A Fine-grained Differentially Private Federated Learning against Leakage from Gradients

Linghui Zhu, Xinyi Liu, Yiming Li, Xue Yang, Shu-Tao Xia, and Rongxing Lu, *Fellow, IEEE*

***Abstract*—Federated learning enables data owners to train a global model with shared gradients while keeping private training data locally. However, recent research demonstrated that the adversary may infer private training data of clients from the exchanged local gradients, e.g., having deep leakage from gradients (DLG). Many existing privacy-preserving approaches take usage of differential privacy to guarantee privacy. Nevertheless, the widely used privacy budget of differential privacy (e.g., evenly distribution) leads to a sharp decline of model accuracy. To improve the model accuracy, some schemes only consider allocating the privacy budget to the fully connected layers. However, we reveal that the adversary may still reconstruct the private training data by adopting the DLG attack with the gradients of convolutional layers. In this paper, we propose a fine-grained differential privacy federated learning (DPFL) scheme, which guarantees privacy and remains high model performance simultaneously. Specifically, inspired by the methods that measure the importance of layers in deep learning, we propose a fine-grained method to allocate noise according to the importance value of layers in order to remain high model performance. Besides, we combine an active client selection strategy with DPFL and perform fine-tuning with a public dataset on the server to further ensure the model performance. We evaluate DPFL under both i.i.d and non-i.i.d data settings to show that our method can achieve similar accuracy as the plain federated learning (e.g., FedAvg). We also demonstrate that our DPFL can resist DLG attack to verify its privacy guarantee.**

***Index Terms*—Federated learning, Deep Leakage from Gradient, Differential Privacy, AI Security.**

I. Introduction

DEEP learning, especially deep neural networks (DNNs), has been widely and successfully adopted in many IoT devices for its high effectiveness and efficiency. In the meanwhile, the widespread use of IoT devices also makes it possible to collect high-quality data from users. Since the success of DNNs relies heavily on a large amount of training data, the server (e.g., service provider) may keep updating the algorithm based on data collected by IoT devices via federated learning (FL) [1]. The purpose of using FL is to preserve users' privacy since FL allows training DNNs in a distributed manner without the access of local data from the clients (i.e., IoT devices). Specifically, each client computes and uploads local gradients to the server, which will further aggregate all collected gradients and update the model.

Although FL improves the level of privacy-preservation, sharing local gradients still faces the risk of sensitive training data being leaked [2]–[4]. For example, [3], [4] demonstrated that adversaries can reconstruct training data from shared gradients. In order to further improve privacy, many privacy-preserving techniques [5]–[8] have been widely adopted to encrypt or mask the shared gradients. Among them, differential privacy (DP) [9]–[12] is more widely used, especially in IoT applications, for its high efficiency and simplicity. Specifically, many recent works perturbed shared gradients by adding Laplacian or Gaussian noise satisfying the requirement of DP. However, due to inappropriate allocation of the privacy budget, these works introduce too much noise, which significantly reduces the model accuracy. To alleviate the decline of accuracy, Liu et al. [6] presented the layer-wise importance propagation (LIP) algorithm, which allocates privacy budget according to the importance of parameters in fully connected layers. Since LIP [6] does not add noise to convolutional layers, the total amount of noise is naturally lower compared with the standard DP-based method. However, for many DNNs with convolutional layers, this algorithm only adds noise on the fully connected layers and therefore still having a risk of data leakage. To verify it, we examine the LIP algorithm with the attack proposed in [3]. As shown in Figure 1, the adversary can still easily reconstruct training images from gradients of convolutional layers. In general, LIP [6] achieves high accuracy by sacrificing privacy to some extent.

In fact, improving the model accuracy while ensuring a high level of privacy-preserving is still a challenge in many DP-based federated learning applications. In this paper, we propose an efficient and fine-grained differential privacy federated learning scheme (DPFL) trying to overcome these challenges. In our DPFL, all shared gradients are perturbed by adding Laplacian noise to ensure that all layers are under protection. Instead of evenly allocated, we specify the privacy budget of DP in each layer according to its importance score, which can greatly improve model accuracy. Besides, we take advantage of an active client selection strategy [13] and perform fine-tuning with a public dataset to further improve the model accuracy while preserving privacy and efficiency. These strategies are

Linghui Zhu, Xinyi Liu,and Yiming Li are with Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen, China, 518023 (e-mail: zlh20@mails.tsinghua.edu.cn, liuxinyi19@mails.tsinghua.edu.cn, li-ym18@mails.tsinghua.edu.cn).

Xue Yang is with the Information Security and National Computing Grid Laboratory, Southwest Jiaotong University, Chengdu, China, 610031 (e-mail: xueyang.swjtu@gmail.com).

Shu-Tao Xia is with Tsinghua Shenzhen International Graduate School, Tsinghua University, and also with the Peng Cheng Laboratory, Shenzhen, China, 518055 (e-mail: xiast@sz.tsinghua.edu.cn).

Rongxing Lu is with the Canadian Institute of Cybersecurity, Faculty of Computer Science, University of New Brunswick, Fredericton, Canada, E3B 5A3 (e-mail: rlu1@unb.ca).

Corresponding Authors: Xue Yang (e-mail: xueyang.swjtu@gmail.com) and Shu-Tao Xia (e-mail: xiast@sz.tsinghua.edu.cn).

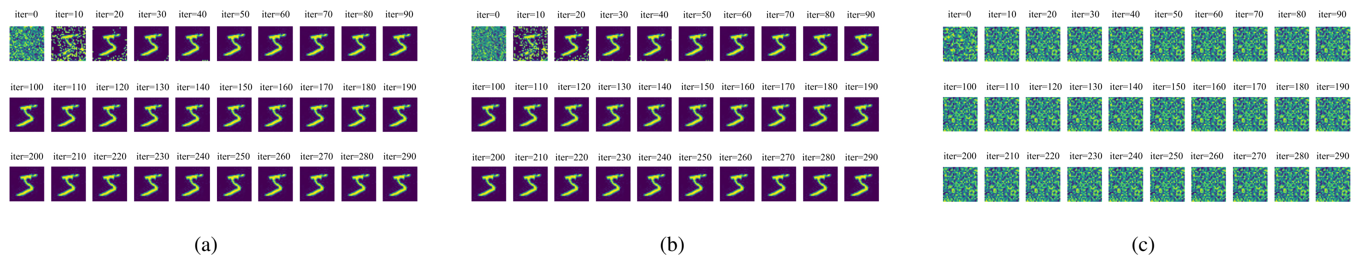

iter=0 iter=10 iter=20 iter=30 iter=40 iter=50 iter=60 iter=70 iter=80 iter=90
iter=100 iter=110 iter=120 iter=130 iter=140 iter=150 iter=160 iter=170 iter=180 iter=190
iter=200 iter=210 iter=220 iter=230 iter=240 iter=250 iter=260 iter=270 iter=280 iter=290
(a)
(b)
(c)

Fig. 1. Recover training data via the attack of deep leakage from gradients (DLG) [3]. **(a)** images recovered from gradients of all layers without noise. **(b)** images recovered from gradients of all convolutional layers trained with LIP [6]. **(c)** images recovered from gradients of all fully connected layers trained with LIP [6]. These results indicate that the adversary can still recover training images even under the protection of LIP, highlighting the necessity of protecting gradients of all layers for preventing data leakage.

effective, especially under the non-i.i.d data setting.

The main contributions of this work are four-folds:

- We reveal that adversaries can reconstruct training data from the gradients of both fully connected and convolutional layers. It highlights the necessity of protecting gradients of all layers for privacy considerations.
- Inspired by the methods measuring the importance of layers in deep learning, we design a fine-grained layer-wise allocation algorithm of privacy budget for differential privacy where more budgets will be allocated to less important layers to maintain high accuracy.
- We take advantage of an active client selection strategy to improve the training efficiency while performing fine-tuning with a public dataset on the server-side to further improve model accuracy. Specifically, the selection strategy can improve performance and accelerate model convergence and therefore decreasing training costs.
- We conduct experiments on multiple benchmark datasets under both non-i.i.d and i.i.d data settings, which verify that our DPFL can have performance on par with that of the plain federated learning (e.g., FedAvg) while resistant to DLG attacks and preserving high efficiency.

The rest of this paper is organized as follows: In Section II, we review some related works. In Section III, we briefly introduce the preliminaries and outline the design goals of our proposed scheme. We introduce the technical details of our DPFL in Section IV. Section V carries out the analysis of our model in performance, privacy preservation, and efficiency. Finally, the conclusion of this paper is given in Section VI.

II. Related Works

In this section, we review some related studies about the privacy challenges and defense methods in FL.

A. *Privacy Challenges in Federated Learning*

Federated learning (FL) [1] allows training DNNs without the direct access of local data. As such, it is widely adopted in the update of IoT devices whose training samples collected from users needs protection for privacy considerations. In FL, the global model is trained without gathering data from data owners which ensures data security to some extent. However, various attack methods still bring huge security threat to FL. For example, deep leakage from gradients (DLG) [3], [4] is designed to steal sensitive information of private local training datasets from the shared gradients. The leakage of DLG [3] is performed by optimizing a randomly generated image. The cost function of DLG [3] is the euclidean distance between the gradients of the generated image and the ground truth image. Optimizing the distance of gradients also makes the generated image close to the ground truth training data. Although a large batch size makes the leakage hard to succeed, Geiping et al. [4] succeeded to recover training data with a batch size of 8 under FL settings. They proposed to use cosine similarity as the cost function to minimize the angle between gradients of generated images and those of ground truth images.

