On Data Distribution Leakage in Cross-Silo Federated Learning

Yangfan Jiang, Xinjian Luo, Yuncheng Wu, Xiaochen Zhu, Xiaokui Xiao, *Fellow, IEEE*, and Beng Chin Ooi, *Fellow, IEEE*

***Abstract*—Federated learning (FL) has emerged as a promising privacy-preserving machine learning paradigm, enabling data owners to collaboratively train a joint model by sharing model parameters instead of private training data. However, recent studies reveal the privacy risks in FL by inferring private training data from model parameters. Therefore, differential privacy (DP) is incorporated into FL to safeguard training data. Nevertheless, DP does not provide a strong theoretical guarantee for protecting data distribution, which is also highly sensitive in the *cross-silo* FL scenarios as it may reflect the business secrets of data owners. In this article, we develop two attack methods to investigate the potential risks of data distribution leakage in differentially private cross-silo FL. We highlight that an honest-but-curious server can successfully infer both the feature and label distributions of each party's training data without any background knowledge. Specifically, the first attack applies when models are differentiable, while the second attack caters to non-differentiable classification models. Extensive experiments on six benchmark datasets validate the effectiveness of the proposed attacks. The results demonstrate that the state-of-the-art DP-SGD algorithm is still vulnerable to the inference attack on data distribution, emphasizing the necessity of designing more advanced privacy-preserving FL frameworks.**

***Index Terms*—Federated learning, data distribution leakage, differential privacy.**

I. Introduction

FEDERATED learning (FL) [1], [2], [3], [4], [5] has recently emerged as a promising privacy-preserving machine learning (ML) paradigm for collaborative analysis. Its basic idea is to let multiple distributed data owners (i.e., parties) conduct model training on their *local* data, while iteratively sharing their model parameters with a server to construct a *global* model that provides improved accuracy than the local models. Compared to *centralized* machine learning (which requires all parties to share their data with a trusted party for model training), FL provides better data privacy protection, since each party's local data is not directly revealed to any other parties.

Manuscript received 11 August 2023; revised 26 November 2023; accepted 23 December 2023. Date of publication 3 January 2024; date of current version 10 June 2024. This work was supported by Singapore Ministry of Education Academic Research Fund Tier 3 under MOEs official under Grant MOE2017-T3-1-007. Recommended for acceptance by Y. Tong. *(Corresponding author: Xiaokui Xiao.)*

The authors are with the School of Computing, National University of Singapore, Singapore 117417 (e-mail: jyangfan@comp.nus.edu.sg; xinjluo@comp.nus.edu.sg; wuyc@comp.nus.edu.sg; xczhu@nus.edu.sg; xkxiao@nus.edu.sg; ooibc@comp.nus.edu.sg).

Digital Object Identifier 10.1109/TKDE.2023.3349323

Nevertheless, existing studies [6], [7], [8], [9], [10], [11], [12], [13], [14] demonstrate that although each party's local data is not directly exposed in FL, an adversary can still recover them from the shared model parameters, because those parameters unintentionally memorize sensitive information of the training datasets [13], [15], [16]. To address this issue, recent works [17], [18], [19], [20], [21], [22] have proposed to incorporate *differential privacy (DP)* [23], [24], [25], [26], [27] into the local training process of each party. The key idea is to let each party perturb the model parameters shared, to make it more difficult for an adversary to derive private information. In particular, DP guarantees that even if an adversary observes all of the perturbed model parameters shared by a party, he is unable to infer whether any specific record exists in the training data of the party. As such, DP prevents the inference of the detailed information in any specific training record.

Motivation: Although DP ensures the protection of individual training records, it does not provide a strong theoretical guarantee in terms of the protection of *aggregate statistics* from each party's data. An alternative solution is *group DP*, where neighboring datasets are defined as differing by multiple records (e.g., s records) rather than just one. As s increases, group DP theoretically offers privacy protection in terms of data distribution. However, as we shall discuss in Section II-B, both DP and group DP approach result in either a meaningless theoretical privacy guarantee or unacceptable model utility. This is of particular concern in the scenario of *cross-silo* FL, where (i) there is a relatively small number of parties participating in the FL task, but (ii) each party has a sizable amount of local data for model training, since the distribution information may reflect business secrets of each party [28].

For example, suppose that multiple commercial banks jointly perform an FL task to construct a model on their customer data. In this case, it is important that the FL method does not leak the data distribution information about any bank's customers, such as age distribution and income distribution; otherwise, such distributions might be utilized by competing banks to their advantage. Furthermore, an adversary can exploit these data distributions to boost the accuracy of individual-level privacy attacks [29], [30]. Hence, the distribution information should be deemed confidential for FL parties.

That said, one may argue that this type of information leakage might be a mere theoretical possibility, not a practical concern, since (i) DP is based on a very pessimistic assumption of the adversary, and (ii) the degree of privacy protection offered

by DP in practice could be much higher than its theoretical guarantee suggests. Recent studies have demonstrated that even when the theoretical DP guarantee is rather weak, DP can still effectively prevent an informed adversary[1] from extracting individual training record from model parameters [31], [32]. This leads to the following question: in a practical FL setting, is it possible for an adversary to infer each party's data distribution when the information shared by each party is safeguarded by DP?

Contributions: In this article, we answer the aforementioned question positively, by developing two practical attacks that enable an honest-but-curious server in FL to infer both the feature and label distributions of each party's private data. Specifically, the first attack is designed for the case when the model is differentiable, whereas the second attack is generalized to non-differentiable models at a moderate cost of attack accuracy. Our attacks do not require the FL server to have any prior knowledge of each party's data; instead, the only information utilized in our attacks is the noisy local model parameters shared by each party during the FL process. This non-reliance on prior knowledge is particularly important in practice, as it is usually difficult for an attacker to obtain much information about each party's local data in advance [33]. Moreover, our attacks enable the adversary to infer fine-grained distribution information about FL parties' local datasets, i.e., the shape of the feature distribution and the proportion of each label, revealing a more serious privacy risk than existing privacy attacks in FL [6], [7], [13], [34].

The key insight that we exploit in our attacks is as follows. First, as each party trains its local model on its local data $\mathcal{D}_k$, the local model tends to be accurate when the test data follows a similar distribution as $\mathcal{D}_k$ does, but inaccurate otherwise. Meanwhile, since the global model is constructed by incorporating the information shared by all parties, it tends to offer accurate results for test data following any party's local data distribution. As the server has access to both the global model and each party's local model, it can examine the difference between the models to infer each party's local data distribution. In particular, the server may generate a large number of synthetic records, and feed them to both the global model $\mathcal{M}_G$ and the local model $\mathcal{M}_k$ of the k-th party. For any synthetic record $\boldsymbol{x}$, if $\mathcal{M}_G$ and $\mathcal{M}_k$ provide very different outputs when given $\boldsymbol{x}$ as input, then it is likely that $\boldsymbol{x}$ differs significantly from all records in the k-th party's local data; otherwise, $\mathcal{M}_G$ and $\mathcal{M}_k$ should have similar outputs, since $\mathcal{M}_G$ generalizes $\mathcal{M}_k$. As such, we can eliminate the part of the synthetic data that are unlikely to follow the distribution of the k-th party's local data, and then use the remaining synthetic data to estimate the distributions of features and labels of the data owned by the k-th party. The efficiency of this inference attack, however, depends on the quality of the synthetic data. In particular, if the synthetic records are sampled uniformly at random, then most of them may differ considerably from the k-th party's local data, and hence, would not be useful for the inference attack. To address this issue, we develop synthetic data generation methods that optimize the efficiency of our inference attack. For the case when the trained model is differentiable, we adopt a constraint optimization method, *projected gradient descent* [35], to generate synthetic records. For the case of non-differentiable models, we design a *hill climbing*-based [36] method for numerical optimization to identify suitable synthetic records.

