An Accuracy-Lossless Perturbation Method for Defending Privacy Attacks in Federated Learning

Xue Yang[*]
Southwest Jiaotong University
Tsinghua University
xueyang.swjtu@gmail.com

Yan Feng[*]
Meiuan
fengyan14@meituan.com

Weijun Fang[†]
Shandong University
nankaifwj@163.com

Jun Shao
Zhejiang Gongshang University

Xiaohu Tang
Southwest Jiaotong University

Shu-Tao Xia
Tsinghua University
Research Center of Artificial Intelligence, Peng Cheng Laboratory

Rongxing Lu
University of New Brunswick

ABSTRACT

Although federated learning improves privacy of training data by exchanging local gradients or parameters rather than raw data, the adversary still can leverage local gradients and parameters to obtain local training data by launching reconstruction and membership inference attacks. To defend against such privacy attacks, many noises perturbed methods (like differential privacy or CountSketch matrix) have been widely designed. However, the strong defence ability and high learning accuracy of these schemes cannot be ensured at the same time, which will impede the wide application of FL in practice (especially for medical or financial institutions that require both high accuracy and strong privacy guarantee). To overcome this issue, we propose *an efficient model perturbation method for federated learning* to defend against reconstruction and membership inference attacks launched by curious clients. On the one hand, similar to the differential privacy, our method also selects random numbers as perturbed noises added to the global model parameters, and thus it is very efficient and easy to be integrated in practice. Meanwhile, the random selected noises are positive real numbers and the corresponding value can be arbitrarily large, and thus the strong defence ability can be ensured. On the other hand, unlike differential privacy or other perturbation methods that cannot eliminate added noises, our method allows the server to recover the true aggregated gradients by eliminating the added noises. Therefore, our method does not hinder learning accuracy at all. Extensive experiments demonstrate that for both regression and classification tasks, our method achieves the same accuracy as non-private approaches and outperforms the state-of-the-art defence schemes. Besides, the defence ability of our method against reconstruction and membership inference attack is significantly better than the state-of-the-art related defence schemes.

CCS CONCEPTS

• **Security and privacy → Privacy-preserving protocols**.

KEYWORDS

privacy-preserving, federated learning, privacy attack

ACM Reference Format:
Xue Yang, Yan Feng, Weijun Fang, Jun Shao, Xiaohu Tang, Shu-Tao Xia, and Rongxing Lu. 2022. An Accuracy-Lossless Perturbation Method for Defending Privacy Attacks in Federated Learning. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3485447.3512233

1 INTRODUCTION

With the continued emergence of privacy breaches and data abuse [30], data privacy and security issues gradually impede the flourishing development of deep learning [32]. In order to mitigate such privacy concerns, *federated learning* (FL) [19] has recently been presented as an appealing solution. FL is essentially a distributed learning framework where clients collaboratively train a shared global model under the orchestration of a central server, while ensuring each client's raw data is stored locally and not exchanged.

FL seemingly can protect the training data of clients by exchanging local gradients and current model parameters rather than raw data, however, sharing local gradients and model parameters is still a well-known privacy risk [23, 35]. Specifically, inference attacks [23] has been widely adopted to reconstruct or infer the property of clients' training data based on the model output. Inference attacks on FL mainly includes two attacks: membership inference attacks and reconstruction attacks. The goal of *reconstruction attacks* is to recover original training data as accurate as possible. The goal of *membership inference attacks* is to infer if a particular individual data record was included in the training dataset. Note that we

[*] The first two authors contributed equally to this research.
[†] Corresponding author.

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France.
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9096-5/22/04…$15.00
https://doi.org/10.1145/3485447.3512233

mainly focus on the privacy attacks that will leak the training data of clients, and thus other attacks are beyond the scope of this paper.

To defend against inference attacks launched by curious clients, many privacy-preserving federated learning schemes [20, 24, 29] widely adopt the *differential privacy* technique due to its theoretical guarantee of privacy protection and low computational and communication complexity. However, there is a trade-off between privacy guarantee and learning accuracy. If the added noise is strong, the differential privacy works for defence, but the noise inevitably hurts the accuracy and may even stop FL from making progress [10]. If the noise is not strong enough, the defence may face failure. In order to overcome such drawback, many defenses [7, 21, 26] have been designed. For example, the work in [26] has suggested to randomly select and share a small fraction of gradient elements to reduce privacy loss. SPN [7] have leveraged element-wise adaptive gradients perturbations to defeat reconstruction attacks and maintain high model accuracy. In DBCL[21], a CountSketch matrix is used to perturb global model parameters for defending against reconstruction and membership inference attacks.

Unfortunately, the defence ability and learning accuracy of these schemes are still unsatisfactory, especially for medical or financial institutions that require both high accuracy and strong privacy guarantee. The main reason is that the perturbed noises added in gradients or model parameters cannot be eliminated, which will inevitably reduce the learning accuracy compared with the non-private training. Meanwhile, the defence ability is not strong enough due to the utility consideration in practice (i.e., It should avoid reducing the learning accuracy too much to make the trained model unusable). Therefore, designing an efficient privacy-preserving scheme applied to FL, which ensures robustness to inference attacks while maintaining a high learning accuracy remains an open problem [13, 16].

In this paper, we aim to efficiently address such challenge and propose *an efficient model perturbation method for FL to defend against inference attacks launched by curious clients and ensure high learning accuracy at the same time.* The main novelty and contributions are two-folds:

- We propose an efficient model perturbation method to prevent clients obtaining true global model parameters and local gradients, and thus can defend against reconstruction and membership inference attacks [23]. On the one hand, similar to the differential privacy, our method also selects random numbers as noises added to the global model parameters, and thus it is very efficient and easy to be integrated. Note that the randomly selected noises are positive real number and the corresponding value can be arbitrarily large, which shows a strong defence ability. On the other hand, unlike differential privacy [26] or other perturbation methods [7, 21] that cannot eliminate the added noises, our method ensures that the server can recover the aggregated true gradients by eliminating the added noises. Therefore, our method does not hinder learning accuracy.
- In order to evaluate our method from ***learning accuracy*** and ***inference attacks defence***, we conduct extensive experiments on several large-sale datasets, and compare our method with several state-of-the-art related schemes. Empirical results show that our scheme can achieve the same learning accuracy as the non-private training scheme (i.e., FedAvg [19]), which demonstrates that our method does not harm learning accuracy. Meanwhile, both learning accuracy and defence ability of our method are better than the related defence schemes [7, 21, 26]. More specifically, for the membership inference attack, our method achieves attack success rate of around 50%, which is equivalent to blind guessing. For the reconstruction attack, the attack success rate of our method is around 10% for 10-classes datasets, which is also equivalent to blind guessing.

Note that we admit that our method alone does not fundamentally defend against all attacks launched from curious clients. As claimed in [21], there does not exist any defense that is effective for all privacy attacks. Meanwhile, similar to [7, 21, 26], we focus on inference attacks launched by curious clients and do not consider the attacks launched by the server. We assume the server is honest and clients are honest-but-curious. To the best of our knowledge, this assumption is widespread in existing works [23, 35].

In recent years, more and more web browsers have replaced traditional tracking tools such as third-party cookies with FL to improve user experience and preserve privacy. For example, Google proposed Federated Learning of Cohorts (FLoC) [1], a type of web tracking through federated learning method. It groups people into "cohorts" based on their browsing history for interest-based advertising. FLoC also prevents web-advertisement fraud. Therefore, research on the efficiency and privacy of such federated learning algorithms can further improve the quality of these browser-based applications and elevate the issue of endpoint security.

2 PRELIMINARIES

In this section, we outline concepts of the FL and the Hadamard product, which will serve as the basis of our scheme.

2.1 Federated learning

We focus on training Deep Neural Networks (DNNs) in the FL framework, which have been widely adopted as the infrastructure of deep learning models to solve many complex tasks [25]. Consider a FL with K clients, denoted as $\{C_1, \dots, C_K\}$, and each client C_k has the local training dataset $\mathcal{D}_k$. The FL aims to solve an optimization problem to obtain the optimal global parameter W [17, 19]:

$$\min_{W} F(W) \triangleq \sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} F(W, \mathcal{D}_k), \tag{1}$$

where $|\mathcal{D}_k|$ is the sample size of the client C_k and $|\mathcal{D}| = \sum_{k=1}^{K} |\mathcal{D}_k|$. $F(W, \mathcal{D}_k)$ is the local object of C_k defined by

$$F(W, \mathcal{D}_k) \triangleq \frac{1}{|\mathcal{D}_k|} \sum_{(\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i) \in \mathcal{D}_k} \mathcal{L}\left(W; (\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i)\right), \tag{2}$$

where $(\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i)$ is the training sample and $\mathcal{L}(\cdot;\cdot)$ is a user-specified loss function.

In order to find the optimal result for Eq. (1), the server and all clients cooperatively perform the following three phases: 1) *global model broadcasting*, 2) *local model training*, and 3) *global model update*.

1) Global model broadcasting. The server broadcasts the current model parameter $W = \{W^{(l)}\}_{l=1}^{L}$ to clients for local model training.

2) Local model training. Once receiving the global model parameter $W = \{W^{(l)}\}_{l=1}^{L}$, each client C_k computes local gradients $\nabla F(W, \mathcal{D}_k)$ with the local training dataset $\mathcal{D}_k$ by running the stochastic gradient descent (SGD) algorithm that includes *forward propagation* and *backward propagation* steps.

- **Forward propagation:** for each training sample $(\boldsymbol{x}, \bar{\boldsymbol{y}}) \in \mathcal{D}_k$, the output vector $\boldsymbol{y}^{(l)} = (y_1^{(l)}, y_2^{(l)}, \ldots, y_{n_l}^{(l)}) \in \mathbf{R}^{n_l}$ of the l-th layer is computed as
$$\boldsymbol{y}^{(l)} = \begin{cases} \text{ReLU}\left(W^{(l)}\boldsymbol{y}^{(l-1)}\right), \text{for } 1 \le l \le L-1, \\ W^{(l)}\boldsymbol{y}^{(l-1)}, \text{ for } l = L, \end{cases} \tag{3}$$
where $\boldsymbol{y}^{(0)} = \boldsymbol{x}$.
- **Backward propagation:** let the gradient of the loss respect to $W^{(l)}$ be $\frac{\partial \mathcal{L}(W;(\boldsymbol{x},\bar{\boldsymbol{y}}))}{\partial W^{(l)}} \in \mathbf{R}^{n_l \times n_{l-1}}$. Based on the chain rule, the (i,j)-th entry of $\frac{\partial \mathcal{L}(W;(\boldsymbol{x},\bar{\boldsymbol{y}}))}{\partial W^{(l)}}$ is computed as:
$$\frac{\partial \mathcal{L}(W;(\boldsymbol{x},\bar{\boldsymbol{y}}))}{\partial w_{ij}^{(l)}} = \frac{\partial \mathcal{L}(W;(\boldsymbol{x},\bar{\boldsymbol{y}}))}{\partial \boldsymbol{y}^{(L)}} \frac{\partial \boldsymbol{y}^{(L)}}{\partial \boldsymbol{y}^{(L-1)}} \cdots \frac{\partial \boldsymbol{y}^{(l+1)}}{\partial y_i^{(l)}} \frac{\partial y_i^{(l)}}{\partial w_{ij}^{(l)}},$$
where $i = 1, 2, \ldots, n_l$ and $j = 1, 2, \ldots, n_{l-1}$.