B. *Defense Methods in Federated Learning*

Some researches have discussed the defense methods in FL [4], [5], [14]. Considering the bottlenecks of FL, privacy-preserving methods should not only ensure data security, but also take model performance and efficiency into consideration.

Differential privacy (DP) [9]–[12] is a widely used mechanism to ensure data security for its high efficiency and simplicity. The goal of DP is to hide a single record in the dataset, i.e., to make the outputs of two similar datasets indistinguishable. Some studies have taken the advantage of DP to guarantee privacy in centralized model training [3], [7], [8] which is inconsistent with the decentralized framework of FL. DP provides privacy protection by adding noise which leads to a significant drop of model accuracy due to the inappropriate allocation of privacy budget. In the study of Zhu et al. [3], DP is one of the possible defense strategies against DLG [3]. However, the privacy budget is allocated evenly in [3] which reduces the accuracy significantly. To better preserve model performance, Liu et al. [6] introduced the layer-wise importance propagation (LIP). It calculated the importance value of every parameter in the fully connected layers, based on which to allocate the privacy budget individually rather than evenly. The more important the parameter is, the smaller the allocated privacy budget. In particular, LIP algorithm [6] only adds noise on the fully-connected layers which satisfies the requirement of DP according to the composition theorems [15]. However, as shown in Figure 1, LIP still faces the risk of data leakage attacked by DLG [3], based on the gradients of convolutional layers. Accordingly, the high accuracy of LIP [6] is achieved at the cost of the privacy guarantee to some extent. How to achieve high model performance while preserving data privacy is still an important open problem in FL.

III. Preliminaries and Problem Statement

In this section, we first give a brief introduction of differential privacy and the framework of federated learning. Then we identify the threat model and the corresponding design goals.

A. Differential Privacy

Differential privacy (DP) was proposed by C. Dwork et al. [9]–[12], which provides a rigorous privacy guarantee based on probability. Its detailed definition is given as follows:

Definition 1 (ϵ-Differential Privacy [10]). *A randomized algorithm $\mathcal{M}$ gives ϵ-differential privacy for the output space of $Range(\mathcal{M})$ and any neighboring datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ differing by at most one record, if $\mathcal{M}$ satisfies: $\forall\ Y \in Range(\mathcal{M})$*

$$\Pr[\mathcal{M}(\mathcal{D}_1) \in Y] \leq e^{\epsilon} \times \Pr[\mathcal{M}(\mathcal{D}_2) \in Y],$$

where ϵ denotes the privacy budget of differential privacy that restricts the privacy guarantee level of $\mathcal{M}$, and a smaller value of ϵ represents a stronger privacy level.

Currently, two classical mechanisms are widely used to achieve ϵ-differential privacy by adding noise, including the Laplace mechanism [10] and the Exponential mechanism [16]. The former one is more suitable for numeric queries and the later one is more suitable for non-numeric queries [17]. Since we intend to add noise to (numeric) shared gradients, we adopt the Laplace mechanism in our method. Specifically, the Laplace mechanism can provide ϵ-differential privacy by adding noise drawn from Laplace distribution, as follows:

Definition 2 (Laplace Mechanism [10]). *For any function f, the Laplace mechanism $\mathcal{M}(\mathcal{D}) = f(\mathcal{D}) + Lap(\frac{\Delta f}{\epsilon})$ satisfies ϵ-differential privacy, where $Lap(\frac{\Delta f}{\epsilon})$ is a random variable following the Laplace distribution, i.e, $\Pr[Lap(\beta) = x] = \frac{1}{2\beta}e^{\frac{-|x|}{\beta}}$, where the corresponding mean and variance are set to 0 and $\beta = \frac{\Delta f}{\epsilon}$, respectively.*

The parameter Δf is the sensitivity of the function f, which is given in Definition 3.

Definition 3 (Sensitivity [10]). *For any pair of neighboring datasets $\mathcal{D}_1$ and $\mathcal{D}_2$, the sensitivity Δf of a function f, denoted by Δf, is shown below*

$$\Delta f = \max_{\mathcal{D}_1, \mathcal{D}_2} ||f(\mathcal{D}_1) - f(\mathcal{D}_2)||_1.$$

The proposed scheme can be regarded as a multi-step mechanism, where the sequential composition theorem and the parallel composition theorem of differential privacy are widely used [15]. We introduce these two composition theorems here:

Property 1 (Sequential Composition Property [15]). *Given a sequence of randomized algorithm $\{\mathcal{M}_1, ..., \mathcal{M}_k\}$ that $\forall \mathcal{M}_i \in \{\mathcal{M}_1, \dots, \mathcal{M}_k\}$: $\mathcal{M}_i$ satisfies ϵ_i-differential privacy. Then $\mathcal{M}(\mathcal{D}) = \langle \mathcal{M}_0(\mathcal{D}), \mathcal{M}_1(\mathcal{D}), ..., \mathcal{M}_k(\mathcal{D})\rangle$ satisfies $(\sum_{i=0}^{k} \epsilon_i)-$differential privacy.*

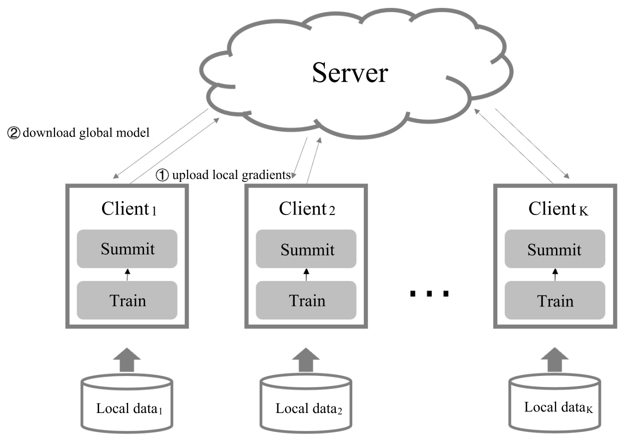


Fig. 2. The framework of federated learning.

Property 2 (Parallel Composition Property [15]). *Given a sequence of randomized algorithm $\{\mathcal{M}_1, ..., \mathcal{M}_k\}$ that $\forall \mathcal{M}_i \in \{\mathcal{M}_1, \dots, \mathcal{M}_k\}$: $\mathcal{M}_i$ satisfies ϵ_i-differential privacy. $\langle \mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_k\rangle$ are the disjoint partitioning of $\mathcal{D}$. Then $\mathcal{M}(\mathcal{D}) = \langle \mathcal{M}_1(\mathcal{D}_1), \mathcal{M}_2(\mathcal{D}_2), ..., \mathcal{M}_k(\mathcal{D}_k)\rangle$ satisfies $(\max_{i\in[1,k]} \epsilon_i)-$differential privacy.*

B. System Model

The framework of federated learning consists of a center server and a number of clients, e.g., K clients denoted as $C = \{C_1, C_2, \dots, C_K\}$, as illustrated in Figure 2. With the help of the server, clients can train a global model cooperatively without exchanging data. Specifically, the roles of different entities are explained as follows:

- **Clients** own training data containing sensitive information. Since different clients are independent and even may have no relation, the datasets of different clients cannot be independent and identically distributed, i.e., the datasets among different clients are non-i.i.d. In FL, each client trains a local model with their local data in parallel and uploads local gradients to the server.
- **Server** is responsible for initializing the global model and aggregating local gradients of different clients for model updating. Similar to [18], [19], the server also owns a public dataset that does not contain any sensitive information to fine-tune the global model for further improving the learning accuracy.

