T \nabla f_i(w) + \frac{\mu}{2}||v - w||_2^2$.

Assumption 3. The variance of stochastic gradients is bounded by σ_i^2, i.e., $\mathbb{E}||\nabla f_i(w; \xi) - \nabla f_i(w)||^2 \leq \sigma_i^2$, where ξ is a data batch of the i^{th} client in the t^{th} FL round.

Assumption 4. The expected squared norm of stochastic gradients is bounded by G^2, i.e., $\mathbb{E}||\nabla f_i(w; \xi)||^2 \leq G^2$.

Based on Assumptions 1-4 and the two properties of FedMut (i.e., Equations 3 and 6), we can obtain the convergence of FedMut as follows.

Theorem 1. *(Convergence of FedMut) Let Assumption 1-4 hold. Assume that the FL training process contains n FL rounds. Let $T = n \times E$ be the total number of SGD iterations and $\eta_t = \frac{2}{\mu(t+\gamma)}$ be the learning rate. Let $\kappa = \frac{L}{\mu}$, $\gamma = max(8\kappa, E)$. We have*

$$\mathbb{E}[f(w_T)] - f^\star \leq \frac{\kappa}{\gamma + T - 1)}[\frac{2B}{\mu} + \frac{\mu\gamma}{2}||w_1 - w^\star||^2],$$

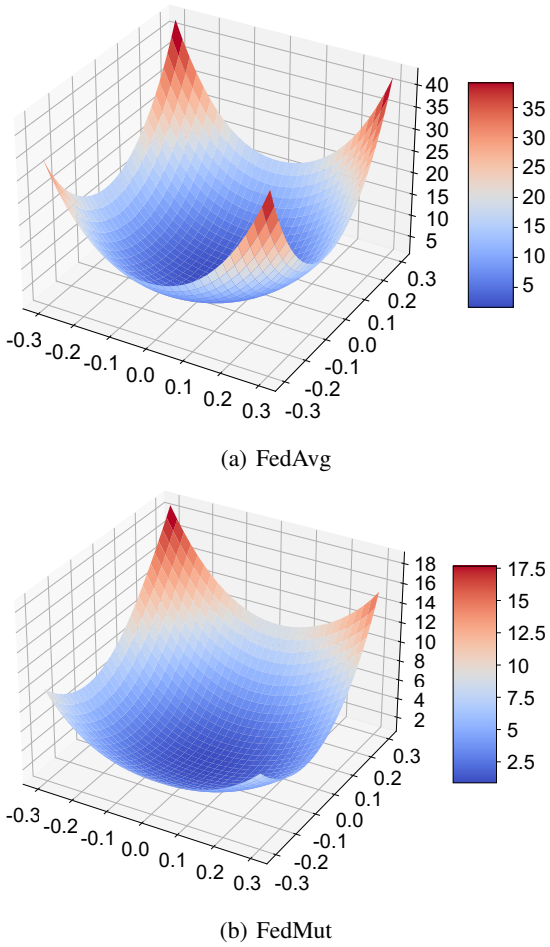
where $B = \frac{1}{K^2}\sum_{i=1}^{K}\sigma_i^2 + 6L\Gamma + (16 + 32\alpha^2)(E-1)^2G^2$.

Note that Theorem 1 indicates the difference between the loss in T^{th} interactions, i.e., $f(w_T)$ and the optimal loss, i.e., $f^\star$. From Theorem 1, we can observe that the convergence rate of FedMut is similar to that of FedAvg, which has been analyzed in (Li et al. 2019). The full proof of FedMut convergence is presented in Appendix.

Experiments

Settings

For all the experiments, we set SGD optimizer with a learning rate of 0.01 and a momentum of 0.9. For each FL training round, we set the batch size to 50 and the number epoch of each local training to 5. We conducted all the experiments on an Ubuntu workstation with an Intel i9 CPU, 64GB memory, and two NVIDIA RTX 4090 GPUs.