In summary, we make the following contributions. First, we present the first formal study on distribution inference attacks in cross-silo FL, and demonstrate the feasibility of this type of attack even when each party enforces DP on the model parameters that they share with the server. This provides an important insight for the future development of cross-silo FL solutions: if we are to safeguard each party's data distribution information, we should not rely solely on DP, but should consider additional mechanisms of data privacy protection.

Second, we propose two attack methods for inferring both the feature and label distributions of each party's training data in FL. Our two methods target differentiable and non-differentiable models, respectively. Notably, our attacks enable the adversary to infer *fine-grained* distribution information, i.e., the shape of the feature distribution and the proportions of labels. In addition, the adversary *do not require any prior knowledge* of any party's training data. This makes our attacks more practical and reveals a more severe privacy issue compared to other data privacy attacks in FL [6], [7], [13], [34] that assume the availability of auxiliary data that is somewhat similar to the target party's private data and can only infer binary properties of features or the presence of labels.

Third, we evaluate our attacks on two machine learning models trained with DP-based FL algorithms on six benchmark datasets. The experimental results demonstrate the effectiveness of the proposed attacks. Based on our findings, we discuss several potential defenses and highlight the need to develop more advanced privacy-preserving FL frameworks.

Roadmap: The rest of the article is organized as follows. Section II introduces the background knowledge of FL and DP. Section III formalizes the problem of data distribution leakage in cross-silo FL. Section IV elaborates our attack methodologies. Section V contains experimental evaluations. Section VI discusses several potential defenses. Finally, we review the related work in Section VII and conclude the article in Section VIII.

II. Preliminaries

A. Federated Learning

Let $D = \{(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}_{i=1}^{n}$ be a training dataset where $\mathcal{X}$ and $\mathcal{Y}$ denote the feature space and label space, respectively. We assume that D is distributed among N parties, each of which has a disjoint subset D_k ($k \in \{1, \ldots, N\}$) of the tuples in D. The objective of federated learning (FL) is to let multiple parties collaboratively build a global model by only sharing their local model parameters with a server. Federated averaging (FedAvg) [1] is the first and one of the most fundamental FL algorithms [3], [37]. In FedAvg algorithm, each party first trains a local model based on their private training data and then submits the local model parameters to a server for global model aggregation. Formally, the local model is represented by a model

[1] A powerful adversary who knows all training records except one.

function $f(\cdot;\boldsymbol{\theta}) : \mathcal{X} \mapsto \mathcal{Y}$ that is parameterized by $\boldsymbol{\theta}$. Let $\boldsymbol{\theta}_k^{(t)}$ denote the local model parameters submitted by the k-th party in the t-th iteration of FL, then we have

$$\boldsymbol{\theta}_k^{(t)} = \arg\min_{\boldsymbol{\theta}} \frac{1}{n_k} \sum_{i=1}^{n_k} \ell\left(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i\right) + R(\boldsymbol{\theta}), \tag{1}$$

where $\ell(\cdot,\cdot)$ is a loss function that measures distance between $f(\boldsymbol{x}_i;\boldsymbol{\theta})$ and the ground truth y_i, $R(\cdot)$ is a regularization term that avoids overfitting, and n_k is the number of training records of party k. Each FL party submits the local model parameters to the server after completing the local training. After collecting all local model parameters, the FL server aggregates these local models into a global model by weighted average as follows:

$$\boldsymbol{\theta}_G^{(t)} \leftarrow \sum_{k=1}^{N} \frac{n_k}{n} \boldsymbol{\theta}_k^{(t)}, \tag{2}$$

where $n = \sum_{k=1}^{N} n_k$ is the total number of training samples for all parties. The global model is then sent back to all parties for the next round of training. In the end, the final round global model is obtained and shared with all parties. In the prediction phase, the model takes an input sample $\boldsymbol{x}$ and outputs a vector. We call the elements in the output vector confidence scores, which indicate the probabilities that $\boldsymbol{x}$ belongs to different classes. The class with the highest confidence score will be selected as the predicted result. Note that although we consider the standard FedAvg algorithm in this article, our attacks can be easily adapted to other popular FL algorithms, such as robust FL algorithms [38], as long as the parties need to share their local parameters for training a federated model.

According to the application scenarios, FL can be further categorized into *cross-silo* FL and *cross-device* FL [3]. In the cross-silo setting, the global model is collaboratively constructed by a small number of data owners with large amounts of training data and strong computing power (e.g., banks or hospitals). In contrast, the number of data owners in the cross-device setting is relatively large, but the computing power and the size of training data of each data owner are commonly limited (e.g., smartphones or IoT devices). In this article, we mainly focus on the cross-silo setting, as the data distribution information is often deemed confidential (e.g., business secrets [28]) in this setting.

Note that data distribution information can be safeguarded using party-level DP (also known as local DP), which is designed to obscure the entire dataset of one party. However, as we shall discuss in Section VII, this technique is primarily suited for cross-device FL but not for cross-silo FL, because its application in the cross-silo settings may significantly impair the model utility.

B. Differential Privacy

Differential privacy (DP) [23], [24], [25], [26] has become the *de facto* standard privacy notion for many privacy-preserving data analysis tasks, and has been incorporated into FL in recent years [17], [18], [19], [20], [21], [22]. In the context of machine learning, DP aims to make the model parameters trained on neighboring datasets with the difference of only one record indistinguishable. In other words, DP ensures that the adversary can hardly infer whether any specific record appears in the training dataset, hence protecting the privacy of individual records. Formally, differential privacy is defined as follows:

Definition 1 (Differential Privacy [24]): A randomized algorithm $\mathcal{A}$ satisfies (ϵ, δ)-differential privacy, if for any two neighboring datasets D, D' differing in only one record and for any set of outputs $\mathcal{O} \subseteq Range(\mathcal{A})$ it holds that

$$\Pr\left[\mathcal{A}(D) \in \mathcal{O}\right] \leq e^{\epsilon} \cdot \Pr\left[\mathcal{A}(D') \in \mathcal{O}\right] + \delta.$$

The privacy parameters ϵ and δ control the privacy and utility tradeoff. Roughly speaking, (ϵ, δ)-DP ensures that the densities of $\mathcal{A}(D)$ and $\mathcal{A}(D')$ are indistinguishable (measured by ϵ and δ). Therefore, smaller ϵ and δ indicate stronger privacy guarantees (but may cause lower model utility). The basic idea to design DP-based FL is to add noises to gradients during the local training process to perturb local model parameters. A smaller privacy parameter (i.e., ϵ or δ) implies a larger DP noise is required.