C. Threat Model

Following the settings of existing works [4], [6], [20], the server is assumed to be 'honest-but-curious', i.e., they execute the proposed scheme honestly while are curious about the sensitive training data. Specifically, the server intends to adopt the DLG attack [3], [4] to reconstruct the training

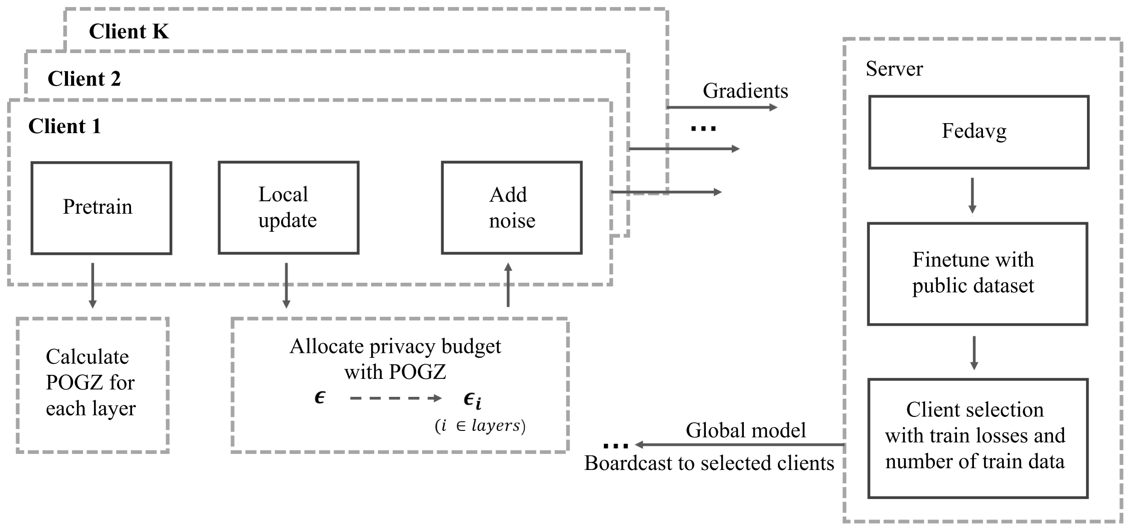

Client K
Client 2
Client 1
Pretrain
Local update
Add noise
Gradients
Server
Fedavg
Finetune with public dataset
Client selection with train losses and number of train data
Calculate POGZ for each layer
Allocate privacy budget with POGZ
ϵ - - - - → ϵ_i ($i \in layers$)
Global model
Boardcast to selected clients

Fig. 3. The framework of DPFL. The server chooses clients with active client selection strategy. The selected clients add noise on local gradients and then share the noisy gradients to server. The server updates the global model and fine-tunes the global model with a public dataset.

data of clients based on sharing gradients. Other attacks are beyond the scope of this paper and we will consider more attack methods in the future work. The benign clients attempt to provide privacy guarantees via DP while preserving high model performance simultaneously, as follows:

- **Privacy-preservation.** The proposed scheme should ensure that the adversary cannot obtain the sensitive training data of victims by deep leakage from gradients [3], [4].
- **Effectiveness.** Although the DP guarantees privacy by sacrificing the model accuracy, the proposed scheme should preserve sufficient accuracy as much as possible.
- **Efficiency.** In practice, efficiency in terms of computational overhead is an important factor in determining whether a scheme is adopted. Thus, the proposed scheme should ensure affordable overheads for both server and clients, especially for IoT devices.

IV. Proposed Scheme

In this section, we introduce our proposed fine-grained differential privacy federated learning (DPFL) scheme. We first give the overall procedure of the proposed scheme (as shown in Figure 3), and then introduce each step in detail. We prove that DPFL satisfies differential privacy in the end. Before we provide technical details of DPFL, we first give the description of notations used in the proposed scheme in Table I.

TABLE I
Notations used in the proposed scheme.

Symbol	Description
$C = \{C_1, \ldots, C_K\}$	The set of clients
$S^{(t)}$	Subset of clients in the t-th iteration
λ	The fraction of selected clients
n_k	The number of C_k's training data
$v_k^{(t)}$	The selected evaluation value of C_k in the t-th iteration
$w^{(t)}$	Global model in the t-th iteration
$\nabla g_k^{(t)}$	Gradients of C_k in the t-th iteration
$\widehat{\nabla g_k^{(t)}}$	Gradients with noise
α	The bound of gradients
ϵ	Privacy budget
Δf	Sensitivity of DP
$O^{(i)}$	Output before activation function of the i-th layer
η	Learning rate
$\alpha_1, \alpha_2, \alpha_3$	Tuning parameters of client selection

A. Overall Procedure

Our DP-based scheme mainly includes four steps:

- *Step 1: Identifying layer importance.* Each client pretrains the local model with their local data and calculates the importance value of each layer.
- *Step 2: Client selection.* During the t-th global update epoch of model training, the server first selects a subset of clients $S^{(t)} \subseteq C$ (the size of $S^{(t)}$ is $|S^{(t)}| = K \times \lambda$, where λ is the ratio of selected clients) to attend current iteration according to the train losses and the number of training data of each client. Then, the server broadcasts the current global model to all selected clients.
- *Step 3: Noisy gradients computation.* Based on the stochastic gradient descent (SGD) technique, each selected client $C_k \in S^{(t)}$ first computes local gradients $\nabla g_k^{(t)}$ with local training data and the received current global model. Then, C_k adds Laplacian noise to computed local gradients according to the importance of each

layer and uploads the noisy local gradients to the server.

- *Step 4: Model update.* Similar to FedAvg [1], the server aggregates and updates the current global model as:
$$w^{(t+1)} = w^{(t)} - \eta \sum_{k}^{C_k \in S^{(t)}} (n_k/n) \times \nabla g_k^{(t)},$$
where $n = \sum_{k=1}^{K} n_k$ and n_k is the number of training data of C_k, η is the learning rate. After that, the server fine-tunes the updated model $w^{(t+1)}$ with a public dataset to ensure a higher learning accuracy [18], [19] and then distributes it to the selected clients for the next iteration.

The server and clients interactively perform the steps (1)-(4) until the convergence, and thereby obtaining a well-trained global model for prediction. In the following, we will introduce each step in detail.

B. Identifying Layer Importance

As explored in [6], allocating privacy budget according to the importance of parameters can alleviate the decline of accuracy. However, the proposed method in [6] does not take convolutional layers into consideration which brings security risks. Besides, allocating privacy budget according to the importance of layers may probably further improve the accuracy of the learned model. Inspired by the Average Percentage of Zeros (APoZ) [21] algorithm which is widely used to decide the importance of layers due to its efficiency and high ratio of pruning, we propose a fine-grained layer-wise algorithm Percentage of numbers Greater than Zero (PoGZ) to measure the importance of each layer, thereby allocating privacy budget ϵ. Specifically, the layer importance computation mainly includes two phases: *pre-training* and *layer importance calculation*, as follows:

- *Pre-training*: Before starting the training process of federated learning, each client first pre-trains local models with their own local training data.
- *Obtaining layer importance*: Each client runs the pre-trained local model on local validation set and calculates the importance of each layer. We use the PoGZ in the output vector before activation function of a layer to stand for the importance of this layer, defined as follows:
$$\texttt{PoGZ}^{(i)} = \frac{\sum_{p=1}^{N} \sum_{j=1}^{M} f(O_j^{(i)}(p) > 0)}{N \times M}, \tag{1}$$
where $O_j^{(i)}(p)$ is the j-th element in the output vector before activation function in the i-th layer when the p-th validation sample is the model input. M is the size of $O^{(i)}(p)$ and the total number of validation sample is N. $f(\cdot)$ is an indicator function satisfying that $f(\cdot) = 1$ if the evaluated event is true and $f(\cdot) = 0$ otherwise.

C. Client Selection

To alleviate the decline in accuracy, we utilize an active client selection strategy [13] that the server selects an optimal subset of clients based on the local training loss and the quantity of training data, instead of random clients selection.

Algorithm 1 Clients selection.

```
Require: The client set $C = \{C_1, C_2, \ldots, C_K\}$ and the fraction $\lambda$ of selected clients in the $t$-th iteration, the corresponding selected evaluation values $v^{(t)} = \{v_1^{(t)}, ..., v_K^{(t)}\}$, tuning parameters $\alpha_1, \alpha_2, \alpha_3$, the number of clients $K$.
Ensure: Subset of selected clients: $S^{(t)}$
 1: function ClientSelection($C, \lambda, v^{(t)}, \alpha_1, \alpha_2, \alpha_3, K$)
 2:     Sort clients by $v_k^{(t)}$
 3:     $m = \lambda \cdot K$
 4:     For the $\alpha_1 \cdot K$ clients with smallest $v_k^{(t)}$, directly set $v_k^{(t)} = -\infty$
 5:     for $k \in [1, K]$ do
 6:         $p_k^{(t)} = e^{\alpha_2 v_k^{(t)}}$
 7:     end for
 8:     Sample $(1 - \alpha_3)m$ clients according to their $p_k^{(t)}$ and thereby producing a set $S'$
 9:     Sample $\alpha_3 m$ clients from the remaining clients uniformly at random, and thereby producing a set $S''$
10:     $S^{(t)} = S' \cup S''$
        return $S^{(t)}$
11: end function
```

The client selection strategy actively adapts to the state of the model and the data on each client which enhances the performance of FL [13]. The details of client selection are shown in Algorithm 1, which mainly includes four steps:

- *Selected evaluation calculation:* The selected evaluation value $v_k^{(t)}$ of each selected client in the $(t-1)$-th global update is calculated based on training loss and the number of training data as:
$$v_k^{(t)} = \begin{cases} \frac{n_k}{\sum_k n_k} \times Loss_k^{(t-1)}, & C_k \in S^{(t-1)} \\ v_k^{(t-1)}, & otherwise \end{cases},$$
where n_k means the number of C_k's training data and $S^{(t-1)}$ is the subset of clients chosen by the server in the $(t-1)$-th update epoch. Note that none of the clients have been selected in the first global epoch. As such, we directly select m clients randomly where $m = \lambda \cdot K$. Specifically, only if the C_k has been selected in the $(t-1)$-th iteration, the $v_k^{(t)}$ is updated.
- *Sort and dropout:* Sort the clients by $v_k^{(t)}$. In order to make the clients that are not selected in the $(t-1)$-th round have the chance to be selected in the next round, we drop out $\alpha_1 K$ clients with the smallest $v_k^{(t)}$.
- *Probability value calculation:* A probability value $p_k^{(t)}$ is calculated with a hyperparameter α_2 and $v_k^{(t)}$ as:
$$p_k^{(t)} = e^{\alpha_2 v_k^{(t)}}.$$
- *Sampling:* Sample $(1 - \alpha_3)m$ clients according to their $p_k^{(t)}$ and sample $\alpha_3 m$ unselected clients randomly.