(a) FedAvg

(b) FedMut

Figure 4: Loss landscapes of FedAvg and FedMut on CIFAR-10 dataset using ResNet-18 with IID scenario

Datasets and Models. We selected three well-known datasets to evaluate the effectiveness of our FedMut approach, i.e., CIFAR-10, CIFAR-100 (Krizhevsky 2009), and Shakespeare (Caldas et al. 2018), respectively, where CIFAR-10 and CIFAR-100 are image datasets and Shakespeare is a text dataset. For CIFAR-10 and CIFAR-100 datasets, we conducted experiments on three models, i.e., CNN (McMahan et al. 2017), ResNet-18, and VGG-16 (TorchvisionModel 2022), respectively. For Shakespeare dataset, we used the LSTM model for experiments. To demonstrate the effectiveness of FedMut in both IID and non-IID scenarios, we adopt *Dirichlet Distribution* (Hsu, Qi, and Brown 2019) to divide CIFAR-10 and CIFAR-100 datasets. Specifically, we set the hyperparameter d for Dirichlet Distribution to 0.1, 0.5, and 1.0, respectively. Note that the smaller value of d means the greater heterogeneity of the client data. For Shakespeare dataset, we directly used

Model	Dataset	Heterogeneity Settings	Test Accuracy (%)				
			FedAvg	FedProx	FedGen	CluSamp	**FedMut**
CNN	CIFAR-10	$d = 0.1$	47.93 ± 3.26	48.21 ± 3.35	47.57 ± 2.64	47.69 ± 1.30	$\mathbf{51.25 \pm 1.07}$
		$d = 0.5$	54.33 ± 0.50	54.43 ± 0.94	53.86 ± 1.03	55.14 ± 0.98	$\mathbf{56.90 \pm 0.45}$
		$d = 1.0$	57.00 ± 0.58	57.00 ± 0.45	55.85 ± 0.49	55.91 ± 0.82	$\mathbf{58.90 \pm 0.59}$
		IID	57.32 ± 0.28	57.58 ± 0.21	57.33 ± 0.21	58.06 ± 0.19	$\mathbf{59.70 \pm 0.20}$
	CIFAR-100	$d = 0.1$	28.98 ± 0.91	29.18 ± 0.85	28.17 ± 1.00	28.81 ± 0.66	$\mathbf{31.39 \pm 0.40}$
		$d = 0.5$	32.62 ± 0.73	32.54 ± 0.78	32.21 ± 0.39	32.20 ± 0.45	$\mathbf{34.52 \pm 0.79}$
		$d = 1.0$	32.77 ± 0.53	32.98 ± 0.57	31.71 ± 0.31	32.32 ± 0.36	$\mathbf{34.80 \pm 0.39}$
		IID	32.37 ± 0.30	32.46 ± 0.26	32.20 ± 0.28	32.05 ± 0.21	$\mathbf{34.51 \pm 0.25}$
ResNet-18	CIFAR-10	$d = 0.1$	44.41 ± 3.13	43.52 ± 3.13	44.37 ± 1.88	43.17 ± 2.82	$\mathbf{55.43 \pm 1.95}$
		$d = 0.5$	61.75 ± 0.35	61.23 ± 0.62	60.69 ± 0.45	61.45 ± 0.40	$\mathbf{67.89 \pm 0.95}$
		$d = 1.0$	65.88 ± 0.20	65.89 ± 0.38	65.09 ± 0.34	65.93 ± 0.26	$\mathbf{70.01 \pm 0.18}$
		IID	64.08 ± 0.13	64.10 ± 0.10	63.84 ± 0.41	64.29 ± 0.18	$\mathbf{70.63 \pm 0.13}$
	CIFAR-100	$d = 0.1$	34.14 ± 0.76	34.21 ± 0.67	34.27 ± 0.48	33.11 ± 0.65	$\mathbf{37.91 \pm 0.35}$
		$d = 0.5$	40.72 ± 0.54	41.52 ± 0.37	41.33 ± 0.45	41.63 ± 0.38	$\mathbf{46.66 \pm 0.23}$
		$d = 1.0$	42.85 ± 0.29	43.60 ± 0.34	42.21 ± 0.16	43.06 ± 0.33	$\mathbf{47.64 \pm 0.20}$
		IID	42.58 ± 0.31	42.41 ± 0.20	42.13 ± 0.26	42.30 ± 0.34	$\mathbf{48.18 \pm 0.10}$
VGG-16	CIFAR-10	$d = 0.1$	56.14 ± 13.61	56.60 ± 6.77	64.84 ± 3.66	60.55 ± 6.63	$\mathbf{68.23 \pm 2.29}$
		$d = 0.5$	77.91 ± 0.35	77.80 ± 0.31	78.00 ± 0.17	77.68 ± 0.17	$\mathbf{80.03 \pm 0.51}$
		$d = 1.0$	79.17 ± 0.23	79.45 ± 0.19	79.03 ± 0.62	79.67 ± 0.67	$\mathbf{81.09 \pm 0.38}$
		IID	80.20 ± 0.05	79.98 ± 0.10	80.00 ± 0.06	80.29 ± 0.07	$\mathbf{81.70 \pm 0.06}$
	CIFAR-100	$d = 0.1$	46.84 ± 1.08	45.63 ± 1.67	47.36 ± 2.84	46.82 ± 0.85	$\mathbf{50.23 \pm 0.65}$
		$d = 0.5$	54.09 ± 0.83	54.80 ± 0.66	54.59 ± 0.73	54.36 ± 0.74	$\mathbf{57.28 \pm 0.53}$
		$d = 1.0$	55.62 ± 0.25	55.52 ± 0.90	55.25 ± 0.65	55.89 ± 0.55	$\mathbf{57.93 \pm 0.36}$
		IID	57.42 ± 0.09	56.74 ± 0.16	55.98 ± 0.32	56.64 ± 0.25	$\mathbf{58.09 \pm 0.21}$