However, the privacy guarantee of DP for aggregate statistics (e.g., data distribution) is rather weak. Suppose we apply an (ϵ, δ)-DP algorithm $\mathcal{A}$ on two datasets D and D'' with different data distributions. Let s be the number of distinct records between D and D'', then s should be relatively large to make their data distributions distinguishable. By the analysis of group differential privacy [39], we have

$$\Pr\left[\mathcal{A}(D) \in \mathcal{O}\right] \leq e^{s\epsilon} \cdot \Pr\left[\mathcal{A}(D'') \in \mathcal{O}\right] + se^{(s-1)\epsilon}\delta.$$

Recall that a smaller ϵ and δ imply that a larger DP noise is required to perturb the model parameters. Thus, while one might set ϵ extremely small (e.g., inversely proportional to s) to protect data distribution information, this necessitates injecting a substantial amount of noise into the model parameters, which would lead to unacceptable model utility. Consequently, the above inequality indicates that the difference between the distributions of $\mathcal{A}(D)$ and $\mathcal{A}(D'')$ can be significant, as long as the model utility is practical. Moreover, group DP's privacy protection – in terms of data distribution information – is excessive; it conceals all details about a group of records, including attribute correlations. Therefore, both DP and group DP might fail to provide a viable privacy-utility trade-off for protecting data distribution, resulting in either a meaningless theoretical privacy guarantee or impractical model utility.

III. Problem Statement

We consider the FL setting shown in Fig. 1, where there are $N (N \geq 2)$ parties and one server that collaboratively train an ML model in a number of iterations. In the first iteration, each party trains a differentially private model on its local data, and submits the parameter of the model to the server. The server then aggregates these local model parameters to construct a global model, and broadcasts the global model parameters to all parties. This process is repeated in each of the subsequent iterations, with one change: each party does not train its local model from scratch, but obtains a new local model by refining the global model parameters using its local data in a differentially private manner.

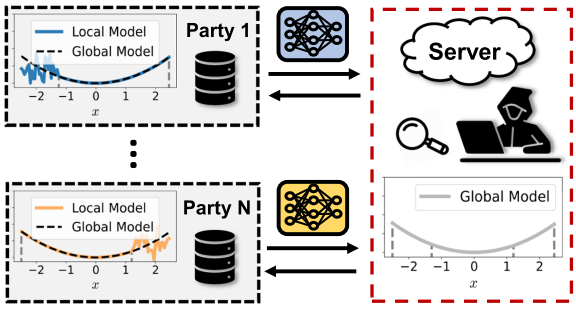


Fig. 1. Data distribution attack in cross-silo FL.

We assume that the server is an honest but curious adversary,[2] i.e., he follows the FL protocol and does not collude with any parties, but attempts to infer sensitive information from the local model parameters shared by the parties. During the FL training process, the local model parameters are revealed to the server for global model aggregation. In addition, we assume that the domain of each feature is known to the server. This is reasonable because this information is considered non-private and is necessary for the anomaly detection and model aggregation [40], [41], [42].

An alternative solution here is to let the parties conceal their model parameters using cryptographic techniques such as multi-party computation [43]; however, as we will detail in Section VI, such techniques have some deficiencies that limit their applicability in practical FL scenarios. In particular, many existing defenses against poisoning attacks depend on the FL server's ability to access and analyze each participant's local model parameters [38], [44], [45], [46]. Without access to the plain-text local model parameters, the FL server can hardly detect malicious behavior of the parties. Our attacks thus highlight the need to develop advanced privacy-preserving frameworks that can conceal data distribution information while still enabling the server to detect malicious parties.

Formally, let T be the number of iterations in FL, $\boldsymbol{\theta}_k^{(t)}$ be the local model parameters submitted by the k-th party in the t-th ($t = 1, 2, \ldots, T$) iteration, and $\boldsymbol{\theta}_G^{(t)}$ be the global model parameters obtained by the server in the t-th iteration. We use $\mathcal{M}_k = \{\boldsymbol{\theta}_k^{(t)}\}$ to denote the set of all local model parameters submitted by the k-th party in all iterations, and $\boldsymbol{\theta}_G^{(T)}$ to denote the final global model obtained by the server. The objective of the adversary (i.e., server) is to infer the distribution of the k-th party's training data, when given $\mathcal{M}_k$ and $\boldsymbol{\theta}_G^{(T)}$. For convenience, we use $\boldsymbol{\theta}_G$ to denote $\boldsymbol{\theta}_G^{(T)}$ in the remainder of the article. The frequently used notations are summarized in Table I.

TABLE I
FREQUENTLY USED NOTATIONS

Notation	Description
f	the machine learning model function
$\boldsymbol{x}$	an input sample of f
N	the number of FL parties
D_k	the local dataset of the k-th party
n_k	the number of training records of the k-th party
n	the total number of training records among all parties, namely, $n = \sum_{k=1}^{N} n_k$
$\mathbf{X}_k$	the features of the k-th party's local dataset
$\mathbf{Y}_k$	the labels of the k-th party's local dataset
$\boldsymbol{\theta}_k^{(t)}$	local model parameters of submitted by the k-th party in the t-th iteration
$\boldsymbol{\theta}_G^{(t)}$	the global model parameters obtained by the server in the t-th iteration
T	the number of global FL iterations

IV. Distribution Inference Attacks

This section presents our attack methods for inferring the distribution of the features and labels in each party's training data. We first introduce the rationale of our methods in Section IV-A, and present how to utilize local models in Section IV-B. Then, we describe the feature distribution inference attacks on differentiable and non-differentiable models in Section IV-C and Section IV-D, respectively. After that, we extend our attacks to the inference on label distribution in Section IV-E. Without loss of generality, we focus on the attack on the k-th party, and use $D_k = (\mathbf{X}_k, \mathbf{Y}_k)$ to denote the k-th party's training data, where $\mathbf{X}_k$ is the features of D_k and $\mathbf{Y}_k$ is the corresponding labels.

A. Rationale

In the FL setting, the local datasets from different parties typically follow different distributions; otherwise, the global model trained via FL would be similar to the model trained on any local dataset, in which case there is no incentive for the parties to conduct the FL task in the first place [1], [37].

When the local datasets follow different distributions, we observe that a local model trained by one party tends to perform worse when tested using the dataset from another party. That is, if Party A trains a local model based on the local training dataset D_A^{train}, and we test the model using Party B's testing dataset D_B^{test}, then the testing accuracy tends to be lower than the case when the testing dataset is from Party A.

On the other hand, the global model constructed by the server combines all local models, and hence, it tends to have high accuracy when tested using any party's local data. Fig. 1 illustrates this phenomenon. Take Party 1 as an example, whose local data is distributed in the range $[-1, 3]$. In contrast, the global data (i.e., the union of the local data of all parties) is distributed in the range $[-3, 3]$. The solid blue curve represents the local model of Party 1, and the black dashed curve represents the global model. When the input data x follows a distribution similar to that of Party 1's local data (i.e., $x \in [-1, 3]$), the outputs of the local model and the global model are close to each other. In contrast, if $x \in [-3, -1]$, the outputs of the local and global models can be significantly different.

Based on the above observations, we propose to exploit the difference between the global model and a party's local model to infer the distribution of the party's data. We elaborate our attack methods in Section IV-C to IV-E.