D. Noisy Gradients Computation

In order to provide privacy guarantee with DP and ensure the accuracy, clients allocate the privacy budget according to PoGZ and add noise on the gradients before sharing them with the server. Concretely, each selected client $C_k \in S^{(t)}$ first trains the local model with the received global model and their own local training data. Then, C_k allocates the privacy budget of DP according to the PoGZ computed in Eq. (1), and injects the Laplacian noise into the computed gradients accordingly. Finally, C_k submits the noisy local gradients to the server for model update. Algorithm 2 gives a description of the noisy gradients computation phase, which mainly includes the following four steps:

- *Local training*: The selected client $C_k \in S^{(t)}$ trains the local model with local training data to obtain the local gradients $\nabla g_k^{(t)}$ and bounds the gradients [7], [8] as:

$$\nabla g_k^{(t)} = \frac{\nabla g_k^{(t)}}{\max(1, \frac{||\nabla g_k^{(t)}||_1}{\alpha})}. \quad (2)$$

  Note that the role of the operation in Eq. (2) is to limit the value of gradients to satisfy $-\alpha \leq \nabla g_k^{(t)} \leq \alpha$, which can help determining the sensitivity of DP.
- *Privacy budget allocation*: The i-th layer gets DP budget ϵ_i according to the $\texttt{PoGZ}^{(i)}$ as follow:

$$\epsilon_i = \frac{\texttt{PoGZ}^{(i)}}{\sum \texttt{PoGZ}^{(i)}} \times \epsilon. \quad (3)$$

- *Noise injection*: For the i-th layer, Laplace noise is added as:

$$\widehat{\nabla g_k^{(t)}(i)} = \nabla g_k^{(t)}(i) + Lap(\frac{\Delta f}{\epsilon_i}), \quad (4)$$

  where the sensitivity is $\Delta f = 2 \times \eta \times \alpha$.
- *Transmission*: After noise injection phase, clients share the noisy gradients to the server-site. Besides, the mean loss of clients is shared as well in order to calculate the selected evaluation value $v_k^{(t)}$.

E. Model Update

In the last step of DPFL, the server needs to update the global model with noisy local gradients and fine-tune it with a public dataset in order to ensure the accuracy. The process of model update is shown in Algorithm 3, which mainly contains the following two steps:

- *Aggregation*: After receiving noisy local gradients from clients, the server implements FedAvg [1] to update the global model as:

$$w^{(t+1)} = w^{(t)} - \eta \sum_{k}^{C_k \in S^{(t)}} (n_k/n) \times \widehat{\nabla g_k^{(t)}},$$

  where $n = \sum_{k=1}^{K} n_k$.
- *Fine-tuning*: In order to further improve the learned accuracy, the server needs to fine-tune the updated global model with a public dataset.

Algorithm 2 Noisy Gradients Computation

```
Require: Global model of the $t$-th global iteration $w^{(t)}$,
    privacy budget $\epsilon$, the bound of gradient $\alpha$.
        // Run on $C_k$
Ensure: $\widehat{C_k}$' local gradients of the $t$-th global iteration with
    noise: $\nabla g_k^{(t)}$, mean loss of $C_k$: $Loss_k^{(t)}$
 1: function ClientUpdate($w^{(t)}$, $\epsilon$, $\alpha$, $k$)
 2:     $w \leftarrow w^{(t)}$
 3:     $Loss_k^{(t)} = 0$
 4:     for each local epoch do
 5:         for Batch $b \in LocalDataset$ do
 6:             $\nabla g = \nabla_w Loss(w, b)$
 7:             $Loss_k^{(t)} = Loss_k^{(t)} + Loss(w, b)$
 8:             $w = w - \eta \nabla g$
 9:         end for
10:     end for
11:     $\nabla g_k^{(t)} = w^{(t)} - w$
12:     $\nabla g_k^{(t)} = \nabla g_k^{(t)} / \max(1, ||\nabla g_k^{(t)}||_1/\alpha)$
13:     $\Delta f = 2 \times \eta \times \alpha$
14:     for $i$-th layer in $w$ do
15:         $\epsilon_i = (\texttt{PoGZ}^{(i)} / \sum \texttt{PoGZ}^{(i)}) \times \epsilon$
16:         $\widehat{\nabla g_k^{(t)}(i)} = \nabla g_k^{(t)}(i) + Lap(\Delta f/\epsilon_i)$
17:     end for
18:     $Loss_k^{(t)} = Loss_k^{(t)} / size(LocalDataset)$
        return $\widehat{\nabla g_k^{(t)}}, Loss_k^{(t)}$
19: end function
```

Algorithm 3 Global Update

```
 1: Clients pre-train local models and calculate PoGZ for each
    layer.
 2: Server Site :
 3: Initialize $w^{(0)}$
 4: for each round $t = 0, 1, 2, ...$ do
 5:     $S^{(t)}$ = ClientSelection($C, \lambda, v^{(t)}, \alpha_1, \alpha_2, \alpha_3, K$)
 6:     for $C_k \in S^{(t)}$ in parallel do
 7:         $\widehat{\nabla g_k^{(t)}}, Loss_k^{(t)}$ =ClientUpdate($w^{(t)}, \epsilon, \alpha, k$)
 8:         $v_k^{(t+1)} = (n_k/n) \times Loss_k^{(t)}$
 9:     end for
10:     $w^{(t+1)} = w^{(t)} - \eta \sum_k^{C_k \in S^{(t)}} (n_k/n) \times \widehat{\nabla g_k^{(t)}}$
11:     Fine-tune $w^{(t+1)}$ with public dataset.
12: end for
```

After fine-tuning, the server distributes the updated global model to the selected clients for the next iteration. And the selected clients start noisy gradients computation again.

F. Privacy Analysis

In this section we show the proof that the noisy gradients which is noted as $\widehat{\nabla g_k^{(t)}}$ satisfies $\epsilon-$differential privacy.

Theorem 1. *The proposed DPFL satisfies ϵ-differential privacy when $\epsilon = \max_{k\in[1,K]} \epsilon_k$, where ϵ_k is the privacy budget of C_k and K is the number of clients.*

Proof. Assuming there are two neighboring datasets $\mathcal{D}_1$ and $\mathcal{D}_2$. In order to calculate the sensitivity of $\nabla g_k^{(t)}$, the gradients are clipped during training as shown in Eq. (2) and α is the bound of gradients (i.e., $\nabla g_k^{(t)} \in [-\alpha, \alpha]$). Thus, the sensitivity Δf in the i-th layer can be calculated by:

$$\begin{aligned}\Delta f &= \max_{\mathcal{D}_1,\mathcal{D}_2} ||\nabla g_k^{(t)}(i,\mathcal{D}_1) - \nabla g_k^{(t)}(i,\mathcal{D}_2)||_1\\ &= 2\times\eta\times\max||\nabla g_k^{(t)}||_1\\ &= 2\times\eta\times\alpha\end{aligned}$$

where η is the learning rate.

The privacy budget is allocated as Eq. (3) and C_k adds Laplace noise to the gradients of i-th layer before sending it to the server as Eq. (4). Thus we have:

$$\begin{aligned}\frac{\Pr(\widehat{\nabla g_k^{(t)}(i,\mathcal{D}_1)} = Y)}{\Pr(\widehat{\nabla g_k^{(t)}(i,\mathcal{D}_2)} = Y)} &= \frac{\Pr(\nabla g_k^{(t)}(i,\mathcal{D}_1) + Lap(\frac{\Delta f}{\epsilon_i}) = Y)}{\Pr(\nabla g_k^{(t)}(i,\mathcal{D}_2) + Lap(\frac{\Delta f}{\epsilon_i}) = Y)}\\ &= \frac{\Pr(Lap(\frac{\Delta f}{\epsilon_i}) = Y - \nabla g_k^{(t)}(i,\mathcal{D}_1))}{\Pr(Lap(\frac{\Delta f}{\epsilon_i}) = Y - \nabla g_k^{(t)}(i,\mathcal{D}_2))}\\ &= \frac{\frac{\epsilon_i}{2\times\Delta f}\times e^{\frac{-|Y-\nabla g_k^{(t)}(i,\mathcal{D}_1)|\times\epsilon_i}{\Delta f}}}{\frac{\epsilon_i}{2\times\Delta f}\times e^{\frac{-|Y-\nabla g_k^{(t)}(i,\mathcal{D}_2)|\times\epsilon_i}{\Delta f}}}\\ &\leq e^{\frac{\epsilon_i\times|\nabla g_k^{(t)}(i,\mathcal{D}_1)-\nabla g_k^{(t)}(i,\mathcal{D}_2)|}{\Delta f}} \leq e^{\epsilon_i}.\end{aligned}$$

Thus, the i-th layer of the shared gradients satisfies ϵ_i-differential privacy. According to Theorem 1, if the i-th layer satisfies ϵ_i-differential privacy, the shared noisy gradients of C_k satisfies ϵ_k-differential privacy where $\epsilon_k = \sum_{i=1}\epsilon_i$. In FL, local datasets are independent and clients can be seen as a parallel sequence $C_1, ..., C_K$. Therefore according to Theorem 2, if the shared gradients of C_k satisfies ϵ_k-differential privacy, the global FL satisfies $\epsilon-$differential privacy, where $\epsilon = \max_{k\in[1,K]}\epsilon_k$ and K is the number of clients. □

V. Experiments

In this section, we evaluate DPFL in terms of model performance, privacy preservation, and efficiency. For model performance, we compare the accuracy of DPFL with the plain FedAvg [1] without any noise on three different datasets under FL scenario. For privacy preservation, we use the DLG attack methods with euclidean distance cost function [3] and cosine similarity cost function [4] to attack DPFL under different privacy budgets. Besides, we illustrate the necessity of adding noise on every layer of the shared gradients by extensive experiments. Finally, we compare the efficiency of DPFL and the LIP [6] in terms of computational cost.