Similarly, for every sample $(\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i) \in \mathcal{D}_k$, C_k can obtain the corresponding local gradient $\frac{\partial \mathcal{L}(W;(\boldsymbol{x}_i,\bar{\boldsymbol{y}}_i))}{\partial W}$, and then computes the average local gradient as:
$$\nabla F(W, \mathcal{D}_k) = \frac{1}{|\mathcal{D}_k|} \sum_{(\boldsymbol{x}_i,\bar{\boldsymbol{y}}_i)\in\mathcal{D}_k} \frac{\partial \mathcal{L}(W;(\boldsymbol{x}_i,\bar{\boldsymbol{y}}_i))}{\partial W}. \tag{4}$$
Finally, C_k uploads $\nabla F(W, \mathcal{D}_k)$ to the server for model update.

3) Global model update. After receiving the local gradients $\nabla F(W, \mathcal{D}_k)$ from all clients, the server aggregates and updates the current model parameter W for the next iteration. Specifically, given the learning rate η, the updated parameter is computed as
$$W \Leftarrow W - \eta \sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} \nabla F(W, \mathcal{D}_k). \tag{5}$$
After that, the server distributes the updated parameter W to all clients for the next iteration.

2.2 Hadamard Product

The Hadamard product [11] takes two matrices of the same dimensions and produces another matrix of the same dimension as the operands.

Definition 1. *For two matrices A and B of the same dimension $m \times n$, the Hadamard product $A \circ B$ (or $A \odot B$) is a matrix of the same dimension as the operands, with elements given by*
$$(A \circ B)_{ij} = (A \odot B)_{ij} = A_{ij}B_{ij}$$

Two properties of Hadamard product are given as follows:

- For any two matrices A and B, and a diagonal matrix D, we have
$$D(A \circ B) = (DA) \circ B \text{ and } (A \circ B)D = (AD) \circ B. \tag{6}$$
- For any two column vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, the Hadamard product is $\boldsymbol{a} \circ \boldsymbol{b} = D_{\boldsymbol{a}}\boldsymbol{b}$, where $D_{\boldsymbol{a}}$ is the corresponding diagonal matrix with the vector $\boldsymbol{a}$ as its main diagonal.

3 THREAT MODEL AND DESIGN GOALS

In this paper, we mainly focus on the privacy attacks launched by clients. Similar to [7, 21], the server and clients are assumed to be honest and honest-but-curious, respectively, which means that all clients honestly follow the underlying scheme, but attempt to infer other clients' data privacy by launching privacy attacks. Note that this assumption has wide practicability [7, 21]. For example, the server can be the service provider who wants to train the deep model with big data from multiple data owners to provide prediction services. Obviously, the model is the most important asset for the server. As introduced in Section 2.1, during each iteration, each client C_k can obtain the updated model parameter W and compute the sum of other clients' gradients denoted as $\sum_{i \neq k} \nabla F(W, \mathcal{D}_k)$. Knowing the model parameters, gradients or both, curious clients may try to infer data privacy of other clients. Similar to [7, 21], we mainly focus on two state-of-the-art inference attacks in federated learning called **reconstruction attack** and **membership inference attack**. In the **reconstruction attack**, the curious client tries to reconstruct sensitive features of the records in the training set of others. In the **membership inference attack**, the curious client attempts to infer whether a certain data record is included in the training dataset with the model parameters.

Design goals. In order to defend against privacy attacks launched by clients and ensure the utility of training model, design goals of our scheme mainly include the following two aspects:

- **Confidentiality**: The proposed scheme should ensure clients cannot infer private data of other clients by reconstruct and membership inference attacks with exchanged global model parameters and local gradients. Similar to [21], we need ensure that clients cannot know true model parameters.
- **High accuracy**: As demonstrated in [7, 13], many differential privacy-based approaches strengthen the privacy preservation capability at the expense of accuracy, which will hinder the application of FL in practice . Therefore, ensuring high model accuracy is also our design goal. Specifically, our scheme tries to achieve the same model accuracy as the plain training model.

4 OUR PROPOSED METHOD

In this section, we describe our proposed privacy-preserving deep model training with ReLU non-linear activation in details, which can be easily applied on the state-of-the-art models such as ResNet [9] and DenseNet [12]. According to Section 2.1, the overall framework of our proposed method, as shown in Fig. 1, mainly includes three steps: global model perturbation, local model training and global model update.

4.1 Global Model Perturbation

The goal of *global model perturbation* is to prevent curious clients from obtaining the true model parameters $W = \{W^{(l)}\}_{l=1}^{L}$ and gradient $\nabla F(W, \mathcal{D}_k)$. Specifically, the server selects random one-time-used noises for different iterations and perturbs global parameters.

- **Random noises selection.** The server selects random one-time-used noises for different iterations as follows.
  - Randomly select multiplicative noisy vectors

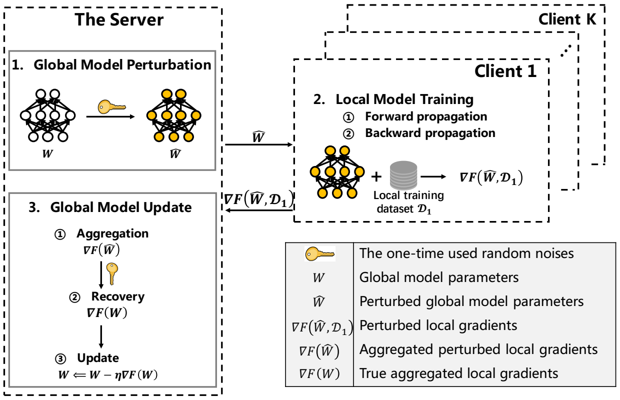
The Server
1. Global Model Perturbation
W
$\widehat{W}$
$\widehat{W}$
$\nabla F(\widehat{W},\mathcal{D}_1)$
Client K
Client 1
2. Local Model Training
① Forward propagation
② Backward propagation
+
Local training dataset $\mathcal{D}_1$
$\nabla F(\widehat{W},\mathcal{D}_1)$
3. Global Model Update
① Aggregation
$\nabla F(\widehat{W})$
② Recovery
$\nabla F(W)$
③ Update
$W \Leftarrow W - \eta\nabla F(W)$
The one-time used random noises
W Global model parameters
$\widehat{W}$ Perturbed global model parameters
$\nabla F(\widehat{W},\mathcal{D}_1)$ Perturbed local gradients
$\nabla F(\widehat{W})$ Aggregated perturbed local gradients
$\nabla F(W)$ True aggregated local gradients

Figure 1: The overall framework of our proposed method.

$\boldsymbol{r}^{(l)} = (r_1^{(l)}, r_2^{(l)}, \dots, r_{n_l}^{(l)}) \in \mathbf{R}_{>0}^{n_l}$ for $1 \le l \le L-1$ and an additive noisy vector $\boldsymbol{r}^{(a)} = (r_1^{(a)}, r_2^{(a)}, \dots, r_{n_L}^{(a)}) \in \mathbf{R}^{n_L}$ with pairwise different components.

- Define a disjoint partition $\sqcup_{s=1}^{m}\{I_s\}$ of $\{1, 2, \dots, n_L\}$ [1] such that $\cup_{s=1}^{m} I_s = \{1, 2, \dots, n_L\}$ and $I_i \cap I_j = \emptyset$ for any $i \ne j$. Randomly select noisy numbers $\gamma_{I_1}, \gamma_{I_2}, \dots, \gamma_{I_m} \in \mathbf{R}$. Then let $\boldsymbol{\gamma} = (\gamma_1, \gamma_2, \dots, \gamma_{n_L})$, whose coordinates are given as $\gamma_i = \gamma_{I_s}$ for $i \in I_s$ and $s = 1, 2, \dots, m$.

- **Parameter perturbation:** With the random selected noises, the global parameters $W = \{W^{(l)}\}_{l=1}^{L}$ are perturbed as: for the l-th layer's parameter matrix $W^{(l)} \in \mathbf{R}^{n_l \times n_{l-1}}$, the perturbed parameter matrix $\widehat{W}^{(l)}$ is

$$\widehat{W}^{(l)} = \begin{cases} R^{(l)} \circ W^{(l)}, \text{ for } 1 \le l \le L-1, \\ R^{(l)} \circ W^{(l)} + R^{(a)}, \text{ for } l = L, \end{cases} \tag{7}$$

where $R^{(l)} \in \mathbf{R}^{n_l \times n_{l-1}}$ and $R^{(a)} \in \mathbf{R}^{n_L \times n_{L-1}}$ satisfy

$$R_{ij}^{(l)} = \begin{cases} r_i^{(1)}, \text{ when } l = 1 \\ r_i^{(l)}/r_j^{(l-1)}, \text{ when } 2 \le l \le L-1 \\ 1/r_j^{(L-1)}, \text{ when } l = L \end{cases} \tag{8}$$

$$R_{ij}^{(a)} = \gamma_i \cdot r_i^{(a)}, \tag{9}$$

where $i \in [1, n_l]$ and $j \in [1, n_{l-1}]$ in Eq. (8), and $i \in [1, n_L]$ and $j \in [1, n_{L-1}]$ in Eq. (9).

Finally, the server keeps $\left(\{\boldsymbol{r}^{(l)}\}_{l=1}^{L-1}, \boldsymbol{\gamma}\right)$ secret and broadcasts the current perturbed parameters $\widehat{W} = \{\widehat{W}^{(l)}\}_{l=1}^{L}$ and the additive noisy vector $\boldsymbol{r}^{(a)}$ to all clients for local model training.

4.2 Local Model Training

After receiving the perturbed parameters $\widehat{W} = \{\widehat{W}^{(l)}\}_{l=1}^{L}$, each client C_k conducts local model training with local training data $\mathcal{D}_k$. As introduced in Section 2.1, for each sample $(\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i) \in \mathcal{D}_k$, C_k executes *forward propagation* and *backward propagation* to compute gradients. Note that we ignore the subscript i of $(\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i)$ for simplicity unless other specification.