B. Utilization of Local Models

Recall that in the t-th iteration ($t = 1, 2, \ldots, T$) of the FL process, the k-th party constructs a local model and submits its

[2] We will extend our attacks to party-side attacks in Section IV-F.

parameters $\boldsymbol{\theta}_k^{(t)}$ to the server. Therefore, the server receives T local models in total from the k-th party, i.e., $\mathcal{M}_k = \{\boldsymbol{\theta}_k^{(t)}\}$. Suppose that the server is to infer the distribution of the k-th party's data based on the set of all model parameters shared by the k-th party. In this case, the first decision that the server should make is: among the T local models from the k-th party, which ones should be utilized to launch an inference attack? A naive solution is to randomly choose one of the local models submitted by the k-th party; however, if this particular model happens to capture an insufficient amount of information about the k-th party's local data distribution, then using the model for inference is unlikely to be effective. A more effective approach is to utilize all T local models submitted by the k-th party for inference; but in that case, we need to combine the information from the T local models in a principled manner, so as to improve the accuracy of inference. Towards this end, we propose to *aggregate* the T local models into one model, and then use the aggregated model for inference. In particular, the parameters $\boldsymbol{\theta}_k$ of the aggregated model is a weighted average of the parameters of the T local models:

$$\boldsymbol{\theta}_k = \sum_{t=1}^{T} w_k^{(t)} \boldsymbol{\theta}_k^{(t)},$$

where $w_k^{(t)} \geq 0$ is the weight assigned to the t-th local model and $\sum_{t=1}^{T} w_k^{(t)} = 1$. In what follows, we explain how we decide $w_k^{(t)}$.

Intuitively, a local model $\boldsymbol{\theta}_k^{(t)}$ should be given a large weight if it encapsulates a large amount of information about the k-th party's local data. Motivated by this, we heuristically measure the amount of local data information in $\boldsymbol{\theta}_k^{(t)}$ as: $C_k^{(t)} = \|\boldsymbol{\theta}_k^{(t)} - \boldsymbol{\theta}_G\|_2$. In other words, we evaluate the difference (measured by L_2 distance) between $\boldsymbol{\theta}_k^{(t)}$ and $\boldsymbol{\theta}_G$. The rationale is that if $C_k^{(t)}$ is large, then $\boldsymbol{\theta}_k^{(t)}$ should carry a substantial amount of information specific to the k-th party; otherwise if the local model parameters closely resemble the global model parameters, it is likely that the local model only carry general information about the global data, rather than party-specific insights. Once we derive $C_k^{(t)}$ for $t = 1, 2, \ldots, T$, we convert them into weights $w_k^{(t)}$ using the softmax function, which is a standard approach to normalize a vector into a probability distribution: $w_k^{(t)} = \frac{\exp(C_k^{(t)})}{\sum_{j=1}^{T} \exp(C_k^{(j)})}$. Algorithm 1 presents the pseudo-code of our algorithm for generating the aggregated model.

Algorithm 1: Aggregate Local Model Parameters.

Input: $\mathcal{M}_k = \{\boldsymbol{\theta}_k^{(t)}\}_{t=1}^{T}$: local model parameters of the k-th party; $\boldsymbol{\theta}_G$: final global model parameters
Output: Aggregated local model parameters $\boldsymbol{\theta}_k$

```
for t = 1, ..., T do
    C_k^(t) ← ||θ_k^(t) − θ_G||_2;
for t = 1, ..., T do                    // Compute weights
    w_k^(t) ← exp(C_k^(t)) / Σ_{j=1}^{T} exp(C_k^(j));    // Softmax
θ_k ← Σ_{t=1}^{T} w_k^(t) θ_k^(t);      // Aggregate local models
return θ_k;
```

C. Attack on Differentiable Models

Once we obtain the aggregate local model $\boldsymbol{\theta}_k$ from Algorithm 1, we may compare it against the final global model $\boldsymbol{\theta}_G$ to launch an inference attack on the k-th party.

The basic idea of our attack is to first generate synthetic tuples $\boldsymbol{x}$ that minimize the distance between $f(\boldsymbol{x}; \boldsymbol{\theta}_k)$ and $f(\boldsymbol{x}; \boldsymbol{\theta}_G)$, where $f(\boldsymbol{x}; \boldsymbol{\theta})$ denotes the model's output when it is parameterized by $\boldsymbol{\theta}$ and given $\boldsymbol{x}$ as the input. After that, we utilize the feature distribution of these synthetic samples to estimate the actual feature distribution of $\mathbf{X}_k$. The rationale is that if $f(\boldsymbol{x}; \boldsymbol{\theta}_k)$ and $f(\boldsymbol{x}; \boldsymbol{\theta}_G)$ have a small distance, then $\boldsymbol{x}$ is likely to resemble some sample in the k-th party's local training data; otherwise, the k-th party's local model can hardly map $\boldsymbol{x}$ to an output similar to $f(\boldsymbol{x}; \boldsymbol{\theta}_G)$ from the global model. The question is: how can we generate a large number of such samples $\boldsymbol{x}$ in an efficient manner?

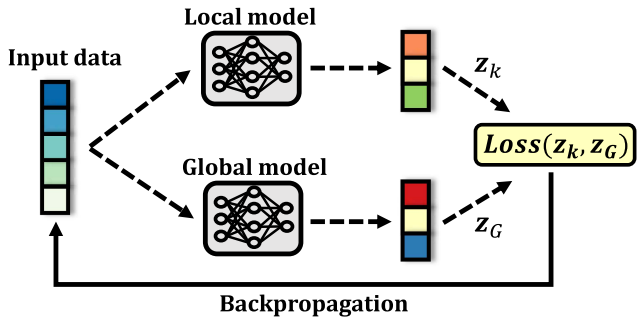


Fig. 2. Workflow of PGDA.

To answer the above question, we consider the following constrained optimization problem:

$$\arg\min_{\boldsymbol{x} \in \mathcal{Q}} \mathcal{L}(\boldsymbol{x}; \boldsymbol{\theta}_k, \boldsymbol{\theta}_G) = \Delta\left(f(\boldsymbol{x}; \boldsymbol{\theta}_k), f(\boldsymbol{x}; \boldsymbol{\theta}_G)\right) + \Omega(\boldsymbol{x}), \tag{3}$$

where $\mathcal{Q}$ is the domain of the feature space, $\Delta(\cdot, \cdot)$ is a distance function that measures the difference between $f(\boldsymbol{x}; \boldsymbol{\theta}_k)$ and $f(\boldsymbol{x}; \boldsymbol{\theta}_G)$, and $\Omega(\cdot)$ is a regularization term. Specifically, we define $\Delta(\cdot, \cdot)$ as the mean square error (MSE) function:

$$\Delta(\boldsymbol{z}_1, \boldsymbol{z}_2) = \frac{1}{c}\|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2^2,$$

where $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ are vectors of length c. In addition, we define

$$\Omega(\cdot) = \gamma \cdot \mathrm{Var}(\boldsymbol{x}),$$

where $\mathrm{Var}(\boldsymbol{x}) = \frac{1}{d}\sum_{i=1}^{d}(\boldsymbol{x}[i] - \frac{1}{d}\sum_{j=1}^{d}\boldsymbol{x}[j])^2$ and γ is a hyper-parameter, with $\boldsymbol{x}[j]$ denoting the j-th element of $\boldsymbol{x}$ and d denoting the number of elements in $\boldsymbol{x}$.