A. Experimental Setups

We use PyTorch to implement DPFL and run the experiments with GeForce GTX 1080.

Datasets. We evaluate the performance of the proposed scheme based on MNIST [22], Fashion-MNIST [23], and CIFAR10 [24]. MNIST and Fashion-MNIST both contain 60,000 1x28x28 images in 10 classes which include 50,000 training images and 10,000 test images. CIFAR10 contains 60,000 3x32x32 images in 10 classes, including 50,000 training images and 10,000 test images.

Model Architectures. We use the same model architecture in the experiments of MNIST and Fashion-MNIST: two 5×5 convolutional layers each followed by a sigmoid layer and then a fully connected layer with 588 neurons. The model in the experiments of CIFAR10 consists of three 5×5 convolutional layers each followed by a sigmoid layer and then a fully connected layer with 768 neurons.

Settings of clients. We assume that the FL contains 100 clients, i.e., $K = 100$. All clients use the Adam optimizer with a learning rate of 0.01 ($\eta = 0.01$). In the experiments of MNIST and Fashion-MNIST, each client contains only two classes of samples which means the local datasets are non-i.i.d. Batch size is set to 128. In the experiments of CIFAR10, training samples are divided equally to each client as their local training data. Each client owns 10 classes of samples and the size of each class is equal. Batch size is set to 50.

Settings of the server. In each global epoch, 10% of the clients ($\lambda = 10\%$) are selected to update the global model. In the experiments of MNIST and Fashion-MNIST, we sample 1,000 images randomly from the test set to simulate the public dataset for fine-tuning. In the experiments on CIFAR10, the public dataset includes 3,000 samples, randomly sampled from the original test set as well. After implementing FedAvg [1], the server fine-tunes the global model for only one epoch.

Baseline Method. We compare our DPFL with the plain Fedavg [1] which does not add any noise on the shared gradients. For fair comparison, the baseline method FedAvg [1] also adopts the client selection strategy and the public dataset fine-tuning as DPFL.

Privacy Attack Setup. For the sake of comparability, we follow the experimental setup of [3]. The batch size of local update in the privacy preservation experiments is set to 1 which is the most simplest setting for DLG [3], [4] to recover training data from the shared gradients.

B. Model Performance

Directly adding noise to the shared gradients leads to a sharp drop of accuracy which destroys the utility of the model. In order to evaluate the impact of DPFL on accuracy, we train a model on MNIST which reaches 98.3% accuracy as a baseline model and test the baseline model after allocating privacy budget and adding noise with and without DPFL. In DPFL, privacy budget is allocated according to the importance value PoGZ of layers. For adding noise without DPFL, the privacy budget is allocated evenly between all layers, for example, if there is three layers in the model, each layer gets a privacy budget of $(1/3)\times\epsilon$. The experiment consists of three steps

TABLE II
PERFORMANCE OF DPFL.

ϵ	Baseline accuracy	w/wo DPFL		Proportion of Promotion
		w	wo	
$\epsilon = 0.5$	98.3%	11.7%	10.3 %	13.6%
$\epsilon = 1.0$	98.3%	26.4%	16.1%	64.0%
$\epsilon = 5.0$	98.3%	89.4%	40.0%	123.5%
$\epsilon = 10.0$	98.3%	97.6%	70.8%	37.9%

TABLE III
ACCURACY OF GLOBAL MODEL IN FEDERATED LEARNING.

Dataset	Baseline	DPFL ($\epsilon = 0.5$)	DPFL ($\epsilon = 1.0$)	DPFL ($\epsilon = 5.0$)	DPFL ($\epsilon = 10.0$)
MNIST	95.2%	88.8%	89.0%	95.2%	95.3%
Fashion-MNIST	84.9%	78.8%	80.8%	83.8%	84.9%
CIFAR10	45.1%	10.0%	10.0%	45.2%	45.1%

as follow: (1) Calculate the importance value PoGZ of the baseline model with validation set. (2) Train the baseline model on the whole train dataset for one epoch and add noise on the gradients. (3) Update the baseline model with the noised gradients and test the accuracy on test set.

From Table II, we can observe that allocating privacy budget with DPFL largely improves the model performance compared with evenly allocation. When $\epsilon = 10.0$, the model adding noise with DPFL reaches 97.6% which is 0.7% lower than the baseline model without noise, while the accuracy is 70.8% with allocating privacy budget evenly. When $\epsilon = 5.0$, the accuracy of DPFL is about 89.4%, while it is 40.0% without DPFL and the proportion of promotion is 123.5%. Besides, we can also observe that the accuracy of DPFL rises with the grow of privacy budget.

However, it is worth noting that in FL, the noise will be injected in every global epoch and the server will average the noisy shared gradients from different clients so that the noise will have greater impact on the accuracy of the new global model. We test the global model accuracy of DPFL and plain FedAvg [1] on MNIST and Fashion-MNIST in non-i.i.d FL setting under different privacy budgets.

The results of the experiments are shown in Figure 4 and Table III. We can observe that for MNIST when $\epsilon \in \{5.0, 10.0\}$, DPFL can achieve a similar accuracy as the baseline (95.2%). When $\epsilon \in \{0.5, 1.0\}$, more noise is needed to guarantee privacy and the accuracy drops slightly (88.8% for $\epsilon = 0.5$ and 89.0% for $\epsilon = 1.0$).

Similarly, as shown in Figure 5, for Fashion-MNIST, DPFL performs as well as the baseline model which reaches 84.9% when $\epsilon \in \{5.0, 10.0\}$ (83.8% for $\epsilon = 5.0$ and 84.9% for $\epsilon = 10.0$). And the accuracy has a moderately decline when $\epsilon \in \{0.5, 1.0\}$ (78.8% for $\epsilon = 0.5$ and 80.8% for $\epsilon = 1.0$).

For CIFAR10, we test the accuracy of DPFL in i.i.d setting. From Table III and Figure 6, we can observe that when $\epsilon \in \{5.0, 10.0\}$ the accuracy of DPFL rises slower than baseline but finally rises to 45.2% and 45.1% (the baseline is 45.1%). However, when $\epsilon \in \{0.5, 1.0\}$, more noise is introduced and DPFL can not guarantee the accuracy. It is a trade-off between privacy and model performance.

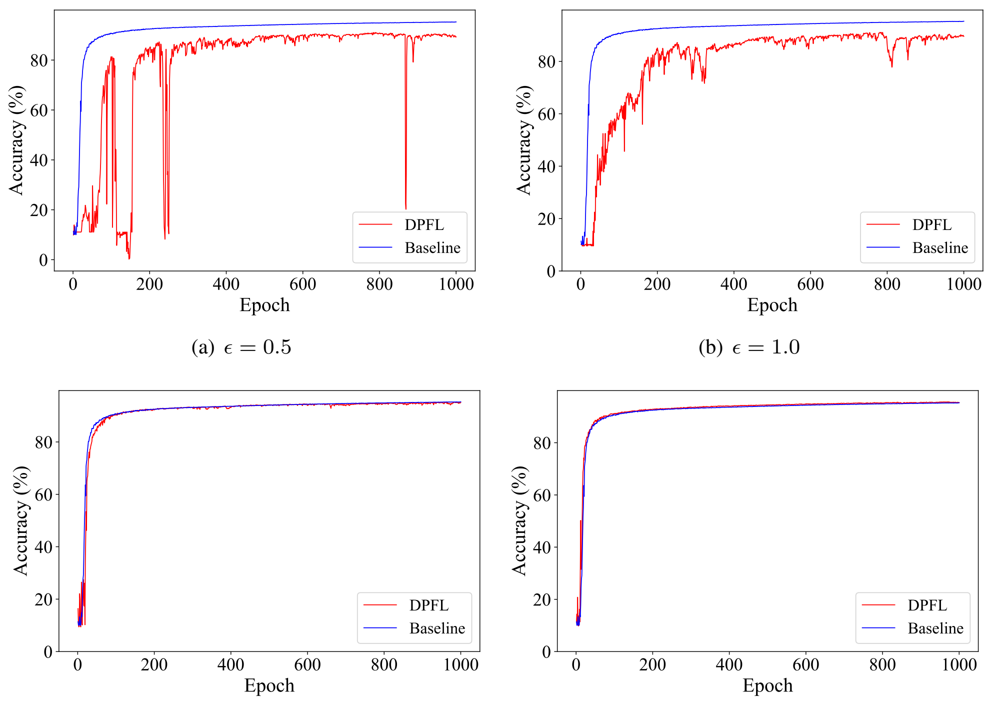
C. Privacy Preservation

We test the model's defense against DLG with euclidean distance cost function [3] under different privacy budgets $\epsilon \in \{0.5, 1.0, 5.0, 10.0\}$. As shown in Figure 7, when $\epsilon \in \{0.5, 1.0, 5.0, 10.0\}$, DPFL can prevent the training data recovery on MNIST, Fashion-MNIST, and CIFAR10. As shown in Figure 8, when the privacy budget is more tightened, the loss of DLG is much more greater which means it is harder for DLG to recover training data from the shared gradients. And we also test DPFL against the state-of-the-art DLG with the loss function in the form of cosine similarity [4]. As shown in Figure 9, DPFL can still protect data privacy.