(1) Forward Propagation. Obviously, with the perturbed parameters $\widehat{W}$, the output of each layer is also perturbed, denoted as $\hat{\boldsymbol{y}}^{(l)} \in \mathbf{R}^{n_l}$ for $1 \le l \le L$, which is computed as

$$\hat{\boldsymbol{y}}^{(l)} = \begin{cases} \text{ReLU}\left(\widehat{W}^{(l)}\hat{\boldsymbol{y}}^{(l-1)}\right), \text{ for } 1 \le l \le L-1. \\ \widehat{W}^{(l)}\hat{\boldsymbol{y}}^{(l-1)}, \text{ for } l = L. \end{cases} \tag{10}$$

where $\hat{\boldsymbol{y}}^{(0)} = \boldsymbol{x}$ is the input feature of the sample $(\boldsymbol{x}, \bar{\boldsymbol{y}})$. Theorem 1 shows the important relations between the perturbed output $\hat{\boldsymbol{y}}^{(l)} \in \mathbf{R}^{n_l}$ and the true output $\boldsymbol{y}^{(l)} \in \mathbf{R}^{n_l}$ given in Eq. (3).

Theorem 1. *For $1 \le l \le L$, the perturbed output vector $\hat{\boldsymbol{y}}^{(l)}$ computed in Eq.* (10) *and the true output vector $\boldsymbol{y}^{(l)}$ computed in Eq.* (3) *have the following relationship:*

$$\hat{\boldsymbol{y}}^{(l)} = \boldsymbol{r}^{(l)} \circ \boldsymbol{y}^{(l)}, \text{ when } 1 \le l \le L-1. \tag{11}$$

$$\hat{\boldsymbol{y}}^{(L)} = \boldsymbol{y}^{(L)} + \alpha\boldsymbol{\gamma} \circ \boldsymbol{r}^{(a)} = \boldsymbol{y}^{(L)} + \alpha\boldsymbol{r}. \tag{12}$$

where $\alpha = \sum_{i=1}^{n_{L-1}} \hat{y}_i^{(L-1)}$ and $\boldsymbol{r} = \boldsymbol{\gamma} \circ \boldsymbol{r}^{(a)}$.

Proof. See Appendix A.1. □

(2) Backward Propagation. After obtaining the perturbed outputs $\{\hat{\boldsymbol{y}}^{(l)}\}_{l=1}^{L}$, C_k calculates corresponding gradients based on the specific loss function. We take the mean squared error loss function as an example to show the relationship of the perturbed gradients and true gradients, which will help understand why the server can exactly recover the true global model. Specifically, since C_k can only obtain the perturbed prediction $\hat{\boldsymbol{y}}^{(L)}$, C_k can only use $\hat{\boldsymbol{y}}^{(L)}$ to compute the loss value, which is defined as:

$$\widehat{\mathcal{L}}\left(\widehat{W}; (\boldsymbol{x}, \bar{\boldsymbol{y}})\right) = \frac{1}{2} \parallel \hat{\boldsymbol{y}}^{(L)} - \bar{\boldsymbol{y}} \parallel_2^2 . \tag{13}$$

With the above perturbed loss function, C_k can compute the perturbed gradient of the l-th layer, denoted as $\frac{\partial\widehat{\mathcal{L}}(\widehat{W};(\boldsymbol{x},\bar{\boldsymbol{y}}))}{\partial\widehat{W}^{(l)}} \in \mathbf{R}^{n_l \times n_{l-1}}$, based on the calculations in Section 2.1. What's more, the following Theorem 2 shows the important relationship between the perturbed gradients and the true gradients.

Theorem 2. *For any $1 \le l \le L$, the perturbed gradients $\frac{\partial\widehat{\mathcal{L}}\left(\widehat{W};(\boldsymbol{x},\bar{\boldsymbol{y}})\right)}{\partial\widehat{W}^{(l)}} \in \mathbf{R}^{n_l \times n_{l-1}}$ and the true gradients $\frac{\partial\mathcal{L}(W;(\boldsymbol{x},\bar{\boldsymbol{y}}))}{\partial W^{(l)}} \in \mathbf{R}^{n_l \times n_{l-1}}$ satisfy*

$$\frac{\partial\widehat{\mathcal{L}}(\widehat{W}; (\boldsymbol{x}, \bar{\boldsymbol{y}}))}{\partial\widehat{W}^{(l)}} = \frac{1}{R^{(l)}} \circ \frac{\partial\mathcal{L}\left(W; (\boldsymbol{x}, \bar{\boldsymbol{y}})\right)}{\partial W^{(l)}} + \boldsymbol{r}^T\boldsymbol{\sigma}^{(l)} - v\boldsymbol{\beta}^{(l)},$$

where $\frac{1}{R^{(l)}}$, $\boldsymbol{r}$, $\boldsymbol{\sigma}^{(l)}$, v and $\boldsymbol{\beta}^{(l)}$ are denoted as

- *$\frac{1}{R^{(l)}}$ is the $n_l \times n_{l-1}$ matrix whose (i, j)-th entry is $\frac{1}{R_{ij}^{(l)}}$;*
- *$\boldsymbol{r} = \boldsymbol{\gamma} \circ \boldsymbol{r}^{(a)}$ and $v = \boldsymbol{r}^T\boldsymbol{r}$;*
- *$\boldsymbol{\sigma}^{(l)} = \alpha\frac{\partial\hat{\boldsymbol{y}}^{(L)}}{\partial\widehat{W}^{(l)}} + \left(\frac{\partial\hat{\mathcal{L}}(\widehat{W}(\boldsymbol{x},\bar{\boldsymbol{y}}))}{\partial\hat{\boldsymbol{y}}^{(L)}}\right)^T \frac{\partial\alpha}{\partial\widehat{W}^{(l)}}$;*
- *$\boldsymbol{\beta}^{(l)} = \alpha\frac{\partial\alpha}{\partial\widehat{W}^{(l)}}$.*

Note that, $\alpha = \sum_{i=1}^{n_{L-1}} \hat{y}_i^{(L-1)}$, and when $l = L$, α is not a function of $\widehat{W}^{(L)}$, and thus $\frac{\partial\alpha}{\partial\widehat{W}^{(L)}} = \mathbf{0}_{n_L \times n_{L-1}}$, which implies that $\boldsymbol{\sigma}^{(L)} = \alpha\frac{\partial\hat{\boldsymbol{y}}^{(L)}}{\partial\widehat{W}^{(l)}}$ and $\boldsymbol{\beta}^{(L)} = \mathbf{0}_{n_L \times n_{L-1}}$.

Proof. See Appendix A.2. □

[1] The value of m $(1 \le m \le n_L)$ determines the defense of predictions, which will be analyzed in Theorem 5.

From Theorem 2, we can observe that $\boldsymbol{\sigma}^{(l)}$ and $\boldsymbol{\beta}^{(l)}$ can be computed directly by clients and both values decide whether the server can recover the true aggregated model parameters (see Section 4.3 for more details). Hence, in addition to the perturbed gradients, the client also needs to compute two noisy items $\boldsymbol{\sigma}^{(l)}$ and $\boldsymbol{\beta}^{(l)}$.

Hence, for any sample $(\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i) \in \mathcal{D}_k$, C_k computes the perturbed local gradients $\frac{\partial \widehat{\mathcal{L}}(\widehat{W};(\boldsymbol{x}_i,\bar{\boldsymbol{y}}_i))}{\partial \widehat{W}^{(l)}}$ and the corresponding noisy items, represented as $\boldsymbol{\sigma}^{(l)}_{(\boldsymbol{x}_i,\bar{\boldsymbol{y}}_i)}$ and $\boldsymbol{\beta}^{(l)}_{(\boldsymbol{x}_i,\bar{\boldsymbol{y}}_i)}$. Then, C_k computes the average local gradients $\nabla F(\widehat{W}^{(l)}, \mathcal{D}_k)$ and $(\boldsymbol{\sigma}_k^{(l)}, \boldsymbol{\beta}_k^{(l)})$ as: for $l = 1, 2, \dots, L$,

$$\begin{cases} \nabla F(\widehat{W}^{(l)}, \mathcal{D}_k) := \dfrac{1}{|\mathcal{D}_k|} \displaystyle\sum_{(\boldsymbol{x}_i,\bar{\boldsymbol{y}}_i)\in\mathcal{D}_k} \dfrac{\partial \widehat{\mathcal{L}}\left(\widehat{W};(\boldsymbol{x}_i,\bar{\boldsymbol{y}}_i)\right)}{\partial \widehat{W}^{(l)}}; \\ \boldsymbol{\sigma}_k^{(l)} := \dfrac{1}{|\mathcal{D}_k|} \displaystyle\sum_{(\boldsymbol{x}_i,\bar{\boldsymbol{y}}_i)\in\mathcal{D}_k} \boldsymbol{\sigma}^{(l)}_{(\boldsymbol{x}_i,\bar{\boldsymbol{y}}_i)}; \\ \boldsymbol{\beta}_k^{(l)} := \dfrac{1}{|\mathcal{D}_k|} \displaystyle\sum_{(\boldsymbol{x}_i,\bar{\boldsymbol{y}}_i)\in\mathcal{D}_k} \boldsymbol{\beta}^{(l)}_{(\boldsymbol{x}_i,\bar{\boldsymbol{y}}_i)}. \end{cases}$$

Note that $\boldsymbol{\gamma} = (\gamma_1, \gamma_2, \dots, \gamma_{n_L})$ is known for the server and $\gamma_i = \gamma_{I_s}$ for $i \in I_s$ and $s = 1, 2, \dots, m$. (See "Random noises selection"). To recover the term $\boldsymbol{r}^T\boldsymbol{\sigma}^{(l)}$ of the equation in Theorem 2, the client C_k should compute the following items:

$$\widetilde{\boldsymbol{\sigma}}^{(l)}_{k,s} := (\boldsymbol{r}^{(a)}_{|I_s})^T \boldsymbol{\sigma}^{(l)}_{k_{|I_s}}, \text{ for } s = 1, 2, \dots, m,$$

where $\boldsymbol{r}^{(a)}_{|I_s}$ is the restriction of the vector $\boldsymbol{r}^{(a)}$ on I_s and $\boldsymbol{\sigma}^{(l)}_{k_{|I_s}}$ is the sub-matrix of $\boldsymbol{\sigma}^{(l)}_k$ consists of the rows indexed by I_s.

Finally, C_k uploads local gradients $\{\nabla F(\widehat{W}^{(l)}, \mathcal{D}_k)\}_{l=1}^{L}$ and noisy terms $\{\widetilde{\boldsymbol{\sigma}}^{(l)}_{k,1}, \widetilde{\boldsymbol{\sigma}}^{(l)}_{k,2}, \cdots, \widetilde{\boldsymbol{\sigma}}^{(l)}_{k,m}, \widetilde{\boldsymbol{\beta}}^{(l)}_k\}_{l=1}^{L}$ to the server for model update.