Deriving an analytical solution to (3) is challenging, especially when f is non-convex (which is often the case in practice). To address this issue, we design an algorithm based on *projected gradient descent (PGD)* [35] to identify a local minimum of the objective function when f is differentiable. We refer to this PGD-based attack methods as *PGDA*. The workflow of PGDA is illustrated in Fig. 2. Specifically, we first randomly generate an input sample and then compute the outputs $f(\boldsymbol{x}; \boldsymbol{\theta}_k)$ and $f(\boldsymbol{x}; \boldsymbol{\theta}_G)$ *w.r.t.* the party's local model and the global model,

respectively. The gradient of the input sample is calculated by backward propagation according to the value of the loss function, based on which the input sample is updated and then projected onto the feature space $\mathcal{Q}$. In the end, we obtain a synthetic sample $\boldsymbol{x} \in \mathcal{Q}$ such that the MSE between $f(\boldsymbol{x}; \boldsymbol{\theta}_k)$ and $f(\boldsymbol{x}; \boldsymbol{\theta}_G)$ is small. We repeat the above process until a sufficient number of synthetic samples are obtained. After that, we compute the feature distribution of the synthetic samples, and use it as an estimation of the feature distribution in the k-th party's local data.

One issue of the PGD-based attack is that the distribution of the synthetic samples $\boldsymbol{x}$ may deviate from the distribution of $\mathbf{X}_k$ because of the randomness of the party's local model on the unseen testing samples, i.e., the outputs of the local model and the global model may happen to be very similar on some data points $\boldsymbol{x} \nsim P_k$, where P_k denotes the distribution of $\mathbf{X}_k$. As a consequence, false-positive samples may be generated by PGD, leading to a low attack accuracy. To reduce the number of false-positive samples and improve the attack accuracy, we design a sampling mechanism based on input perturbation, ensuring that the outputs of the local model and the global model are also close on the synthetic samples' neighbors. Specifically, after obtaining a synthetic sample $\boldsymbol{x}$ by PGD, we generate r perturbed samples $\{\hat{\boldsymbol{x}}_i\}_{i=1}^{r}$, where $\hat{\boldsymbol{x}}_i = \boldsymbol{x} + \nu_i$ and ν_i is a random noise vector drawn from a Gaussian distribution with variance σ. Accordingly, the average difference of model outputs is computed based on these samples:

$$\overline{\Delta} = \frac{1}{r} \sum_{i=1}^{r} \Delta \left(f(\hat{\boldsymbol{x}}_i; \boldsymbol{\theta}_k), f(\hat{\boldsymbol{x}}_i; \boldsymbol{\theta}_G) \right).$$

Based on the value of $\overline{\Delta}$, we determine whether to choose the synthetic sample $\boldsymbol{x}$: if $\overline{\Delta}$ is greater than a threshold Δ_τ, then we discard $\boldsymbol{x}$. The rationale is that for any synthetic sample $\boldsymbol{x}$ generated by PGD, a large value of $\overline{\Delta}$ indicates a significant difference between the local and global model outputs computed from its neighboring samples, i.e., $\Delta(f(\boldsymbol{x}; \boldsymbol{\theta}_k), f(\boldsymbol{x}; \boldsymbol{\theta}_G))$ is small on $\boldsymbol{x}$ but large on its neighbors. This indicates that $\boldsymbol{x}$ is likely to be a false-positive. By avoiding such potential false-positives, we can improve the overall quality of the synthetic samples, thus producing a more accurate estimation of the k-th party's data distribution.

Algorithm 2 presents the pseudo-code of PGDA. The loss function $\mathcal{L}$ in line 2 is defined identically to the objective function given in (3), and the Δ_τ in line 2 represents the empirical median of $\overline{\Delta}$. By setting Δ_τ as an empirical median value, each generated sample will be rejected with a probability of roughly $1/2$, thus keeping the attack process effective and efficient. In the experiments, we use the median value of the first 100 $\overline{\Delta}$s as the Δ_τ.

Algorithm 2: Projected Gradient Descent Attack.

Input: $\boldsymbol{\theta}_k$: local model parameters; $\boldsymbol{\theta}_G$: global model parameters; n: number of synthetic samples
Output: Synthetic data set S

1 $S \leftarrow \emptyset$;
2 **while** $|S| < n$ **do**
3 $\quad \boldsymbol{x}_0 \leftarrow \text{RANDSAMPLE}()$; `// Random point`
4 $\quad$ **for** $t = 1, 2, \ldots, \text{iter}_{\max}$ **do**
5 $\quad\quad \nabla \leftarrow \partial \mathcal{L}(\boldsymbol{x}_{t-1}; \boldsymbol{\theta}_k, \boldsymbol{\theta}_G) / \partial \boldsymbol{x}_{t-1}$;
6 $\quad\quad \boldsymbol{y}_t \leftarrow \boldsymbol{x}_{t-1} - \alpha_t \nabla$; `// Gradient descent`
7 $\quad\quad \boldsymbol{x}_t \leftarrow \underset{\boldsymbol{x} \in \mathcal{Q}}{\text{argmin}} \frac{1}{2} \|\boldsymbol{x} - \boldsymbol{y}_t\|_2^2$; `// Projection`
8 $\quad$ **for** $i = 1, 2, \ldots, r$ **do**
9 $\quad\quad \nu_i \sim \mathcal{N}(0, \sigma^2 \mathbb{I})$;
10 $\quad\quad \hat{\boldsymbol{x}}_i \leftarrow \boldsymbol{x}_t + \nu_i$; `// Input perturbation`
11 $\quad \overline{\Delta} \leftarrow \frac{1}{r} \sum_{i=1}^{r} \Delta \left(f(\hat{\boldsymbol{x}}_i; \boldsymbol{\theta}_k), f(\hat{\boldsymbol{x}}_i; \boldsymbol{\theta}_G) \right)$;
12 $\quad$ **if** $\overline{\Delta} < \Delta_\tau$ **then**
13 $\quad\quad S \leftarrow S \cup \{\boldsymbol{x}_t\}$;
14 **return** S;

D. Attack on Non-Differentiable Models

Our PGDA approach in Section IV-C requires that the model function f is differentiable. For the case when f is non-differentiable, we propose to use numerical optimization methods to search for some $\boldsymbol{x}$ that makes the outputs of the local model and the global model close enough to each other. Towards this end, we design a hill-climbing (HC) [36] algorithm to solve (3) on non-differentiable models. In a nutshell, the HC algorithm searches for a local optimum by first starting from a random solution and then iteratively modifying the current solution to improve its quality. We refer to the HC-based attack on non-differentiable models as the *generic distribution inference attack (GDIA)*.

Algorithm 3 details the GDIA method. The hill-climbing searching subroutine is described in Lines 3-3. First, the adversary randomly initializes a data sample $\boldsymbol{x}^*$ from the valid space $\mathcal{Q}$ and computes the corresponding objective function $\mathcal{L}(\boldsymbol{x}^*; \boldsymbol{\theta}_k, \boldsymbol{\theta}_G)$. The objective function is the same as that in (3). Then, to obtain a better candidate $\boldsymbol{x}$, the adversary randomly selects d features of the current solution $\boldsymbol{x}^*$ and randomly modifies their values. If the resulting solution makes the objective function $\mathcal{L}$ smaller, the adversary considers it a better solution and replaces $\boldsymbol{x}^*$ with it; otherwise, the adversary rejects this solution and attempts to find a better candidate by repeating the above process (line 3-3 in Algorithm 3). Note that the parameter d controls the step size of the searching process. We set the initial d to the total number of features at the beginning of the search process, and reduce d to half after $\text{rej}_{\max}$ continuous rejections. Finally, if the searching algorithm finds a solution that makes the value of $\mathcal{L}$ smaller than the threshold $\tau_{\min}$ before reaching the maximum number of iterations $\text{iter}_{\max}$, it stops the iteration and returns the solution. Otherwise, it returns a $\perp$ symbol, indicating that the valid solution cannot be found efficiently. The adversary repeats the above process until collecting a sufficient number of samples, and then uses them to infer the feature distribution of the private training data. To further enhance the effectiveness of the attack, the adversary also integrates the input perturbation-based sampling mechanism introduced in Section IV-C into Algorithm 3.