Besides, in order to illustrate the necessity of adding noise on all layers, we also perform the state-of-the-art DLG [4] when the client only adds noise on part of layers of the gradients. The shared gradients still satisfies DP according to the composition property of DP when some layers are without noise. The research of Geiping J et al. [4] proved that the input to any fully connected layer can be reconstructed analytically independent of the remaining network architecture. That means it is necessary to add noise on the fully connected layers. However as shown in Figure 1(b), when only adding noise on fully connected layers, training data can be easily recovered with the gradients of convolutional layers. And as Figure 10 shows, when noise is only injected into the CONV1 and FC layers or CONV2 and FC layers, the leakage can still be performed. Only when noise is injected into all the layers, DLG [4] can not recover local data form the shared gradients. Thus, it is necessary to add noise on all layers of gradients.

D. Efficiency

We test the computation consumption of DPFL and LIP [6] during the FL training phase. It is worth noting that the



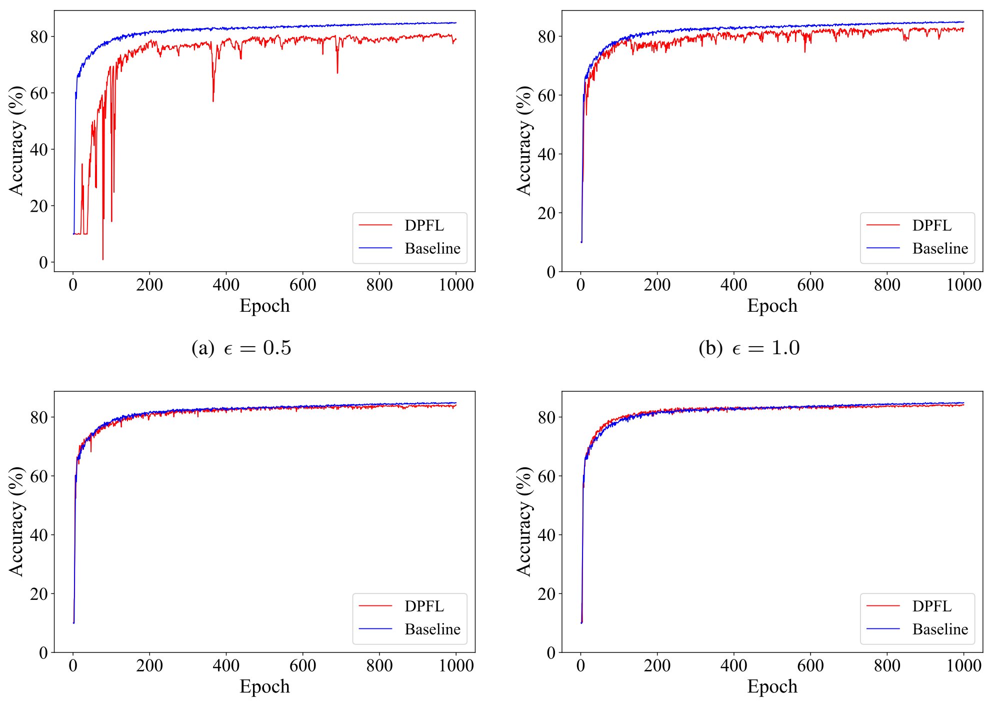
(a) $\epsilon = 0.5$

(b) $\epsilon = 1.0$

(c) $\epsilon = 5.0$

(d) $\epsilon = 10.0$

Fig. 4. Accuracy on MNIST (non-i.i.d). When $\epsilon \in \{5.0, 10.0\}$, DPFL can achieve a similar accuracy to the baseline which is without any noise. When $\epsilon \in \{0.5, 1.0\}$, the accuracy of DPFL drops slightly.



(a) $\epsilon = 0.5$

(b) $\epsilon = 1.0$

(c) $\epsilon = 5.0$

(d) $\epsilon = 10.0$

Fig. 5. Accuracy on Fashion-MNIST (non-i.i.d). DPFL performs well when $\epsilon \in \{5.0, 10.0\}$, and there is a slight drop when $\epsilon \in \{0.5, 1.0\}$.

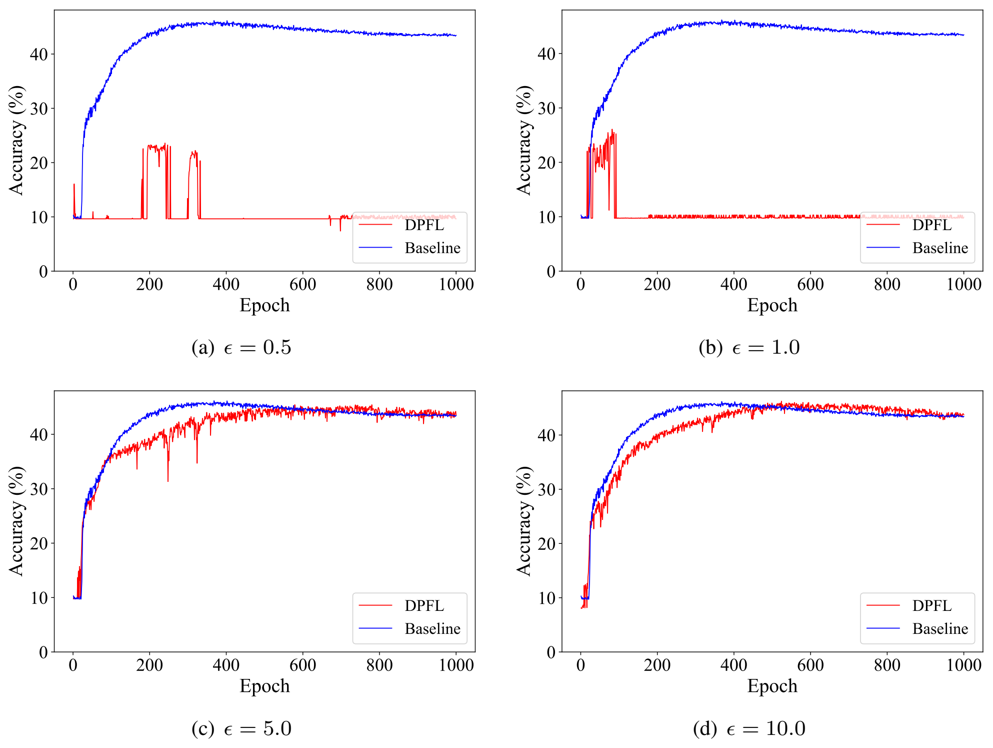


Fig. 6. Accuracy on CIFAR10 (i.i.d). When $\epsilon \in \{5.0, 10.0\}$, DPFL remains a similar accuracy to the baseline. When $\epsilon \in \{0.5, 1.0\}$, DPFL can not remain the model performance since more noise is introduced.