4.3 Global Model Update

After receiving $\{\nabla F(\widehat{W}^{(l)}, \mathcal{D}_k)\}_{l=1}^{L}$ and $(m+1)$ noisy terms $\{\widetilde{\boldsymbol{\sigma}}^{(l)}_{k,1}, \cdots, \widetilde{\boldsymbol{\sigma}}^{(l)}_{k,m}, \boldsymbol{\beta}^{(l)}_k\}_{l=1}^{L}$ from all clients, the server updates the current global model. Specifically, in addition to aggregate all received local gradients, the server needs to recover exact aggregated results to ensure training accuracy.

- For $l = 1, 2, \dots, L$, perform the aggregation operation as:

$$\begin{cases} \nabla F(\widehat{W}^{(l)}) = \displaystyle\sum_{k=1}^{K} \dfrac{|\mathcal{D}_k|}{|\mathcal{D}|} \nabla F(\widehat{W}^{(l)}, \mathcal{D}_k), \\ \boldsymbol{\sigma}_s^{(l)} = \displaystyle\sum_{k=1}^{K} \dfrac{|\mathcal{D}_k|}{|\mathcal{D}|} \widetilde{\boldsymbol{\sigma}}^{(l)}_{k,s}, s = 1, 2, \dots, m, \\ \boldsymbol{\beta}^{(l)} = \displaystyle\sum_{k=1}^{K} \dfrac{|\mathcal{D}_k|}{|\mathcal{D}|} \boldsymbol{\beta}^{(l)}_k. \end{cases} \tag{14}$$

- According to Theorem 2, recover the true aggregated gradients with the secret noises $R^{(l)}$, γ_{I_s} and v:

$$\nabla F(W^{(l)}) = R^{(l)} \circ \left(\nabla F(\widehat{W}^{(l)}) - \left(\sum_{s=1}^{m} \gamma_{I_s} \boldsymbol{\sigma}_s^{(l)}\right) + v\boldsymbol{\beta}^{(l)}\right),$$

  where $l = 1, 2, \dots, L$.

- Update the current global model for the next iteration as:

$$W^{(l)} \Leftarrow W^{(l)} - \eta \nabla F(W^{(l)}), \text{ for } l = 1, 2, \dots, L \tag{15}$$

Next, we theoretically demonstrate that the server can obtain the true global model from the perturbed gradients.

Theorem 3. *$\nabla F(W^{(l)})$ is the true aggregated gradients meeting*

$$\nabla F(W^{(l)}) = \sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} \nabla F\left(W^{(l)}, \mathcal{D}_k\right) \tag{16}$$

where $1 \le l \le L$ and $\nabla F\left(W^{(l)}, \mathcal{D}_k\right)$ is the true local gradients denoted in Eq. (4).

Proof. See Appendix A.3. □

According to Theorem 3, Eq. (15) is derived as

$$W^{(l)} \Leftarrow W^{(l)} - \eta \sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} \nabla F\left(W^{(l)}, \mathcal{D}_k\right),$$

which is equal to Eq. (5), i.e., the update without noises perturbation.

5 THEORETICAL ANALYSIS

We theoretically analyze the inference attack defence of our method. Essentially, we should ensure that clients cannot obtain the true information from perturbed global model parameters $\widehat{W}$, the perturbed output $\hat{\boldsymbol{y}}^{(l)}$ and the perturbed gradients $\frac{\partial \widehat{\mathcal{L}}(\widehat{W};(\boldsymbol{x},\bar{\boldsymbol{y}}))}{\partial \widehat{W}^{(l)}}$ of each layer. Therefore, our method should meet the following three privacy requirements: (1) the privacy-preservation of global model parameters, (2) the privacy-preservation of outputs and (3) the privacy-preservation of local gradients.

(1) Privacy-preservation of global model parameters. As introduced in Section 4.1, the server perturbs current global model parameters by randomly selected positive real number before distributing. Thus, we prove that clients cannot obtain true global model parameters W from the received perturbed parameters $\widehat{W}$. Specifically, from the perspective of measure theory, we prove that the set of possible true model parameters such that clients receive the same noisy model has positive measure. The true model parameters corresponds to one point in the real field, which has 0 measure. Therefore the possibility that clients obtain the true model parameters is 0

Theorem 4. *For any given perturbed global model parameters $\widehat{W} = \{\widehat{W}^{(l)}\}_{l=1}^{L}$, the possibility that clients obtain the true global model parameters $W = \{W^{(l)}\}_{l=1}^{L}$ is equal to zero.*

Proof. See Appendix B.1 □

(2) Privacy-preservation of outputs. As shown in [23], curious clients can take the final prediction or intermediate outputs of models as input to launch membership inference attacks. Thus, we prove that curious clients cannot obtain the true output of each layer in forward propagation.

As shown in Theorem 1, clients can only obtain the perturbed outputs $\hat{\boldsymbol{y}} = \{\hat{\boldsymbol{y}}^{(l)}\}_{l=1}^{L}$, which satisfy that

$$\begin{cases} \hat{\boldsymbol{y}}^{(l)} = \boldsymbol{r}^{(l)} \circ \boldsymbol{y}^{(l)}, \text{ when } 1 \leq l \leq L-1, \\ \hat{\boldsymbol{y}}^{(L)} = \boldsymbol{y}^{(L)} + \alpha\boldsymbol{\gamma} \circ \boldsymbol{r}^{(a)} = \boldsymbol{y}^{(L)} + \alpha\boldsymbol{r}, \end{cases}$$

Thus, we show that clients cannot determine the true output $\boldsymbol{y}^{(l)}$ from the perturbed output $\hat{\boldsymbol{y}}^{(l)}$. Specifically, for $1 \leq l \leq L-1$, since the noise vector $\boldsymbol{r}^{(l)} = (r_1^{(l)}, r_2^{(l)}, \ldots, r_{n_l}^{(l)}) \in \mathbf{R}_{>0}^{n_l}$ is randomly chosen by the server, similar to Theorem 4, the client obviously cannot determine the true output $\boldsymbol{y}^{(l)}$ from the perturbed output $\hat{\boldsymbol{y}}^{(l)} = \boldsymbol{r}^{(l)} \circ \boldsymbol{y}^{(l)}$. For the prediction vector $\hat{\boldsymbol{y}}^{(L)} = \boldsymbol{y}^{(L)} + \alpha\boldsymbol{\gamma} \circ \boldsymbol{r}^{(a)}$, the parameter α and $\boldsymbol{r}^{(a)} = (r_1^{(a)}, r_2^{(a)}, \ldots, r_{n_L}^{(a)})$ are known to clients, but $\boldsymbol{\gamma} = (\gamma_1, \gamma_2, \ldots, \gamma_{n_L})$ is chosen by the server randomly and is unknown to clients. Thus, we show that clients have no advantage to guess the true prediction from $\hat{\boldsymbol{y}}^{(L)}$ without knowing $\boldsymbol{\gamma}$. Recall that there exists a partition $\sqcup_{s=1}^{m}\{I_s\}$ of $\{1, 2, \ldots, n_L\}$, such that for any i, j in the same I_s satisfy that $\gamma_i = \gamma_j$. Thus, the privacy of predictions is mainly influenced by the parameter m. Specifically, we give the privacy of predictions in terms of m under our proposed method in the following Theorem.

Theorem 5. *When the number of classes $n_L = 1$, clients cannot obtain any information about the true prediction $y^{(L)}$; When $n_L \geq 2$ and $m = 1$, the probability clients obtain the true predictions $\boldsymbol{y}^{(L)}$ from the perturbed predictions $\hat{\boldsymbol{y}}^{(L)}$ is less than 1; When $n_L \geq 2$ and $2 \leq m \leq n_L$, the probability is less than or equal to $1/m$.*

Proof. See Appendix B.2. □

According to the above proofs, clients cannot obtain the correct prediction $\boldsymbol{y}^{(L)}$ of a given sample feature $\boldsymbol{x}$. Hence, they cannot efficiently launch the membership inference attack to infer if a certain data record was the part of training dataset.

(3) Privacy-preservation of gradients. As demonstrated in [35], local gradients can be used to reconstruct training data. Thus, we need to show that our method can efficiently prevent curious clients from obtaining the true local gradients. Based on Theorem 2, the perturbed gradients $\frac{\partial \widehat{\mathcal{L}}\left(\widehat{W};(\boldsymbol{x},\bar{\boldsymbol{y}})\right)}{\partial \widehat{W}^{(l)}}$ and the true gradients $\frac{\partial \mathcal{L}(W;(\boldsymbol{x},\bar{\boldsymbol{y}}))}{\partial W^{(l)}}$ satisfy the following condition:

$$\frac{\partial \widehat{\mathcal{L}}(\widehat{W};(\boldsymbol{x},\bar{\boldsymbol{y}}))}{\partial \widehat{W}^{(l)}} = \frac{1}{R^{(l)}} \circ \frac{\partial \mathcal{L}\,(W;(\boldsymbol{x},\bar{\boldsymbol{y}}))}{\partial W^{(l)}} + \boldsymbol{r}^{T}\boldsymbol{\sigma}^{(l)} - \upsilon\boldsymbol{\beta}^{(l)},$$

where $\frac{\partial \widehat{\mathcal{L}}(\widehat{W};(\boldsymbol{x},\bar{\boldsymbol{y}}))}{\partial \widehat{W}^{(l)}}$, $\boldsymbol{\sigma}^{(l)}$ and $\boldsymbol{\beta}^{(l)}$ are computed by clients. However, $R^{(l)}$, $\boldsymbol{r}^{T}$ and υ are chosen by the server, which are unknown to the client. Similar to the above proofs, it is obviously that clients cannot obtain the true gradient from perturbed local gradients.

6 PERFORMANCE EVALUATION

In this section, we empirically evaluate our method on real-world datasets in terms of **learning accuracy** and **inference attacks defence** (i.e., reconstruction and membership inference attacks). Meanwhile, we give a comparison with the state-of-the-art schemes to show the advantages of our method.

6.1 Experimental Setup

Our method is implemented based on PyTorch and conducted on the single Tesla M40 GPU. In order to show that our method would not decrease the learning accuracy and can efficiently defense against inference attacks, we adopt the state-of-the-art schemes, i.e., the FedAvg [19], PPDL[26], SPN [7] and DBCL [21], as the baseline. Specifically, PPDL trade-offs privacy and accuracy with differential privacy. SPN adopts gradient perturbation to defend privacy attacks. DBCL perturbs global model parameters with CountSketch matrix. In all experiments, the training epochs and the batch-size of each client are set to be 200 and 32, respectively.

Datasets and Metrics. Experiments are conducted on two privacy-sensitive datasets and one image dataset.