Note that the threshold $\tau_{\min}$ is a hyper-parameter, and a smaller $\tau_{\min}$ leads to better attack results but causes higher computation costs. To achieve a better accuracy-computation trade-off, we dynamically relax $\tau_{\min}$ if the time overhead is too

Algorithm 3: Generic Distribution Inference Attack.

Input: $\boldsymbol{\theta}_k$: local model parameters; $\boldsymbol{\theta}_G$: global model parameters; n: number of synthetic samples; $\tau^*_{\min}$: threshold; $\text{fail}_{\max}$: timing of the relaxation of $\tau_{\min}$; $\text{rej}_{\max}$: max number of continuous rejections; d: number of features; $d_{\min}$: minimum dimension of searching space

Output: Synthetic data set S

1 $S \leftarrow \varnothing$;
2 $j \leftarrow 0$;
3 $\tau_{\min} \leftarrow \tau^*_{\min}$; // Initial threshold
4 **for** $t = 1, 2, \ldots, n$ **do**
5 $\quad \boldsymbol{x} \leftarrow \texttt{Search}(\tau_{\min})$;
6 $\quad$ **while** $\boldsymbol{x} = \perp$ **do** // Failed to synthesize
7 $\quad\quad \boldsymbol{x} \leftarrow \texttt{Search}(\tau_{\min})$;
8 $\quad\quad j \leftarrow j + 1$;
9 $\quad\quad$ **if** $j > \text{fail}_{\max}$ **then** // Relax threshold
10 $\quad\quad\quad \tau_{\min} \leftarrow 2 \cdot \tau_{\min}$;
11 $\quad\quad\quad j \leftarrow 0$;
12 $\quad j \leftarrow 0$;
13 $\quad S \leftarrow S \cup \{\boldsymbol{x}\}$;
14 **return** S;
15 **Function** $\texttt{Search}(\tau_{\min})$:
16 $\boldsymbol{x}^* \leftarrow \text{RANDSAMPLE}()$;
17 $\tau^* \leftarrow \mathcal{L}(\boldsymbol{x}^*; \boldsymbol{\theta}_k, \boldsymbol{\theta}_G)$;
18 $j \leftarrow 0$;
19 **for** $t = 1, 2, ..., \text{iter}_{\max}$ **do**
20 $\quad$ **if** $\tau^* < \tau_{\min}$ **then**
21 $\quad\quad$ **return** $\boldsymbol{x}^*$; // Accept the sample
22 $\quad \boldsymbol{x} \leftarrow \text{RANDSAMPLE}(\boldsymbol{x}^*, d)$; // Randomize d features
23 $\quad \tau \leftarrow \mathcal{L}(\boldsymbol{x}; \boldsymbol{\theta}_k, \boldsymbol{\theta}_G)$;
24 $\quad$ **if** $\tau < \tau^*$ **then** // A better solution
25 $\quad\quad \boldsymbol{x}^* \leftarrow \boldsymbol{x}$;
26 $\quad\quad \tau^* \leftarrow \tau$;
27 $\quad\quad j \leftarrow 0$;
28 $\quad j \leftarrow j + 1$;
29 $\quad$ **if** $j > \text{rej}_{\max}$ **then** // Reduce search space
30 $\quad\quad d \leftarrow \max(k_{\min}, \lceil d/2 \rceil)$;
31 $\quad\quad j \leftarrow 0$;
32 **return** $\perp$;

high to find a feasible solution by the hill-climbing searching. Specifically, we use a maximum fail parameter $\text{fail}_{\max}$ to help determine the timing of the relaxation of $\tau_{\min}$: the adversary first sets a small $\tau^*_{\min}$ then dynamically relaxes the value of $\tau_{\min}$ to $2\tau_{\min}$ (lines 3-3) if the hill-climbing subroutine returns $\text{fail}_{\max}$ number of continuous $\perp$. We defer details of hyper-parameter setting to Section V-A.

Complexity Analysis: We now analyze the time complexity of GDIA. Suppose that all the feature values are in the domain $[a, b]$. Let $\boldsymbol{v}_k = f(\boldsymbol{x}; \boldsymbol{\theta}_k)$, $\boldsymbol{v}_G = f(\boldsymbol{x}; \boldsymbol{\theta}_G)$, and let $\boldsymbol{x}[i]$ denote the i-th element of the input data $\boldsymbol{x}$. Because $\boldsymbol{v}_k$ and $\boldsymbol{v}_G$ are confidence score vectors, we have $MSE(\boldsymbol{v}_k, \boldsymbol{v}_G) = \frac{1}{c}\|\boldsymbol{v}_k - \boldsymbol{v}_G\|_2^2 \leq 1$. Let $\mu = \frac{1}{d}\sum_{i=1}^{d} \boldsymbol{x}[i]$ with d denoting the number of elements in $\boldsymbol{x}$ and let $M = \max\{(a-\mu)^2, (b-\mu)^2\}$, then we have $\Omega(\boldsymbol{x}) = \gamma \cdot \text{Var}(\boldsymbol{x}) = \gamma \cdot \frac{1}{d}\sum_{i=1}^{d}(\boldsymbol{x}[i] - \mu)^2 \leq \gamma M$. Accordingly, we have $\mathcal{L}(\boldsymbol{x}; \boldsymbol{\theta}_k, \boldsymbol{\theta}_G) = MSE(\boldsymbol{v}_k, \boldsymbol{v}_G) + \Omega(\boldsymbol{x}) \leq 1 + \gamma M$. This indicates that the threshold $\tau_{\min}$ will be relaxed at most $O(\log \frac{1+\gamma M}{\tau^*_{\min}})$ times. After relaxing the threshold to a suitable value, the algorithm will find $|S|$ solutions in $O(|S| \cdot \text{fail}_{\max} \cdot \text{iter}_{\max} \cdot n_\theta)$ time where n_θ denotes the inference time complexity of the model. Thus, the overall time complexity is $O((\log \frac{1+\gamma M}{\tau^*_{\min}} + |S|) \cdot \text{fail}_{\max} \cdot \text{iter}_{\max} \cdot n_\theta)$.

E. Attack on Label Distribution

The labels $\mathbf{Y}_k$ of the k-th party's training data D_k are also highly sensitive because they are the most meaningful and informative indicators for classification tasks [47]. We now describe how to extend the feature distribution attacks to the label distribution attack. Notice that in both PGDA and GDIA, the adversary generates a set of synthetic samples S without label information. A straightforward approach is to compute the predicted labels of these synthetic samples using the local model $f(\cdot; \boldsymbol{\theta}_k)$ and estimate the real label distribution of $\mathbf{Y}_k$ based on the predicted labels. For example, the predicted labels of a synthetic dataset S can be computed by $L = \left\{\arg\max_{1 \leq i \leq c} f_i(\boldsymbol{x}; \boldsymbol{\theta}_k) \middle| \boldsymbol{x} \in S\right\}$, where c is the number of classes and $f_i(\boldsymbol{x}; \boldsymbol{\theta}_k)$ represents the i-th element of the output vector.