Table 1: Test accuracy comparison for both non-IID and IID scenarios using three DL models

the non-IID settings in LEAF (Caldas et al. 2018).

Baselines. To demonstrate the effectiveness of FedMut, we selected the most classical FL method FedAvg (McMahan et al. 2017) and three state-of-the-art FL methods, i.e., FedProx (Li et al. 2020), FedGen (Zhu, Hong, and Zhou 2021), and ClusteredSampling (CluSamp) (Fraboni et al. 2021), respectively. FedProx is a global variable-based FL method. We set its hyperparameter μ to 0.01. CluSamp is a client grouping-based FL method. Similarly to (Fraboni et al. 2021), we select the model gradient similarity as the metric for clustering. FedGen is a knowledge distillation-based FL method. We use the same settings presented in (Zhu, Hong, and Zhou 2021).

Validation of Motivation

To validate the motivation that the global trained by FedMut is located in a flatter area than that trained by FedAvg, we obtained two ResNet-18 models trained by FedMut and FedAvg respectively on CIFAR-10 dataset with the IID scenario and visualized the loss landscapes of two models using the technology presented in (Li et al. 2018). Figure 4 shows the loss landscapes of two models trained by FedAvg and FedMut, where each model is located in the center of its loss landscape. We observe that the model trained by FedMut is located in a flatter area than that trained by FedAvg. Therefore, our FedMut approach can guide FL training towards a flatter area than FedAvg.