- **UCI Bank Marketing Dataset (UBMD)** [22] aims to predict the possibility of clients for subscribing deposits. It contains 41188 instances of 17 dimensional bank data.
- **Lesion Disease Classification (LDC)** [27] [5] provides 8k training and 2k test skin images for the classification of lesion disease.
- **CIFAR-10** [15] contains 60000 color images divided into 10 classes, and each of class contains 6000 images.

6.2 Learning Accuracy

In this section, we evaluate the training accuracy of our algorithm on both regression and classification tasks, and compare with the non-private approach i.e., FedAvg [19], and three state-of-the-art defence methods, i.e., PPDL[26], SPN [7] and DBCL [21].

6.2.1 *Regression.*

We evaluate the learning accuracy of regression tasks on the UBMD dataset in terms of different layers $L \in \{3, 5, 7\}$ and numbers of clients $K \in \{1, 5, 10\}$. Table 1 shows results of these schemes for the final converged model. From the table, the learning accuracy of our method elegantly aligns with that of the FedAvg [19] under various settings and outperforms the PPDL[26], SPN [7] and DBCL [21]. The main reason is that our method can eliminate the perturbed noises to obtain the true updated parameters, and thus would not reduce the learning accuracy. However, PPDL[26], SPN [7] and DBCL [21] cannot eliminate the perturbed noises, which will inevitably reduce the learning accuracy.

6.2.2 *Classification.*

In this section, we evaluate our method for the classification task with ResNet20, ResNet32 and ResNet56 models on the LDC and CIFAR-10 datasets. All the training methods adopt the cross entropy as the loss function. The accuracy of converged models is shown in Table 2. From the table, we can obtain the learning accuracy of our method elegantly aligns with the FedAvg [19] and has significant advantage over baseline methods.

6.3 Scalability Evaluation

To make the results more convincing, we evaluate the scalability of our defense and conduct experiments over industrial dataset of Meituan Co-Branded Credit Cards. The dataset consists of about 600000 instances and aims to predict whether the user will use the credit card in the next month. We conduct experiments on the training duration (seconds) for each stage for 100 epochs with 5 and 50 clients, respectively, and provide the results in Table 3.

Table 1: MSE comparison of different defenses for Regression Tasks. Lower MSE Means Better Performance.

	FedAvg [19]			**PPDL**			**DBCL**			**SPN**			**Ours**		
Clients	$L = 3$	$L = 5$	$L = 7$	$L = 3$	$L = 5$	$L = 7$	$L = 3$	$L = 5$	$L = 7$	$L = 3$	$L = 5$	$L = 7$	$L = 3$	$L = 5$	$L = 7$
1	**0.059**	**0.059**	**0.058**	0.063	0.062	0.068	0.062	0.063	0.062	0.064	0.060	0.067	0.061	**0.059**	0.060
5	**0.079**	**0.079**	0.086	0.084	0.088	0.085	0.091	0.086	0.085	0.082	0.084	0.088	0.081	0.080	**0.084**
10	**0.097**	**0.100**	0.113	0.115	0.108	0.119	0.124	0.118	0.120	0.126	0.123	0.115	0.100	0.101	**0.102**

Table 2: Accuracy result for classification task on LDC and CIFAR10 dataset.

		FedAvg			**PPDL**			**DBCL**			**SPN**			**Ours**		
	participants	ResNet20	ResNet32	ResNet56	ResNet20	ResNet32	ResNet56	ResNet20	ResNet32	ResNet56	ResNet20	ResNet32	ResNet56	ResNet20	ResNet32	ResNet56
LDC	1	**69.42**	70.96	72.79	66.73	67.21	67.34	67.33	67.56	67.83	68.25	68.63	69.31	69.33	**71.05**	**72.83**
	5	69.27	70.78	71.61	67.41	67.45	67.63	69.23	69.85	70.46	68.47	68.63	68.94	**69.29**	**70.84**	71.55
	10	69.14	**70.64**	71.10	67.21	67.45	67.55	68.93	70.42	70.95	68.62	68.74	69.31	**69.22**	70.62	**71.23**
CIFAR10	1	**92.31**	92.59	93.18	88.75	89.26	89.32	91.56	91.73	92.05	89.03	89.36	89.72	92.24	**92.61**	**93.20**
	5	**92.17**	92.47	**93.09**	88.67	89.16	89.24	91.53	91.48	91.69	88.45	89.63	90.12	92.14	**92.50**	93.05
	10	**91.93**	92.19	92.32	88.41	88.53	88.74	91.25	91.72	92.03	89.59	89.68	89.81	91.89	**92.24**	**92.35**

From the table, we can see that compared to the FedAvg (the most widely-adopted plain-training method), the increased computational costs of our scheme are mainly caused by the local model training in client-side. Since clients perform training in parallel in FL, the training duration won't be severely affected as the number of clients scales up.

Table 3: Scalability evalutation over industrial dataset (S).

5 clients	LocalForward	LocalBackward	ModelUpdate	AUC
FedAvg	123.17	2247.63	2403.42	0.7812
Ours	128.43	2968.10	2895.61	0.7816
50 clients	LocalForward	LocalBackward	ModelUpdate	AUC
FedAvg	124.08	2284.28	2389.91	0.7797
Ours	134.22	3017.56	2944.27	0.7794

6.4 Experiments against Privacy Attacks

In this section, we evaluate the performance of different defenses against state-of-the-art privacy attacks, i.e., reconstruction attacks and membership inference attacks, launched by curious clients.

6.4.1 Defense against membership inference Attacks. In this part, we launch the membership inference attack that attempts to identify whether a data record is used during the training phase, against the FL training approaches. The goal of membership inference attack is to infer if a particular individual data record is included in the training dataset by finding the difference of the model outputs between seen and unseen samples of the training-set. Our method defends against this attack by preserving clients from obtaining the true output of each layer.

Experiment setup: We implement this attack based on [23] to tell if a certain data record was part of training set with a surrogate attack model. The attack model takes the final prediction or intermediate output as input and outputs two classes: "*Member*" or "*Non-member*". Following the settings in [23], we assume the attacker can access a fraction of (i.e., 50%) the training set and some non-member samples. In this case, to balance the training, we select half of each batch to include member instances and the other half non-member instances from the attacker's background knowledge, which prevents the attack model from a bias towards member or non-member instances.

Experiment result: Tab. 4 shows the comparison results about membership inference attacks in our method, FedAvg [19], PPDL[26] , SPN [7] and DBCL [21]. The result shows that our method achieves the best learning accuracy while maintaining the strongest defence ability compared with other methods. Specifically, our method achieves attack success rate of around 50%, which means clients cannot infer any information from model outputs except blind guessing. Note that for membership inference attacks that infer a particular record is either a "member" or a "non-member", the worst attack success rate is 50% (i.e., blind guessing). However, the attack success rate of other defence methods is around 60%, which means that clients have a certain advantage to attack successfully compared with blind guessing.

Table 4: The tracing attack results on LDC and CIFAR-10 dataset. We report the accuracy and attack success rate (ASR).

	Method	**FedAvg**	**PPDL**	**DBCL**	**SPN**	**Ours**
LDC	**Accuracy**	72.79	67.34	67.83	69.31	72.83
	ASR (Last Layer)	70.21	59.53	64.75	62.37	50.24
	ASR (Second to Last)	68.50	59.23	53.44	62.02	49.96
	ASR (Third to Last)	66.45	58.73	52.71	61.50	49.87
	ASR (Last three)	72.33	61.84	64.25	62.41	50.02
CIFAR10	**Accuracy**	93.18	89.32	92.05	89.72	93.20
	ASR (Last Layer)	77.31	61.33	66.44	63.95	50.24
	ASR (Second to Last)	75.17	60.59	52.71	61.72	50.13
	ASR (Third to Last)	76.48	60.83	51.62	61.25	49.83
	ASR (Last three)	77.56	61.72	65.32	63.80	49.93

6.4.2 Defense against Reconstruction Attacks. In this section, we launch reconstruction attacks, which aims to recover original training data as accurately as possible from the exchanged model parameters and the calculated gradients. Our method defends against reconstruction attacks by preventing curious clients from obtaining true global model parameters and updates. On the one hand, recent research, dubbed GMI [34], found that private images from the training set of a certain model can be easily reconstructed with the help of a generative model (i.e. GAN [3]). Thus, we first perform the experiments to show the defence ability of the attack that reconstructs training images of certain category with model outputs. On the other hand, Zhu et. al. [35] presented the reconstruct attack, dubbed DLG, to recover training data from local gradients in FL. Thus, we also perform experiments to show the defence ability of the attack that reconstruct the training data from local gradients.

Experiment setup: We evaluate the defence ability of existing methods against GMI on LDC and CIFAR-10 datasets. Similar to [34], we evenly split the training set based on labels, and use one half as private set and the remainder as public set. We evaluate the defence ability under the most strict setting, i.e., the attacker cannot own any auxiliary knowledge about private images, where he/she will recover the image from scratch. Similar to [34], we adopt the following three metrics: 1) Attack success rate (ASR): the prediction accuracy of the attacked model over the reconstructed images; 2) Feature Distance (Feat Dist): the feature distance between the reconstructed image and the centroid of the target class; and 3) K-Nearest Neighbor Distance (KNN Dist): the l_2 distance of the closest data point from the training-set to the reconstructed image.

Experiment results: For the defence performance of the reconstruction attack against the GMI, we present the comparison results in Tab 5, which shows that our method achieves dominant advantages in all three metrics compared with FedAvg [19], PPDL[26], SPN [7] and DBCL [21]. Specifically, the ASR of our method is around 10% for both LDC and CIFAR-10 datasets, which is almost the optimal defensive ability. Note that the number of categories in both two datasets is 10, and thus the probability of random guessing is 10%. However, the ASR of the non-defensive method FedAvg [19] for the LDC and CIFAR-10 are 60% and 68%, respectively. The ASR of other defence methods (i.e., PPDL[26], SPN [7] and DBCL [21]) are around 50%, which demonstrates a relative poor defence ability. Besides, we also provide some visualization of reconstructed images from CIFAR-10 in Fig. 2, where the reconstructed results of our method leak almost zero information about the training dataset.

For the defence performance of reconstruction attacks against the DLG, we train ResNet20 models in CIFAR-100 on 5 clients with our method and take the PPDL[26] as a comparison. In the PPDL[26], the variance of added Gaussian noises are ranging from 10^{-4} to 10^{-1}. Fig. 3 gives the visualization of reconstructed results on a randomly sampled image from CIFAR-100. From the figure, we can see that our method has the same level of defensive ability as the PPDL with the highest level of noise (i.e., 10^{-1}).

Table 5: Comparison of different defense methods against reconstruction attack.