Nevertheless, this naive method may cause inaccurate estimations of labels $\mathbf{Y}_k$. Consider a binary classifier $f(\boldsymbol{x}; \boldsymbol{\theta})$ that outputs a confidence score vector $\boldsymbol{v} = (v_1, v_2)$. Suppose for each sample in the synthetic data set $\boldsymbol{x} \in S$, the model always outputs $\boldsymbol{v} = (0.8, 0.2)$, which can be caused by a biased training dataset with 80% positive samples and 20% negative samples. The estimated label set L produced by the naive method will only contain the first label, leading to a biased estimation of the true label distribution.

To address this issue, we make full use of the information contained in confidence score vectors, rather than only utilizing predicted labels. Specifically, given a set of synthetic samples S, the adversary does not add the prediction outputs into L directly. Instead, he attempts to add up confidence score vectors of all synthetic samples

$$\boldsymbol{v} = \sum_{\boldsymbol{x} \in S} f(\boldsymbol{x}; \boldsymbol{\theta_k}) \in \mathbb{R}^c,$$

and then normalize the vector $\boldsymbol{v}$ into a probability distribution, and use the normalized vector to estimate the label distribution of party's local data. For example, in the above case, the adversary first adds up all confidence score vectors: $\boldsymbol{v} = (0.8 \cdot |S|, 0.2 \cdot |S|)$, and then computes the normalized vector by $\hat{\boldsymbol{v}} = \boldsymbol{v} / \sum_{i=1}^{c} \boldsymbol{v}_i$, where $\boldsymbol{v}_i$ is the i-th element of vector $\boldsymbol{v}$. Finally, the adversary uses $\hat{\boldsymbol{v}}$ as the estimated label distribution. The details of the label distribution attack is presented in Algorithm 4.

F. Party-Side Attacks

So far, we have demonstrated how FL servers can launch distribution inference attacks. It is worth noting that our methods can be easily generalized to party-side attacks. When multiple FL parties collude to attack the remaining parties, party-side attacks can also lead to serious privacy leakage issues. Specifically, the colluding parties may obtain the aggregated local models

Algorithm 4: Label Distribution Attack.

Input: S: synthetic dataset; $\boldsymbol{\theta}_k$: local model parameter; c: number of output classes
Output: Estimated label distribution (proportion) $\hat{\boldsymbol{v}}$

$\boldsymbol{v} \leftarrow \mathbf{0}$; // $\mathbf{0} \in \mathbb{R}^c$
forall $\boldsymbol{x} \in S$ **do**
  $\boldsymbol{v} \leftarrow \boldsymbol{v} + f(\boldsymbol{x}; \boldsymbol{\theta}_k)$;
$\hat{\boldsymbol{v}} \leftarrow \boldsymbol{v} / \sum_{i=1}^{c} \boldsymbol{v}_i$; // Normalization
return $\hat{\boldsymbol{v}}$;

from rest parties as follows[3]

$$\sum_{i \in \{1,\ldots,N\} \setminus A} \frac{n_i}{n - \sum_{j \in A} n_j} \boldsymbol{\theta}_i^{(t)} = \frac{n \boldsymbol{\theta}_G^{(t)} - \sum_{j \in A} n_j \boldsymbol{\theta}_j^{(t)}}{n - \sum_{j \in A} n_j}, \quad (4)$$

where A denotes the set of IDs of colluded FL parties. In this case, the colluded parties may infer the data distribution of aggregated data from other parties by launching the attack methods described in previous sections.

V. Experimental Evaluation

In this section, we conduct extensive experiments to evaluate the performance of our attack methods. The experimental setup is described in Section V-A, while Section V-B presents the visualization of the key intuition behind our attacks. The results of the attacks under various settings are reported in Section V-C to V-H. Finally, Section V-I summarizes the experimental findings.

A. Setup

We implement the proposed algorithms in Python and use *PyTorch* (www.pytorch.org) to train the FL models. All experiments are performed on a machine with two Xeon Gold 6240@2.60 GHz CPUs and 384 GB of RAM.

Datasets: Our evaluations are based on six benchmark datasets: Abalone [48], Cardiotocograph [48], Credit Card [49], Drive Diagnosis [48], News Popularity [50], and Wine Quality [51]. All the above datasets are downloaded from the UCI Machine Learning Repository [48]. In addition, we utilize two synthetic datasets generated through the scikit-learn library.[4] Each dataset contains 200,000 records and is classified into 5 classes. The first dataset comprises 50 features, while the second has 100 features. In both cases, 70% of the features are designated as useful. The remaining features (i.e., non-useful features) are randomly drawn from a standard normal distribution. The statistics of all datasets are shown in Table II. The value ranges of all features in the training datasets are normalized into [0, 1] before the experiments [41], [52]. Following previous works on cross-silo FL [53], [54], we use four parties in our experiments by default. The impact of the number of parties on the attack performance will be discussed in Section V-E.

[3]Equation (4) can be derived directly from (2).

[4][Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_classification.html

TABLE II
Details of the Datasets

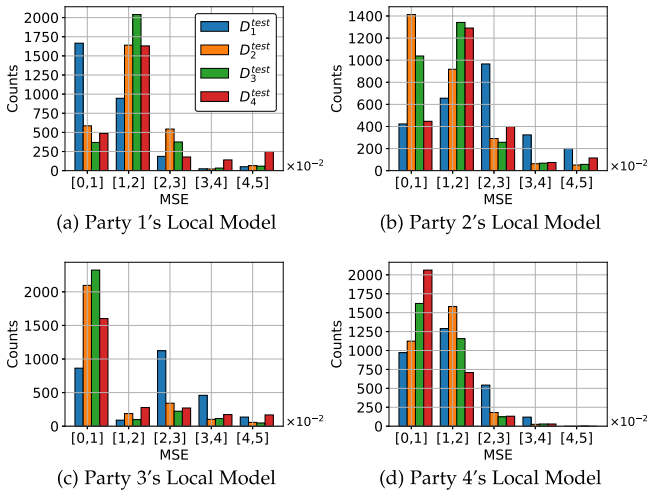
Name	# Records	# Features	# Classes
Abalone	4,177	9	5
Cardiotocograph	2,126	21	10
Credit Card	30,000	24	2
Drive Diagnosis	58,509	48	11
Online News Popularity	39,797	59	5
Wine Quality	4,898	11	6
Synthetic-1	200,000	50	5
Synthetic-2	200,000	100	5

Datasets Partation: Before the training phase, we need to generate four non-i.i.d. local datasets from a dataset D. Considering that the label distributions of different parties are typically biased in the cross-silo setting [1], [2], we first select the top five features which have the highest Pearson correlation coefficients with the labels, then divide D into four non-i.i.d. local datasets based on each of these five features respectively to generate five groups of non-i.i.d. local datasets. We conduct experiments independently for each group of datasets, and report the final attack result by the average results.

Specifically, suppose the dataset partition is based on the f-th feature. First, we divide the training datasets D into four disjoint datasets $\{D_1, D_2, D_3, D_4\}$ with the same size based on the value of the f-th feature. Formally, let $\boldsymbol{x}_f^{i,m}$ denote the value of the f-th feature with respect to the m-th sample in D_i, we have $\boldsymbol{x}_f^{i,m} \leq \boldsymbol{x}_f^{j,m'}$ for any $1 \leq i < j \leq 4$, $m \in \{1, \ldots, |D_i|\}$ and $m' \in \{1, \ldots, |D_j|\}$, i.e., the four datasets are partially ordered with respect to $\boldsymbol{x}_f$. Then, we compose the local dataset of party k by randomly sampling $|D_k|/2$ data records from D_k and randomly sampling $|D_k|/2$ records from $D \setminus D_k$ to vary the feature distribution across parties, which is consistent with the cross-silo FL settings in [3], [54]. In this way, we can generate four non-i.i.d. local training datasets for the four FL parties of the same size. Because of the strong correlations between the selected features (dividers) and the labels, we can further ensure that the labels of the local datasets are also non-i.i.d.