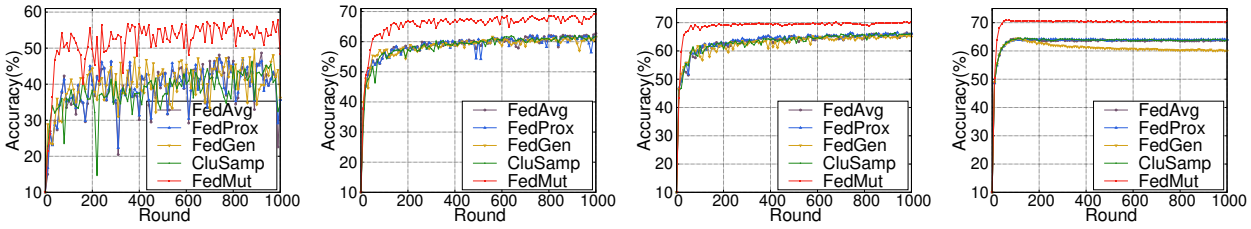
Comparison Evaluation

Comparison of Accuracy. We evaluated FedMut on Computer Vision (CV) Tasks using CIFAR-10 and CIFAR-100 datasets and Natural Language Processing (NLP) Task on Shakespeare dataset. ❶ **Computer Vision (CV) Tasks.** Table 1 presents the inference accuracy of our FedMut approach and four baselines on CIFAR-10 and CIFAR-100 datasets using three models in IID and three non-IID scenarios. Here, we set $\alpha = 4.0$ for all the cases in Table 1. We set $\beta_0 = 0.3$ for CNN and VGG-16, and $\beta_0 = 0.5$ for ResNet-18. From Table 1, we can observe that FedMut can outperform all the baselines in all the cases. We find that FedMut can achieve up to 11.02% accuracy improvements on the CIFAR-10 dataset using ResNet-18 models in $d = 0.1$ non-IID scenarios. ❷ **Natural Language Processing (NLP) Task.** Table 2 presents the accuracy of FedMut and four baselines on Shakespeare dataset using LSTM model. From Table 2, it is evident that our FedMut approach outperforms all the baselines. Consequently, our FedMut method is still suitable for FL training of NLP task models.

Dataset	Model	Method	Test Accuracy
Shakespeare	LSTM	FedAvg	52.08
		FedProx	52.53
		FedGen	52.69
		CluSamp	49.74
		FedMut	**55.53**

Table 2: Test Accuracy on Text Dataset

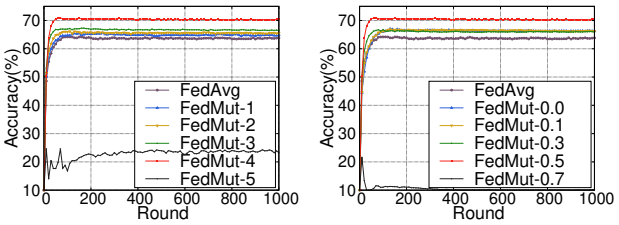
Comparison of Convergence. Figure 5 shows the learning curves of all the methods on CIFAR-10 dataset using ResNet-18. We observe that our FedMut approach can converge more quickly than all the four baselines. In addition,



(a) $d = 0.1$ (b) $d = 0.5$ (c) $d = 1.0$ (d) IID

Figure 5: Learning curves of FedMut and four baselines on CIFAR-10 using ResNet-18 model

we find that FedMut can achieve higher accuracy than all the four baselines in almost all the FL rounds. We also find that as the level of non-IID increases, the magnitude of the learning curve rises and the number of rounds needed for convergence increases.



(a) α (b) β_0

Figure 6: Ablation studies for α and β_0

Ablation Studies

Mutation Range. To demonstrate the effectiveness of our mutation strategy, we conducted experiments to evaluate the impacts of mutation range α. Figure 6(a) presents the learning curves of FedAvg and FedMut variations with different settings of α. Note that we use the notation "FedMut-a" in Figure 6(a) to denote the FedMut variation with $\alpha = a$. Note that when $\alpha = 0$, FedMut is the same as FedAvg. We can observe that when $\alpha < 5$, the inference accuracy of FedMut increases with increasing value of α. This is because a larger value of α can guide FL to converge in a flatter area. We can also observe that when $\alpha = 5$, FedMut cannot train a usable model. This happens because a too-large mutation range may cause training to be unable to find a flat enough area, which results in failed convergence.