		FedAvg	PPDL	DBCL	SPN	Ours
LDC	Accuracy	72.79	67.34	67.83	69.31	72.83
	KNN Dist	682.23	1386.48	862.51	804.39	2248.56
	Feat Dist	736.47	1683.99	1052.85	879.13	2203.71
	ASR	60.22	33.70	51.66	55.37	11.49
CIFAR10	Accuracy	93.18	89.32	92.05	89.72	93.20
	KNN Dist	364.71	781.43	662.82	721.66	1866.47
	Feat Dist	405.61	723.90	703.98	768.36	1953.94
	ASR	68.41	49.62	58.34	52.15	10.63

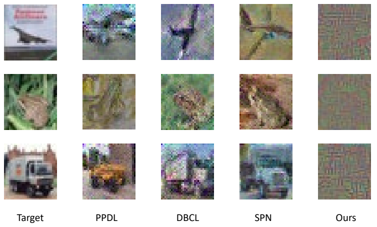

Figure 2: Reconstructed results of GMI.

7 RELATED WORKS

Federated learning (FL) was formally introduced by Google in 2016 [14] to address data privacy in machine learning. However, recent works [23, 35] proved that the original FL schemes still face the risk of privacy leakage. Specifically, two types of inference attacks called reconstruction and membership inference attacks have been widely leveraged to obtain local training data of clients. To defend against these two privacy attacks, some compression methods such as selective gradients sharing [26], reducing dimension [8] and dropout [28] were presented to decrease information disclosure, thereby defending against privacy attacks. However, these methods were proved to be almost ineffective for the defence ability or had a negative impact on the quality of the FL [7]. After that, due to the high efficiency and easy integration, differential privacy technique [6] was widely adopted in FL to improve the defence ability [2, 20, 24, 29]. For example, [2] demonstrated how to maintain data privacy by adding Gaussian noise to shared gradients during the training of deep neural network. [35] also suggested adding Gaussian noise to shared gradients to defend against reconstruction attacks. However, differential privacy-based methods sacrifice model accuracy in exchange for privacy [31], which is not suitable in some FL applications requiring high model accuracy [13, 33]. Following the idea of differential privacy that injects random noises to perturb the shared parameters, the matrix sketching method [4, 18] was considered in FL to perturb the global model parameters. Besides, Fan et. al. [7] leveraged element-wise adaptive gradients perturbations to defend against reconstruction and membership inference attacks.

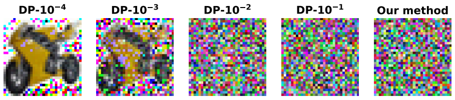

Figure 3: Reconstructed results of DLG against different defenses.

8 CONCLUSION

In this paper, we have presented an efficient model perturbation method in federated learning to defend against the state-of-the-art privacy attacks launched by honest-but-curious clients. Specifically, the server perturbs the global model parameters by adding random selected noises, which prevents clients from obtaining true model parameters including true local gradients. Therefore, it can defend against two famous privacy attacks, i.e., reconstruction and membership inference attacks. Meanwhile, unlike the differential privacy that cannot remove the added noises, the proposed method ensures that the server can remove the added random noises from the aggregated local gradients, and thus would not reduce the learning accuracy. Extensive experiments about both regression and classification tasks demonstrate that the model accuracy of our scheme are almost the same as the plain training, and better than the state-of-the-art privacy defence schemes. Besides, extensive experiments about both reconstruction and membership inference attacks show that the defence ability of our scheme significantly outperforms the state-of-the-art privacy defence schemes.

ACKNOWLEDGMENTS

This work is supported in part by the National Natural Science Foundation of China under Grant 62171248, the R&D Program of Shenzhen under Grant JCYJ20180508152204044, and the PCNL KEY project (PCL2021A07).

REFERENCES

[1] 2021. Federated Learning of Cohorts. http://en.wiki.hancel.org/wiki/Federated_Learning_of_Cohorts.

[2] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*. 308–318.

[3] Martín Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein GAN. *CoRR* (2017).

[4] Kenneth L. Clarkson and David P. Woodruff. 2013. Low rank approximation and regression in input sparsity time. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*. 81–90.

[5] Noel C. F. Codella, Veronica Rotemberg, Philipp Tschandl, M. Emre Celebi, Stephen W. Dusza, David Gutman, Brian Helba, Aadi Kalloo, Konstantinos Liopyris, Michael A. Marchetti, Harald Kittler, and Allan Halpern. 2019. Skin Lesion Analysis Toward Melanoma Detection 2018: A Challenge Hosted by the International Skin Imaging Collaboration (ISIC). *CoRR* abs/1902.03368 (2019).

[6] Cynthia Dwork. 2006. Differential Privacy. In *Automata, Languages and Programming, 33rd International Colloquium*, Vol. 4052. 1–12.

[7] Lixin Fan, KamWoh Ng, Ce Ju, Tianyu Zhang, Chang Liu, Chee Seng Chan, and Qiang Yang. 2020. Rethinking Privacy Preserving Deep Learning: How to Evaluate and Thwart Privacy Attacks. In *Federated Learning - Privacy and Incentive*. 32–50.

[8] Shuhao Fu, Chulin Xie, Bo Li, and Qifeng Chen. 2019. Attack-Resistant Federated Learning with Residual-based Reweighting. *CoRR* abs/1912.11464 (2019).

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

[10] Briland Hitaj, Giuseppe Ateniese, and Fernando Pérez-Cruz. 2017. Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 603–618.

[11] Roger A. Horn and Charles R. Johnson. 2012. *Matrix Analysis, 2nd Ed.* Cambridge University Press.

[12] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. 2017. Densely Connected Convolutional Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. 2261–2269.

[13] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, and et al. 2019. Advances and Open Problems in Federated Learning. *CoRR* abs/1912.04977 (2019).

[14] Jakub Konecný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated Learning: Strategies for Improving Communication Efficiency. *CoRR* abs/1610.05492 (2016).

[15] Alex Krizhevsky and Geoffrey Hinton. 2009. Learning multiple layers of features from tiny images. (2009).

[16] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Process. Mag.* 37, 3 (2020), 50–60.

[17] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2020. On the Convergence of FedAvg on Non-IID Data. In *8th International Conference on Learning Representations*.

[18] Michael W. Mahoney. 2011. Randomized Algorithms for Matrices and Data. *Found. Trends Mach. Learn.* 3, 2 (2011), 123–224.

[19] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, Vol. 54. 1273–1282.

[20] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning Differentially Private Recurrent Language Models. In *6th International Conference on Learning Representations*.

[21] Shusen Wang Mengjiao Zhang. 2020. Matrix Sketching for Secure Collaborative Machine Learning. *ICML*.

[22] Sérgio Moro, Paulo Cortez, and Paulo Rita. 2014. A data-driven approach to predict the success of bank telemarketing. *Decis. Support Syst.* 62 (2014), 22–31.

[23] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. In *2019 IEEE Symposium on Security and Privacy*. 739–753.

[24] Manas A. Pathak, Shantanu Rane, and Bhiksha Raj. 2010. Multiparty Differential Privacy via Aggregation of Locally Trained Classifiers. In *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010*. 1876–1884.

[25] Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural Networks* 61 (2015), 85–117.

[26] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-Preserving Deep Learning. In *ACM SIGSAC*.

[27] Philipp Tschandl, Cliff Rosendahl, and Harald Kittler. 2018. The HAM10000 Dataset: A Large Collection of Multi-Source Dermatoscopic Images of Common Pigmented Skin Lesions. *CoRR* abs/1803.10417 (2018).

[28] Stefan Wager, Sida I. Wang, and Percy Liang. 2013. Dropout Training as Adaptive Regularization. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*. 351–359.

[29] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H. Yang, Farhad Farokhi, Shi Jin, Tony Q. S. Quek, and H. Vincent Poor. 2020. Federated Learning With Differential Privacy: Algorithms and Performance Analysis. *IEEE Trans. Inf. Forensics Secur.* 15 (2020), 3454–3469.

[30] Wikipedia. 2018. Facebook-Cambridge Analytica Data Scandal. https://en.wikipedia.org/wiki/Facebook%E2%80%93Cambridge_Analytica_data_scandal.

[31] Guowen Xu, Hongwei Li, Sen Liu, Kan Yang, and Xiaodong Lin. 2020. VerifyNet: Secure and Verifiable Federated Learning. *IEEE Trans. Inf. Forensics Secur.* 15 (2020), 911–926.

[32] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated Machine Learning: Concept and Applications. *ACM Trans. Intell. Syst. Technol.* 10, 2 (2019), 12:1–12:19.

[33] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. 2020. BatchCrypt: Efficient Homomorphic Encryption for Cross-Silo Federated Learning. In *2020 USENIX Annual Technical Conference*. 493–506.

[34] Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song. 2020. The Secret Revealer: Generative Model-Inversion Attacks Against Deep Neural Networks. In *CVPR*.

[35] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep Leakage from Gradients. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019*. 14747–14756.

A NOTATIONS DESCRIPTION

In this part, we give the description of notations used in this paper in Table 6.

Table 6: Notations used in the proposed scheme

Notation	Description
$\{C_1, C_2, \ldots, C_K\}$	K clients, the k-th client is C_k
$\mathcal{D}_k$	Local training dataset of C_k
$R^{(l)}$ and $R^{(a)}$	Random noises for perturbation
$W = \{W^{(l)}\}_{l=1}^{L}$	True global model parameters
$\widehat{W} = \{\widehat{W}^{(l)}\}_{l=1}^{L}$	Perturbed global model parameters
$\boldsymbol{y}^{(l)}$	True output vector of the l-th layer
$\hat{\boldsymbol{y}}^{(l)}$	Perturbed output vector of the l-th layer
$\frac{\partial \mathcal{L}(W;(\boldsymbol{x},\bar{\boldsymbol{y}}))}{\partial W^{(l)}}$	Gradient matrix of the l-th layer
$\frac{\partial \mathcal{L}\left(\widehat{W};(\boldsymbol{x},\bar{\boldsymbol{y}})\right)}{\partial \widehat{W}^{(l)}}$	Perturbed gradient matrix of the l-th layer
$\nabla F\left(\widehat{W}, \mathcal{D}_k\right)$	Perturbed average local gradients of C_k
$\nabla F\left(\widehat{W}^{(l)}\right)$	Perturbed aggregated gradients of the l-th layer
$\nabla F\left(W^{(l)}\right)$	True aggregated gradients of the l-th layer