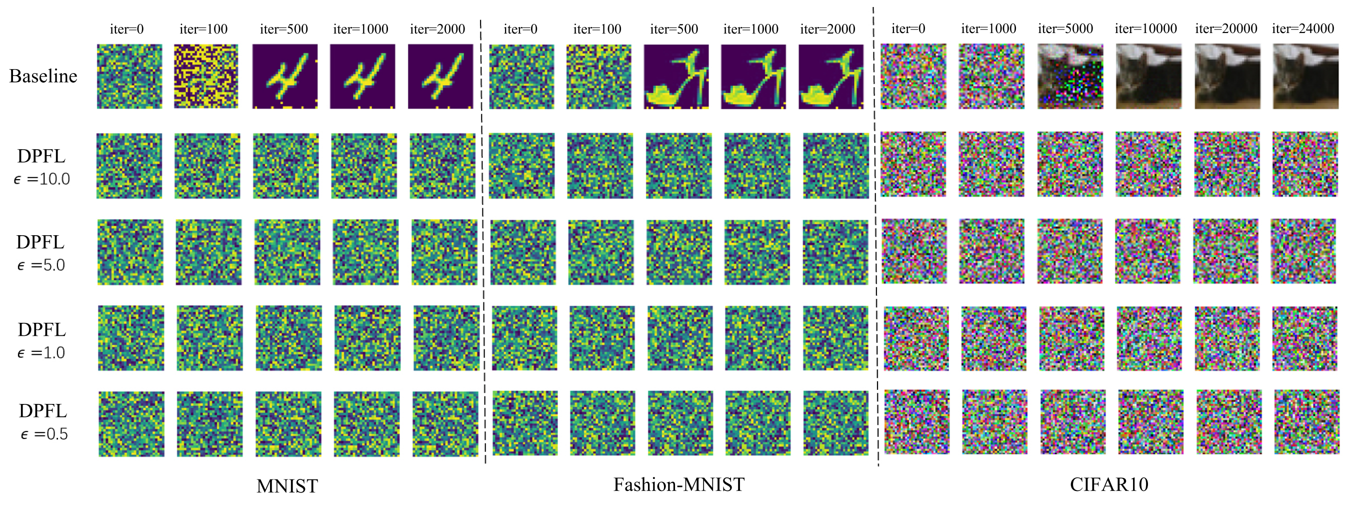


Fig. 7. Defense against DLG [3] with a euclidean distance as the loss function. The noise is injected into all the layers of the shared gradients with DPFL. The adversary tries to recover local data from the gradients and the leakage can not be preformed.

TABLE IV
COMPUTATION CONSUMPTION OF ADDING NOISE (MS).

	The model of MNIST	The model of CIFAR10
DPFL	4.08 ms	5.12 ms
LIP	1,508.98 ms	2,070.39 ms

pre-training phase of DPFL and LIP [6] on every client site is executed parallel in FL. And compared with the computation consumption of hundreds of global update epochs, the computation consumption of pre-training is negligible.

As shown in Table IV, the average time of adding Laplacian noise on gradients with LIP [6] is $1,508.98ms$ and $2,070.39ms$ for the two different models we use for MNIST and CIFAR10. And for DPFL, the time is $4.08ms$ and $5.12ms$ which means DPFL has a great advantage of the calculation efficiency on client sites. Even if DPFL needs fine-tuning the global model on the server, the total computation consumption of adding noise with DPFL and fine-tuning is still less than LIP [6] as it shown in Figure 11.

In general, DPFL can protect local training data from the

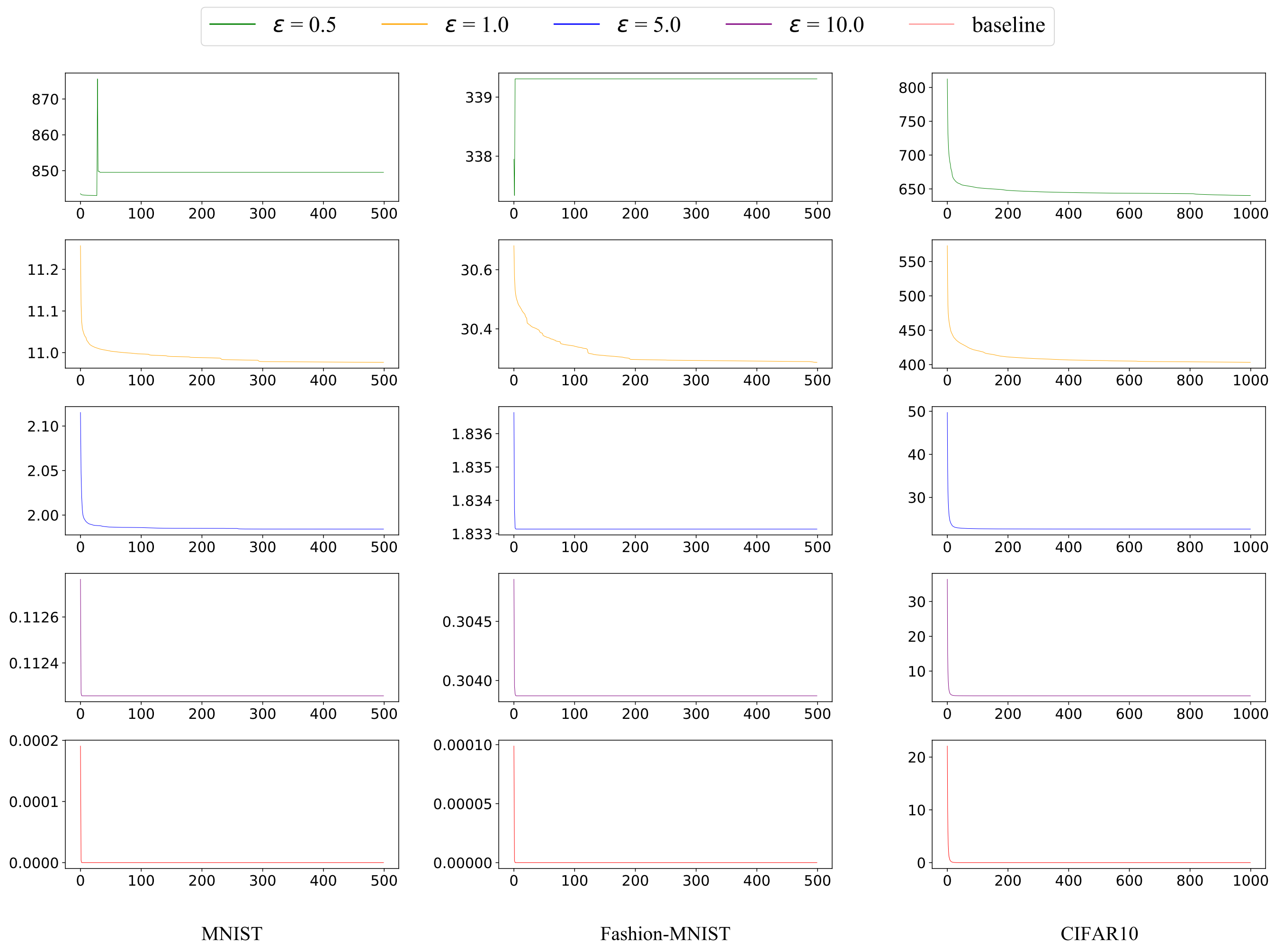


Fig. 8. Loss of DLG [3] with a euclidean distance loss function. We observe that the noised gradients makes the loss of DLG [3] converge to a large value which means it is harder for the adversary to recover the training data.

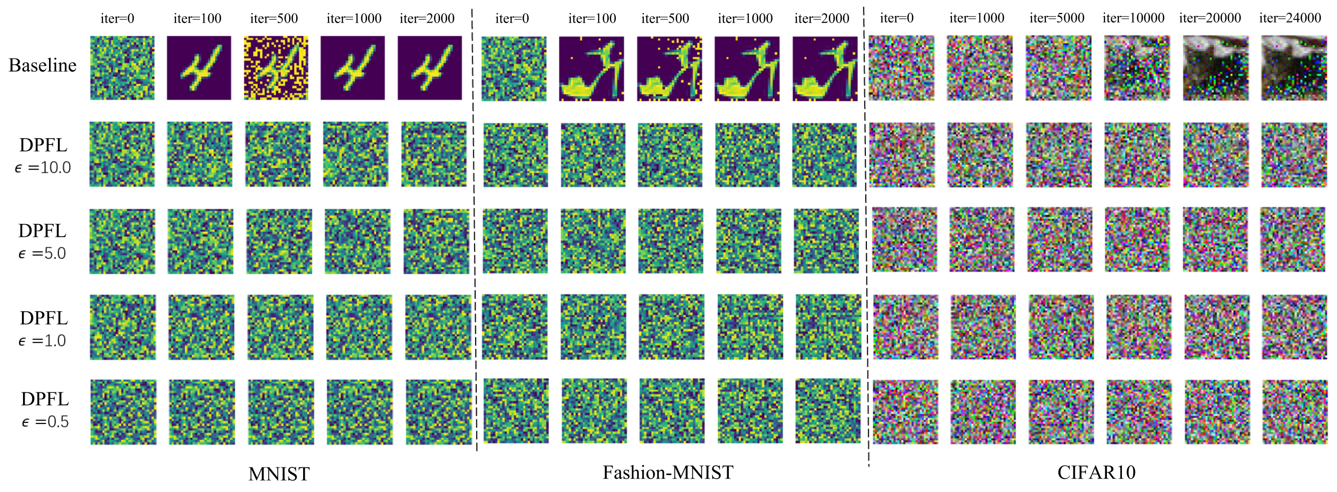


Fig. 9. Defense against DLG [4] with a cosine similarity loss function. DPFL still provides privacy guarantee.