Dynamic Preference Mutation Strategy. To demonstrate the effectiveness of our dynamic preference mutation strategy, we performed experiments to evaluate the impacts of β_0. Figure 6(b) presents the learning curves of FedAvg and FedMut variations with different settings of β_0. Note that we use the notation "FedMut-b" in Figure 6(b) to denote the FedMut variation with $\beta_0 = b$. From Figure 6(b), we observe that the higher value of β, the faster the training converges. We also find that a too large value of β may cause training to fail.

Discussion

Privacy Protection. Similar to the traditional FedAvg, the mutation strategy in FedMut is based on the aggregated global model rather than local models. Therefore, our FedMut approach can directly use the secure aggregation mechanism. Consequently, in FedMut, the cloud server cannot obtain raw data or the local model of clients.

Communication Overhead. Same as FedAvg, the communication overhead of FedMut only includes K models for dispatching and K models for uploading in each FL round. Therefore, FedMut is without any additional communication overhead compared to FedAvg.

Conclusion

To address the low-performance problems in traditional FedAvg-based methods, inspired by the observation that well-generalized solutions are located in flat areas rather than sharp areas, this paper presents a novel FL approach named *FedMut*. By mutating the global according to the gradients, FedMut generates multiple different mutated models for local training instead of the same global model. In this way, FedMut can guide FL training to converge into a flatter neighborhood composed of mutation models, thus achieving a well-generalized global model. The comprehensive experimental results show that our FedMut approach can effectively improve the performance of FL in terms of inference accuracy and convergence rate.

Acknowledgments

This research/project is supported by the National Research Foundation Singapore and DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-RP-2020-019), the Natural Science Foundation of China (62272170), the National Research Foundation, Singapore, and the Cyber Security Agency under its National Cybersecurity R&D Programme (NCRP25-P04-TAICeN), and "Digital Silk Road" Shanghai International Joint Lab of Trustworthy Intelligent Software (22510750100). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not reflect the views of National Research Foundation, Singapore, and Cyber Security Agency of Singapore. Anran Li is the corresponding author (anran.li@ntu.edu.sg).

References

Bonawitz, K.; Ivanov, V.; Kreuter, B.; Marcedone, A.; McMahan, H. B.; Patel, S.; Ramage, D.; Segal, A.; and Seth, K. 2017. Practical secure aggregation for privacy-preserving machine learning. In *Proc. of ACM SIGSAC Conference on Computer and Communications Security*, 1175–1191.

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*.

Cha, J.; Chun, S.; Lee, K.; Cho, H.-C.; Park, S.; Lee, Y.; and Park, S. 2021. Swad: Domain generalization by seeking flat minima. *Advances in Neural Information Processing Systems*, 34: 22405–22418.

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

Cui, Y.; Cao, K.; Cao, G.; Qiu, M.; and Wei, T. 2021. Client scheduling and resource management for efficient training in heterogeneous IoT-edge federated learning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(8): 2407–2420.

Foret, P.; Kleiner, A.; Mobahi, H.; and Neyshabur, B. 2020. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*.

Fraboni, Y.; Vidal, R.; Kameni, L.; and Lorenzi, M. 2021. Clustered sampling: Low-variance and improved representativity for clients selection in federated learning. In *Proc. of International Conference on Machine Learning*, 3407–3416. PMLR.

Hochreiter, S.; and Schmidhuber, J. 1997. Flat minima. *Neural computation*, 9(1): 1–42.

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.; 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 *Proc. of IEEE Real-Time Systems Symposium*, 145–157. IEEE.

Huang, Y.; Cao, Y.; Li, T.; Juefei-Xu, F.; Lin, D.; Tsang, I. W.; Liu, Y.; and Guo, Q. 2023. On the robustness of segment anything. *arXiv preprint arXiv:2305.16220*.

Huang, Y.; Chu, L.; Zhou, Z.; Wang, L.; Liu, J.; Pei, J.; and Zhang, Y. 2021. Personalized cross-silo federated learning on non-iid data. In *Proc. of the AAAI conference on artificial intelligence*, volume 35, 7865–7873.