B PROOFS FOR SECTION 4

B.1 Proof of Theorem 1

Based on Eq. (8), we can deduce that

$$R^{(1)} = D_{\boldsymbol{r}^{(1)}} E^{(1)}, R^{(L)} D_{\boldsymbol{r}^{(L-1)}} = E^{(L)},$$
$$R^{(l)} D_{\boldsymbol{r}^{(l-1)}} = D_{\boldsymbol{r}^{(l)}} E^{(l)}, \text{ for } 2 \leq l \leq L-1,$$

where $D_{\boldsymbol{r}^{(1)}}$ is the $n_l \times n_l$ diagonal matrix whose main diagonal is $\boldsymbol{r}^{(1)}$ and $E^{(l)}$ is the $n_l \times n_{l-1}$ matrix whose entries are all 1s, for $l = 1, 2, \ldots, L$. Next, we first prove Eq. (11) by mathematical induction. Specifically, when $l = 1$, we can obtain

$$\begin{aligned}
\hat{\boldsymbol{y}}^{(1)} &= ReLU\left(\widehat{W}^{(1)}\boldsymbol{x}\right) = ReLU\left(\left(R^{(1)} \circ W^{(1)}\right)\boldsymbol{x}\right) \\
&= ReLU\left(\left(D_{\boldsymbol{r}^{(1)}} E^{(1)} \circ W^{(1)}\right)\boldsymbol{x}\right) \\
&= ReLU\left(D_{\boldsymbol{r}^{(1)}} \left(E^{(1)} \circ W^{(1)}\right)\boldsymbol{x}\right) \\
&= ReLU\left(D_{\boldsymbol{r}^{(1)}} W^{(1)}\boldsymbol{x}\right) = ReLU\left(\boldsymbol{r}^{(1)} \circ (W^{(1)}\boldsymbol{x})\right) \\
&= \boldsymbol{r}^{(1)} \circ ReLU(W^{(1)}\boldsymbol{x}) = \boldsymbol{r}^{(1)} \circ \boldsymbol{y}^{(1)}.
\end{aligned}$$

Then, for $2 \leq l \leq L-1$, assuming $\hat{\boldsymbol{y}}^{(l-1)} = \boldsymbol{r}^{(l-1)} \circ \boldsymbol{y}^{(l-1)}$ by induction. Then, we have

$$\begin{aligned}
\hat{\boldsymbol{y}}^{(l)} &= ReLU\left(\widehat{W}^{(l)}\hat{\boldsymbol{y}}^{(l-1)}\right) = ReLU\left(\left(R^{(l)} \circ W^{(l)}\right)\left(\boldsymbol{r}^{(l-1)} \circ \boldsymbol{y}^{(l-1)}\right)\right) \\
&= ReLU\left(\left(R^{(l)} \circ W^{(l)}\right) D_{\boldsymbol{r}^{(l-1)}} \boldsymbol{y}^{(l-1)}\right) \\
&= ReLU\left(\left((R^{(l)} D_{\boldsymbol{r}^{(l-1)}}) \circ W^{(l)}\right)\boldsymbol{y}^{(l-1)}\right) \\
&= ReLU\left(\left(D_{\boldsymbol{r}^{(l)}} E^{(l)} \circ W^{(l)}\right)\boldsymbol{y}^{(l-1)}\right) \\
&= ReLU\left(\left(D_{\boldsymbol{r}^{(l)}} (E^{(l)} \circ W^{(l)})\right)\boldsymbol{y}^{(l-1)}\right) \\
&= ReLU\left(D_{\boldsymbol{r}^{(l)}} W^{(l)}\boldsymbol{y}^{(l-1)}\right) = ReLU\left(\boldsymbol{r}^{(l)} \circ (W^{(l)}\boldsymbol{y}^{(l-1)})\right) \\
&= \boldsymbol{r}^{(l)} \circ ReLU\left(W^{(l)}\boldsymbol{y}^{(l-1)}\right) = \boldsymbol{r}^{(l)} \circ \boldsymbol{y}^{(l)}.
\end{aligned}$$

Then, we prove Eq. (12) as follows.

$$\begin{aligned}
\hat{\boldsymbol{y}}^{(L)} &= \widehat{W}^{(L)}\hat{\boldsymbol{y}}^{(L-1)} = (R^{(L)} \circ W^{(L-1)} + R^{(a)})\hat{\boldsymbol{y}}^{(L-1)} \\
&= (R^{(L)} \circ W^{(L-1)})(\boldsymbol{r}^{(L-1)} \circ \boldsymbol{y}^{(L-1)}) + R^{(a)}\hat{\boldsymbol{y}}^{(L-1)} \\
&= (R^{(L)} D_{\boldsymbol{r}^{(L-1)}}) \circ W^{(L)}\boldsymbol{y}^{(L-1)}) + R^{(a)}\hat{\boldsymbol{y}}^{(L-1)} \\
&= W^{(L)}\boldsymbol{y}^{(L-1)} + R^{(a)}\hat{\boldsymbol{y}}^{(L-1)} = \boldsymbol{y}^{(L)} + \alpha\boldsymbol{\gamma} \circ \boldsymbol{r}^{(a)} \\
&= \boldsymbol{y}^{(L)} + \alpha\boldsymbol{r}\,.
\end{aligned}$$

B.2 Proof of Theorem 2

Before giving the proof, we recall some notations about the derivatives of vector-valued functions. Specifically, for any vectors $\boldsymbol{f} \in \mathbf{R}^m$ and $\boldsymbol{x} \in \mathbf{R}^n$, the partial derivative $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}$ of $\boldsymbol{f}$ with respect to $\boldsymbol{x}$ is an $m \times n$ matrix, whose (i, j)-th entry is given as

$$\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)_{ij} = \frac{\partial f_i}{\partial x_j}.$$

Moreover, when $\boldsymbol{x}$ is a $u \times v$ matrix, we can regard $\boldsymbol{x}$ as a vector of $\mathbf{R}^{uv}$, then $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}$ is also well-defined.

According to Theorem 1, the prediction is computed as $\hat{\boldsymbol{y}}^{(L)} = \boldsymbol{y}^{(L)} + \alpha\boldsymbol{r}$, and thus the perturbed loss function shown in Eq. (13) is

$$\widehat{\mathcal{L}}\left(\widehat{W}; (\boldsymbol{x}, \bar{\boldsymbol{y}})\right) = \frac{1}{2} \parallel \boldsymbol{y}^{(L)} + \alpha\boldsymbol{r} - \bar{\boldsymbol{y}} \parallel_2^2 .$$

Then, the derivative of the loss with respect to the prediction is

$$\frac{\partial \widehat{\mathcal{L}}\left(\widehat{W}; (\boldsymbol{x}, \bar{\boldsymbol{y}})\right)}{\partial \hat{\boldsymbol{y}}^{(L)}} = \left(\hat{\boldsymbol{y}}^{(L)} - \bar{\boldsymbol{y}}\right)^T = \frac{\partial \mathcal{L}\left(W; (\boldsymbol{x}, \bar{\boldsymbol{y}})\right)}{\partial \boldsymbol{y}^{(L)}} + \alpha\boldsymbol{r}^T,$$

By the chain rule, we can derive that

$$\begin{aligned}
\frac{\partial \widehat{\mathcal{L}}(\widehat{W}; (\boldsymbol{x}, \bar{\boldsymbol{y}}))}{\partial \widehat{W}^{(l)}} &= \frac{\partial \widehat{\mathcal{L}}(\widehat{W}; (\boldsymbol{x}, \bar{\boldsymbol{y}}))}{\partial \hat{\boldsymbol{y}}^{(L)}} \frac{\partial \hat{\boldsymbol{y}}^{(L)}}{\partial \widehat{W}^{(l)}} \\
&= \frac{\partial \mathcal{L}(W; (\boldsymbol{x}, \bar{\boldsymbol{y}}))}{\partial \boldsymbol{y}^{(L)}} \frac{\partial \hat{\boldsymbol{y}}^{(L)}}{\partial \widehat{W}^{(l)}} + \alpha\boldsymbol{r}^T \frac{\partial \hat{\boldsymbol{y}}^{(L)}}{\partial \widehat{W}^{(l)}} \\
&= \frac{\partial \mathcal{L}(W; (\boldsymbol{x}, \bar{\boldsymbol{y}}))}{\partial \boldsymbol{y}^{(L)}} \left(\frac{\partial \boldsymbol{y}^{(L)}}{\partial \widehat{W}^{(l)}} + \frac{\partial (\alpha\boldsymbol{r})}{\partial \widehat{W}^{(l)}}\right) + \alpha\boldsymbol{r}^T \frac{\partial \hat{\boldsymbol{y}}^{(L)}}{\partial \widehat{W}^{(l)}} \\
&= \frac{\partial \mathcal{L}(W; (\boldsymbol{x}, \bar{\boldsymbol{y}}))}{\partial \widehat{W}^{(l)}} + \alpha\boldsymbol{r}^T \frac{\partial \hat{\boldsymbol{y}}^{(L)}}{\partial \widehat{W}^{(l)}} + \left(\frac{\partial \widehat{\mathcal{L}}(\widehat{W}; (\boldsymbol{x}, \bar{\boldsymbol{y}}))}{\partial \hat{\boldsymbol{y}}^{(L)}} - \alpha\boldsymbol{r}^T\right) \frac{\partial (\alpha\boldsymbol{r})}{\partial \widehat{W}^{(l)}} \\
&= \frac{\partial \mathcal{L}(W; (\boldsymbol{x}, \bar{\boldsymbol{y}}))}{\partial \widehat{W}^{(l)}} - v\alpha \frac{\partial \alpha}{\partial \widehat{W}^{(l)}} + \boldsymbol{r}^T \left(\alpha \frac{\partial \hat{\boldsymbol{y}}^{(L)}}{\partial \widehat{W}^{(l)}} + \left(\frac{\partial \widehat{\mathcal{L}}(\widehat{W}; (\boldsymbol{x}, \bar{\boldsymbol{y}}))}{\partial \hat{\boldsymbol{y}}^{(L)}}\right)^T \frac{\partial \alpha}{\partial \widehat{W}^{(l)}}\right) \\
&= \frac{1}{R^{(l)}} \circ \frac{\partial \mathcal{L}(W; (\boldsymbol{x}, \bar{\boldsymbol{y}}))}{\partial W^{(l)}} + \boldsymbol{r}^T\boldsymbol{\sigma}^{(l)} - v\boldsymbol{\beta}^{(l)}.
\end{aligned}$$

B.3 Proof of Theorem 3

According to Theorem 2, we can derive that

$$\begin{aligned}
\nabla F(\widehat{W}^{(l)}, \mathcal{D}_k) &= \frac{1}{|\mathcal{D}_k|} \sum_{(\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i) \in \mathcal{D}_k} \frac{\partial \widehat{\mathcal{L}}\left(\widehat{W}; (\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i)\right)}{\partial \widehat{W}^{(l)}} \\
&= \frac{1}{|\mathcal{D}_k|} \sum_{(\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i) \in \mathcal{D}_k} \left(\frac{1}{R^{(l)}} \circ \frac{\partial \mathcal{L}\left(W; (\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i)\right)}{\partial W^{(l)}} \right. \\
&\quad \left. + \boldsymbol{r}^T\boldsymbol{\sigma}_{(\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i)}^{(l)} - v\boldsymbol{\beta}_{(\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i)}^{(l)}\right) \\
&= \frac{1}{R^{(l)}} \circ \nabla F(W^{(l)}, \mathcal{D}_k) + \boldsymbol{r}^T\boldsymbol{\sigma}_k^{(l)} - v\boldsymbol{\beta}_k^{(l)}.
\end{aligned}$$