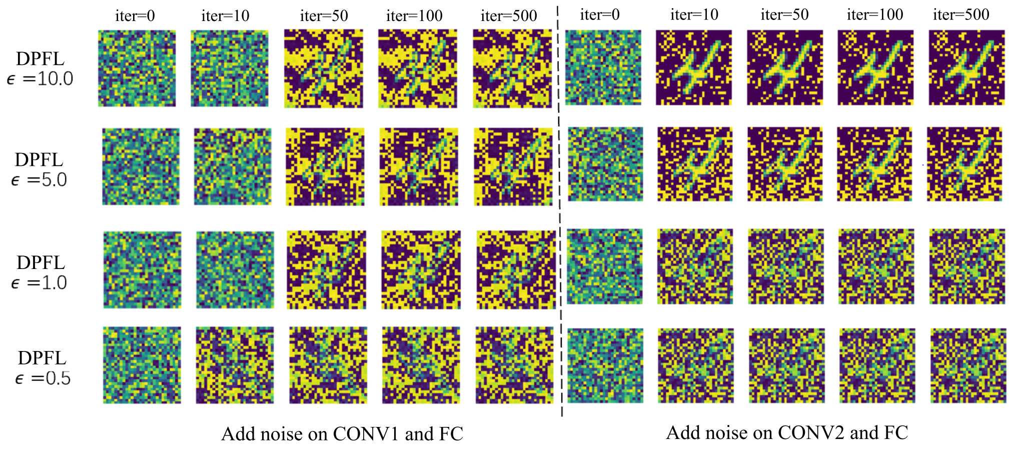


Fig. 10. Defense against DLG [4] when only part of the model is noised. The adversary tries to recover local data from the shard gradients when only (1) CONV1 and FC layers (2) CONV2 and FC layers are injected with noise. The leakage can still be performed when some layers are not under protection.

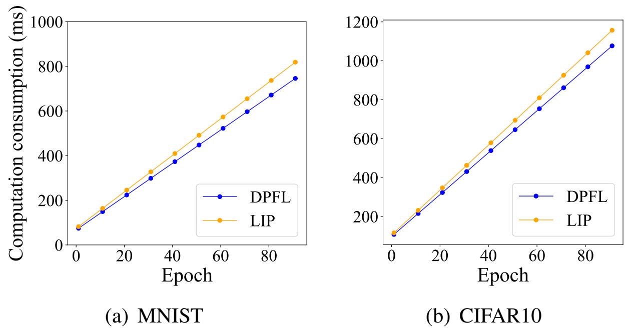


Fig. 11. Computational consumption of DPFL and LIP for model training.

state-of-the-art DLG attack [3], [4] while remaining high model performance and efficiency.

VI. Conclusion

Although federated learning largely improves the level of data security, sharing gradients still faces huge privacy challenges, e.g., deep leakage from gradients (DLG) [3], [4]. In this paper, we proposed an efficient and fine-grained differential privacy federated learning (DPFL) scheme, to protect shared gradients of local clients while preserving model performance and efficiency. Specifically, we proposed a fine-grained method to allocate the privacy budget according to the importance of layers. We also adopted an active client selection strategy [13] and fine-tuned the global model to further improve efficiency and model accuracy. We evaluated our DPFL on three benchmark datasets under both i.i.d. and non-i.i.d. scenarios. The results showed that our method can preserve data privacy while maintaining high accuracy and efficiency.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of China under Grant 62171248, the Guangdong Basic and Applied Basic Research Foundation under Grant 2019A1515110644, the China Postdoctoral Science Foundation under Grant 2020M670374, the R&D Program of Shenzhen under Grant JCYJ20180508152204044, and the PCNL KEY project (PCL2021A07).

VII. References Section

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *AISTATS*, 2017, pp. 1273–1282.
[2] M. A. Rahman, T. Rahman, R. Laganière, and N. Mohammed, “Membership inference attack against differentially private deep learning model,” *Trans. Data Priv.*, pp. 61–79, 2018.
[3] L. Zhu, Z. Liu, and S. Han, “Deep leakage from gradients,” in *NeurIPS*, 2019, pp. 14 747–14 756.
[4] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, “Inverting gradients - how easy is it to break privacy in federated learning?” in *NeurIPS*, 2020.
[5] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, “Verifynet: Secure and verifiable federated learning,” *IEEE Trans. Inf. Forensics Secur.*, pp. 911–926, 2020.
[6] X. Liu, H. Li, G. Xu, S. Liu, Z. Liu, and R. Lu, “PADL: privacy-aware and asynchronous deep learning for iot applications,” *IEEE Internet Things J.*, pp. 6955–6969, 2020.
[7] M. Abadi, A. Chu, I. J. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, “Deep learning with differential privacy,” in *ACM SIGSAC*, 2016, pp. 308–318.
[8] Z. Bu, J. Dong, Q. Long, and W. J. Su, “Deep learning with gaussian differential privacy,” *Harvard data science review*, 2020.
[9] C. Dwork, “Differential privacy,” in *ICALP*, 2006, pp. 1–12.
[10] C. Dwork, F. McSherry, K. Nissim, and A. D. Smith, “Calibrating noise to sensitivity in private data analysis,” in *TCC*, 2006, pp. 265–284.
[11] C. Dwork, “A firm foundation for private data analysis,” *Commun. ACM*, pp. 86–95, 2011.
[12] C. Dwork and A. Roth, “The algorithmic foundations of differential privacy,” *Found. Trends Theor. Comput. Sci.*, pp. 211–407, 2014.
[13] J. Goetz, K. Malik, D. Bui, S. Moon, H. Liu, and A. Kumar, “Active federated learning,” *arXiv preprint arXiv:1909.12641*, 2019.
[14] R. C. Geyer, T. Klein, and M. Nabi, “Differentially private federated learning: A client level perspective,” *arXiv preprint arXiv:1712.07557*, 2017.
[15] P. Kairouz, S. Oh, and P. Viswanath, “The composition theorem for differential privacy,” in *ICML*, 2015, pp. 1376–1385.
[16] F. McSherry and K. Talwar, “Mechanism design via differential privacy,” in *FOCS*, 2007, pp. 94–103.
[17] J. Bai, Y. Li, J. Li, X. Yang, Y. Jiang, and S.-T. Xia, “Multinomial random forest,” *Pattern Recognition*, p. 108331, 2021.
[18] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, “Federated learning with non-iid data,” *arXiv preprint arXiv:1806.00582*, 2018.
[19] X. Yao, T. Huang, R.-X. Zhang, R. Li, and L. Sun, “Federated learning with unbiased gradient aggregation and controllable meta updating,” *arXiv preprint arXiv:1910.08234*, 2019.

[20] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Secur.*, pp. 3454–3469, 2020.

[21] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," *arXiv preprint arXiv:1607.03250*, 2016.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, pp. 2278–2324, 1998.

[23] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[24] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

Linghui Zhu received the B.S. degree in Computer Science and Technology from Nankai University, Tianjin, China, in 2020. She is currently pursuing the master degree in Tsinghua Shenzhen International Graduate School, Tsinghua University. Her research interests are in the domain of the big data security and federated learning.

Xinyi Liu received the B.S. degree in electronic information engineering from School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, China, in 2019. She is currently working toward the master's degree with Tsinghua Shenzhen International Graduate School, Tsinghua University, Beijing, China. Her research interests include robust federated learning and weakly supervised learning.

Yiming Li is currently a Ph.D. candidate from Tsinghua-Berkeley Shenzhen Institute, Tsinghua Shenzhen International Graduate School, Tsinghua University. Before that, he received his B.S. degree in Mathematics and Applied Mathematics from Ningbo University in 2018. His research interests are in the domain of AI security, especially backdoor learning, adversarial learning, and data privacy. He is the senior program committee member of AAAI'22 and the reviewer of TDSC, TCSVT, TII, etc.

Dr. Xue Yang received the Ph.D. degree in information and communication engineering from Southwest Jiaotong University, China, in 2019. She was a visiting student at the Faculty of Computer Science, University of New Brunswick, Canada, from 2017 to 2018. She is currently a postdoctoral fellow with the Tsinghua Shenzhen International Graduate School, Tsinghua University, China. Her research interests include big data security and privacy, applied cryptography, and federated learning.

Dr. Shu-Tao Xia received the B.S. degree in mathematics and the Ph.D. degree in applied mathematics from Nankai University, Tianjin, China, in 1992 and 1997, respectively. Since January 2004, he has been with the Graduate School at Shenzhen of Tsinghua University, Guangdong, China, where he became a full professor in 2007. From March 1997 to April 1999, he was with the research group of information theory, Department of Mathematics, Nankai University, Tianjin, China. From September 1997 to March 1998 and from August to September 1998, he visited the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong. His current research interests include coding and information theory, networking, machine learning and computer vision. He published more than one hundred papers on peer-reviewed journals and conferences, including TIT, TSP, TCOM, TNNLS, NeurIPS/ICML/ICLR, CVPR/ICCV/ECCV, etc.

Dr. Rongxing Lu is currently an associate professor at the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Canada. He is a Fellow of IEEE. His research interests include applied cryptography, privacy enhancing technologies, and IoT-Big Data security and privacy. He has published extensively in his areas of expertise, and was the recipient of 9 best (student) paper awards from some reputable journals and conferences. Currently, Dr. Lu serves as the Vice-Chair (Conferences) of IEEE ComSoc CIS-TC (Communications and Information Security Technical Committee). Dr. Lu is the Winner of 2016-17 Excellence in Teaching Award, FCS, UNB.