Jia, C.; Hu, M.; Chen, Z.; Yang, Y.; Xie, X.; Liu, Y.; and Chen, M. 2023. AdaptiveFL: Adaptive Heterogeneous Federated Learning for Resource-Constrained AIoT Systems. *arXiv preprint arXiv:2311.13166*.

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

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images. *Master's thesis, University of Tront*.

Li, A.; Zhang, L.; Tan, J.; Qin, Y.; Wang, J.; and Li, X.-Y. 2021a. Sample-level data selection for federated learning. In *Proc. of IEEE Conference on Computer Communications*, 1–10. IEEE.

Li, A.; Zhang, L.; Wang, J.; Han, F.; and Li, X.-Y. 2021b. Privacy-preserving efficient federated-learning model debugging. *IEEE Transactions on Parallel and Distributed Systems*, 33(10): 2291–2303.

Li, A.; Zhang, L.; Wang, J.; Tan, J.; Han, F.; Qin, Y.; Freris, N. M.; and Li, X.-Y. 2021c. Efficient federated-learning model debugging. In *Proc. of IEEE International Conference on Data Engineering (ICDE)*, 372–383. IEEE.

Li, H.; Xu, Z.; Taylor, G.; Studer, C.; and Goldstein, T. 2018. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31.

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

Li, T.; Li, Z.; Li, A.; Du, M.; Liu, A.; Guo, Q.; Meng, G.; and Liu, Y. 2023b. Fairness via group contribution matching. In *Proc. of International Joint Conference on Artificial Intelligence*, 436–445.

Li, T.; Liu, A.; Liu, X.; Xu, Y.; Zhang, C.; and Xie, X. 2021d. Understanding adversarial robustness via critical attacking route. *Information Sciences*, 547: 568–578.

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

Li, X.; Huang, K.; Yang, W.; Wang, S.; and Zhang, Z. 2019. On the Convergence of FedAvg on Non-IID Data. In *Proc. of International Conference on Learning Representations*.

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

Long, G.; Tan, Y.; Jiang, J.; and Zhang, C. 2020. Federated learning for open banking. In *Federated Learning: Privacy and Incentive*, 240–254. Springer.

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

Rieke, N.; Hancox, J.; Li, W.; Milletari, F.; Roth, H. R.; Albarqouni, S.; Bakas, S.; Galtier, M. N.; Landman, B. A.; Maier-Hein, K.; et al. 2020. The future of digital health with federated learning. *NPJ digital medicine*, 3(1): 119.

Sattler, F.; Korjakow, T.; Rischke, R.; and Samek, W. 2021. Fedaux: Leveraging unlabeled auxiliary data in federated

learning. *IEEE Transactions on Neural Networks and Learning Systems*.

Stutz, D.; Hein, M.; and Schiele, B. 2021. Relating adversarially robust generalization to flat minima. In *Proc. of the IEEE/CVF International Conference on Computer Vision*, 7807–7817.

TorchvisionModel. 2022. Models and pre-trained weight. https://pytorch.org/vision/stable/models.html.

Wang, G.; Guo, H.; Li, A.; Liu, X.; and Yan, Q. 2023. Federated iot interaction vulnerability analysis. In *Proc. of IEEE International Conference on Data Engineering (ICDE)*, 1517–1530. IEEE.

Yue, Z.; Xia, J.; Ling, Z.; Hu, M.; Wang, T.; Wei, X.; and Chen, M. 2023. Model-Contrastive Learning for Backdoor Elimination. In *Proc. of the ACM International Conference on Multimedia*, 8869–8880.

Zhang, C.; Liu, A.; Liu, X.; Xu, Y.; Yu, H.; Ma, Y.; and Li, T. 2020a. Interpreting and improving adversarial robustness of deep neural networks with neuron sensitivity. *IEEE Transactions on Image Processing*, 30: 1291–1304.

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

Zhang, X.; Zhang, C.; Li, T.; Huang, Y.; Jia, X.; Xie, X.; Liu, Y.; and Shen, C. 2023. A Mutation-Based Method for Multi-Modal Jailbreaking Attack Detection. arXiv:2312.10766.

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. PMLR.