Thus, we can obtain that

$$\begin{aligned}
\nabla F(W^{(l)}) &= R^{(l)} \circ \left(\nabla F(\widehat{W}^{(l)}) - \left(\sum_{s=1}^{m} \boldsymbol{\gamma}_{I_s} \boldsymbol{\sigma}_s^{(l)}\right) + \upsilon \boldsymbol{\beta}^{(l)}\right) \\
&\overset{(a)}{=} R^{(l)} \circ \sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} \left(\nabla F(\widehat{W}^{(l)}, \mathcal{D}_k) - \boldsymbol{r}^T \boldsymbol{\sigma}_k^{(l)} + \upsilon \boldsymbol{\beta}_k^{(l)}\right) \\
&= R^{(l)} \circ \sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} \left(\frac{1}{R^{(l)}} \circ \nabla F(W^{(l)}, \mathcal{D}_k) + \boldsymbol{r}^T \boldsymbol{\sigma}_k^{(l)}\right. \\
&\quad \left. -\upsilon \boldsymbol{\beta}_k^{(l)} - \boldsymbol{r}^T \boldsymbol{\sigma}_k^{(l)} + \upsilon \boldsymbol{\beta}_k^{(l)}\right) \\
&= R^{(l)} \circ \frac{1}{R^{(l)}} \circ \sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} \nabla F(W^{(l)}, \mathcal{D}_k) \\
&= \sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} \nabla F(W^{(l)}, \mathcal{D}_k).
\end{aligned}$$

where the equation (a) satisfies the following deduction

$$\begin{aligned}
\sum_{s=1}^{m} \boldsymbol{\gamma}_{I_s} \widetilde{\boldsymbol{\sigma}}_s^{(l)} &= \sum_{s=1}^{m} \boldsymbol{\gamma}_{I_s} \left(\sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} \widetilde{\boldsymbol{\sigma}}_{k,s}^{(l)}\right) \\
&= \sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} \boldsymbol{r}^T \widehat{\boldsymbol{\sigma}}_k^{(l)} = \sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} \boldsymbol{r}^T \boldsymbol{\sigma}_k^{(l)}.
\end{aligned}$$

C PROOFS FOR SECTION 5

C.1 Proof of Theorem 4

As shown in Section 4.1, the noises $R^{(l)} \in \mathbf{R}^{n_l \times n_{l-1}}$ and $R^{(a)} \in \mathbf{R}^{n_L \times n_{L-1}}$ used for perturbing are given as:

$$\begin{aligned}
R_{ij}^{(l)} &= \begin{cases} r_i^{(1)}, & \text{when } l = 1 \\ r_i^{(l)}/r_j^{(l-1)}, & \text{when } 2 \leq l \leq L-1 \\ 1/r_j^{(L-1)}, & \text{when } l = L \end{cases} \\
R_{ij}^{(a)} &= \gamma_i \cdot r_i^{(a)}.
\end{aligned}$$

where the vectors $\boldsymbol{r}^{(l)} = (r_1^{(l)}, r_2^{(l)}, \ldots, r_{n_l}^{(l)}) \in \mathbf{R}_{>0}^{n_l}$ for $l = 1, 2, \ldots, L-1$, and $\boldsymbol{r}^{(a)} = (r_1^{(a)}, r_2^{(a)}, \ldots, r_{n_L}^{(a)}) \in \mathbf{R}^{n_L}$ with pairwise different components are randomly selected and kept secret by the server. Next, we show that for arbitrary noises $\{\boldsymbol{r}^{(l)}\}_{l=1}^{L-1}$ and $\boldsymbol{r}^{(a)}$, it can find $W = \{W^{(l)}\}_{l=1}^{L}$ to construct a given $\widehat{W} = \{\widehat{W}^{(l)}\}_{l=1}^{L}$. Specifically, for any given perturbed global model parameters $\widehat{W} = \{\widehat{W}^{(l)}\}_{l=1}^{L}$, based on the above equations, we can construct the model parameters $W = \{W^{(l)}\}_{l=1}^{L}$ as follows:

$$\begin{cases} W^{(l)} = \dfrac{1}{R^{(l)}} \circ \widehat{W}^{(l)}, \text{ for } l = 1, 2, \ldots, L-1, \\ W^{(L)} = \dfrac{1}{R^{(L)}} \circ (\widehat{W}^{(L)} - R^{(a)}), \end{cases} \tag{17}$$

where for $l = 1, 2, \ldots, L$, $\frac{1}{R^{(l)}} \in \mathbf{R}^{n_l \times n_{l-1}}$ is defined as: the (i, j)-th entry of $\frac{1}{R^{(l)}}$ is $\frac{1}{R^{(l)}}_{ij} = (R_{ij}^{(l)})^{-1}$.

Then, we can show that the noises $(\{R^{(l)}\}_{l=1}^{L}, R^{(a)})$ and the model parameters $W = \{W^{(l)}\}_{l=1}^{L}$ constructed by Eq. (17) satisfy Eq. (7) as follows:

$$R^{(l)} \circ W^{(l)} = R^{(l)} \circ \frac{1}{R^{(l)}} \circ \widehat{W}^{(l)} = \widehat{W}^{(l)}, \text{ for } 1 \leq l \leq L-1,$$

and

$$\begin{aligned}
R^{(L)} \circ W^{(L)} + R^{(a)} &= R^{(L)} \circ \frac{1}{R^{(L)}} \circ (\widehat{W}^{(L)} - R^{(a)}) + R^{(a)} \\
&= \widehat{W}^{(L)} - R^{(a)} + R^{(a)} = \widehat{W}^{(L)}.
\end{aligned}$$

Note that $\{R^{(l)}\}_{l=1}^{L}$ and $R^{(a)}$ are determined by the vectors $\{\boldsymbol{r}^{(l)}\}_{l=1}^{L-1}$ and $\boldsymbol{r}^{(a)}$. Due to the arbitrariness of $\{\boldsymbol{r}^{(l)}\}_{l=1}^{L-1}$ and $\boldsymbol{r}^{(a)}$, the set $\{W = \{W^{(l)}\}_{l=1}^{L}$: the corresponding perturbed global model parameters are the given $\widehat{W} = \{\widehat{W}^{(l)}\}_{l=1}^{L}$ $\}$ is a positive measure set. However, $\{W = \{W^{(l)}\}_{l=1}^{L}$: $W = \{W^{(l)}\}_{l=1}^{L}$ are exactly the true global model parameters$\}$ is a zero measure set. Therefore, the possibility that the clients obtain the ture global model parameters is equal to 0.

C.2 Proof for Theorem 5

When $n_L = 1$, then the prediction is one dimensional, denoted as $\hat{y}^{(L)} = y^{(L)} + \alpha\gamma r^{(a)}$, which usually represents regression tasks. Since γ is chosen randomly by the server, clients cannot know the true prediction $y^{(L)}$.

When $n_L \geq 2$, then the prediction is multi-dimensional, denoted as $\hat{\boldsymbol{y}}^{(L)} = \boldsymbol{y}^{(L)} + \alpha\boldsymbol{\gamma} \circ \boldsymbol{r}^{(a)}$. If $m = 1$, then $\boldsymbol{\gamma}$ satisfies $\gamma_1 = \gamma_2 = \cdots = \gamma_{n_L} = \gamma$. Thus, $\hat{y}_i^{(L)} = y_i^{(L)} + \alpha\gamma r_i^{(a)}$ for $i = 1, 2, \ldots, n_L$. Note that $r_1^{(a)}, r_2^{(a)}, \ldots, r_{n_L}^{(a)}$ are pairwise distinct and γ is randomly selected by the server, thus it cannot determine the largest one among $y_1^{(L)}, y_2^{(L)}, \ldots, y_{n_L}^{(L)}$. Hence the probability that clients obtain the true prediction is obviously less than 1.

If $2 \leq m \leq n_L$, then $\boldsymbol{\gamma}$ satisfies $\gamma_i = \gamma_{I_s}$ for $1 \leq s \leq m$ and $i \in I_s$. Let $s' \in I_s$ such that $y_{s'}^{(L)} = \max_{i \in I_s}\{y_i^{(L)}\}$, then $\max_{1 \leq i \leq n_L}\{y_i^{(L)}\} = \max\{y_{1'}^{(L)}, y_{2'}^{(L)}, \ldots, y_{m'}^{(L)}\}$. Hence the probability that clients obtain the true prediction is less than or equal to the probability that clients obtain the maximal one among $y_{1'}^{(L)}, y_{2'}^{(L)}, \ldots, y_{m'}^{(L)}$. Concretely, the noisy prediction $\{\hat{y}_{i'}^{(L)}\}_{i=1}^{m}$ and the true prediction $\{y_{1'}^{(L)}\}_{i=1}^{m}$ satisfy the following m equations:

$$\begin{cases} \hat{y}_{1'}^{(L)} = y_{1'}^{(L)} + \alpha\gamma_{I_1} r_{1'}^{(a)}, \\ \hat{y}_{2'}^{(L)} = y_{2'}^{(L)} + \alpha\gamma_{I_2} r_{2'}^{(a)}, \\ \qquad \vdots \\ \hat{y}_{m'}^{(L)} = y_{m'}^{(L)} + \alpha\gamma_{I_m} r_{m'}^{(a)}. \end{cases} \tag{18}$$

Note that $(\hat{y}_{1'}^{(L)}, \hat{y}_{2'}^{(L)}, \ldots, \hat{y}_{m'}^{(L)})$ and α are known to the clients, and $\gamma_{I_1}, \gamma_{I_2}, \ldots, \gamma_{I_m}$ are independent randomly chosen by the server. Thus for any m-tuple $(y_{1'}^{(L)}, y_{2'}^{(L)}, \ldots, y_{m'}^{(L)}) \in \mathbf{R}^m$, there always exists an m-tuple $(\gamma_{I_1}, \gamma_{I_2}, \ldots, \gamma_{I_m})$ satisfying Eq. (18). Hence the client cannot obtain any information about the true prediction vector $(y_{1'}^{(L)}, y_{2'}^{(L)}, \ldots, y_{m'}^{(L)})$. So the probability that the clients obtain the maximal one among $y_{1'}^{(L)}, y_{2'}^{(L)}, \ldots, y_{m'}^{(L)}$ is equal to $1/m$. Therefore, the probability that clients obtain the true predictions $\boldsymbol{y}^{(L)}$ from the perturbed predictions $\hat{\boldsymbol{y}}^{(L)}$ is less than or equal to $1/m$.