Models: The LR model consists of one linear layer with the input size equal to the number of features and the output size equal to the number of classes. For NN models, we use the four-layer perceptrons, where the sizes of the input and output layers are the same as that of the LR models, and the two hidden layers consist of 600 and 100 neurons, respectively. ReLU is used as the activation function of the NN models, and Dropout [55] with $p = 0.2$ is used as the regularization method for the NN training. As for the DP mechanism, we use the state-of-the-art DP-SGD algorithm [19] to train local models. We use *TensorFlow Privacy* (github.com/tensorflow/privacy) to calibrate DP noises. The cross-entropy loss are used as the loss function (i.e., $\ell(\cdot, \cdot)$ in (1)) to train all models.

Hyper-Parameters: In the attack experiments, we use the same hyper-parameter setting for all the datasets. The weight of the regularization term γ in (3) is set to 0.1. For the PGDA attack, we set the number of total iterations $\text{iter}_{\max} = 25$, the learning rate $\alpha = 0.1$, the number of random noises for input perturbation $r = 100$, and the variance of random noises $\sigma^2 = 0.015^2$. For the GDIA attack, inspired by the experimental results shown in



(a) Party 1's Local Model (b) Party 2's Local Model

(c) Party 3's Local Model (d) Party 4's Local Model

Fig. 3. Distribution histogram of party k's $MSE(\boldsymbol{v}_k, \boldsymbol{v}_G)$ ($k \in \{1, \ldots, 4\}$) tested on different datasets $\{D_k^{\text{test}}\}_{k=1}^4$.

Fig. 3, we set the initial threshold $\tau_{\min}^* = 10^{-3}$, $\text{fail}_{\max} = 10$, $\text{iter}_{\max} = 1,000$, $\text{rej}_{\max} = 10$, and $k_{\min} = 4$ so that the attack can be completed efficiently (within 1 h) for each dataset. For both PGDA and GDIA, the number of synthetic samples is set to $n = 1,000$.

As for the FL model training, we set the number of local and global iterations to 100 and 5, respectively. The per-example gradient norm clipping threshold is set to 2 and 4 for LR model and NN model, respectively. For all datasets except Cardiotocograph, we set the sampling rate of DP-SGD to 0.05. For Cardiotocograph, since the training records is relatively small, we increase the sampling rate to 0.2 to reduce the impact of DP noise on model accuracy. All models are trained with a fixed learning rate of 0.15.

Metrics: We measure the performance of the distribution inference attacks by computing the first-Wasserstein distance (Earth Mover's distance) [56] between the estimated distribution and the ground truth distribution. The first-Wasserstein distance between two distributions P and Q is defined as follows:

$$W(P, Q) = \inf_{\pi \in \Gamma(P,Q)} \int |x - y| d\pi(x, y),$$

where $\Gamma(P, Q)$ is a set of joint distributions π for (x, y) that have marginals P and Q on the first and second factors, respectively. We choose Wasserstein distance instead of other metrics such as Kullback-Leibler and Jensen-Shannon divergence because the Wasserstein distance is more robust: the value of Wasserstein distance is meaningful even when two distributions have no intersections [56]. We evaluate the attack performance by averaging the Wasserstein distance over all features for a dataset:

$$W_{avg} = \frac{1}{N n_f} \sum_{k=1}^{N} \sum_{i=1}^{n_f} W(P_k^{(i)}, Q_k^{(i)}), \tag{5}$$

where n_f is the number of features, N is the number of parties, $Q_k^{(i)}$ and $P_k^{(i)}$ are the estimated distribution and the ground truth distribution of the i-th feature for the k-th party, respectively. Note that a smaller Wasserstein distance indicates a better attack result. We use *scipy* (scipy.org) to calculate the Wasserstein distance.

Baselines: To our best knowledge, there are no studies considering the distribution inference attacks in federated learning. Therefore, we compare the proposed attacks with random guess baselines. For the *feature distribution inference* attack, two baselines are used: the first baseline generates samples from a uniform distribution $\mathcal{U}(0, 1)$; the second baseline generates samples from a Gaussian distribution $\mathcal{N}(0.5, 0.25^2)$, which can produce values falling into the normalized feature range $[0, 1]$ with a probability of at least 95% [41], [52].

For the *label distribution inference* attack, since labels are discrete data, we adopt a different baseline. To generate random guesses for labels, we first draw c random variables $\{g_1, \ldots, g_c\}$ from the uniform distribution $\mathcal{U}(0, 1)$, then calculate the normalized probability as $\overline{g}_i = g_i / \sum_{j=1}^{c} g_j$. We use $\overline{g}_i$ as the random guess result of the i-th label. We independently generate 100 random guesses in the experiment and use the corresponding averaged Wasserstein distance as the baseline. For simplicity, we call all the baselines the *random guess (RG)* in the following sections.

B. Visualization of the Attack Intuition

We present the visualization results of our attack intuition that is described in Section IV-A. We perform FL training using NN model on Drive Diagnosis dataset with privacy budget $\epsilon = 4$ and $\delta = 10^{-5}$. For the local dataset D_k of party k, we randomly split it into an 80% training dataset D_k^{train} and a 20% testing dataset D_k^{test}. After the training phase, we compare the difference of outputs between local model $\boldsymbol{\theta}_k$ and global model $\boldsymbol{\theta}_G$ with the testing datasets $\{D_k^{\text{test}}\}_{k=1}^4$ by computing the mean square error (MSE). The results are shown in Fig. 3, where the x axis denotes the $MSE(\boldsymbol{v}_k, \boldsymbol{v}_G)$ intervals, and the y axis denotes the number of testing samples whose outputs fall into the corresponding intervals. We observe that the $MSE(\boldsymbol{v}_k, \boldsymbol{v}_G)$ *w.r.t.* party k tends to be small when the testing data x is in D_k^{test}, which inspires us that we can synthesize a dataset with small values of $MSE(\boldsymbol{v}_k, \boldsymbol{v}_G)$ and use its distribution as the estimation of the ground truth local data distribution.

C. Evaluation of Feature Distribution Attacks

Since the privacy level of DP depends on the privacy budgets (ϵ, δ), we evaluate the effectiveness of feature distribution inference attacks against different (ϵ, δ)-DP settings.

PGDA Attack on Differentiable Models: We evaluate PGDA on NN and LR models with different (ϵ, δ) settings. Specifically, we vary ϵ by $\{1/4, 1/2, 1, 2, 4, 8, \infty\}$ and fix $\delta = 1/2n$, where n is the number of training samples of each party, as suggested in [19], [57]. Note that a smaller ϵ offers a stronger privacy guarantee but harms the model utility, and $\epsilon = 4$ is a widely used setting [19], [57] for achieving a good privacy-utility trade-off. Figs. 4 and 5 show the attack results on NN and LR models, respectively. We can observe that for all datasets, the average Wasserstein distances between the ground truth and our attack

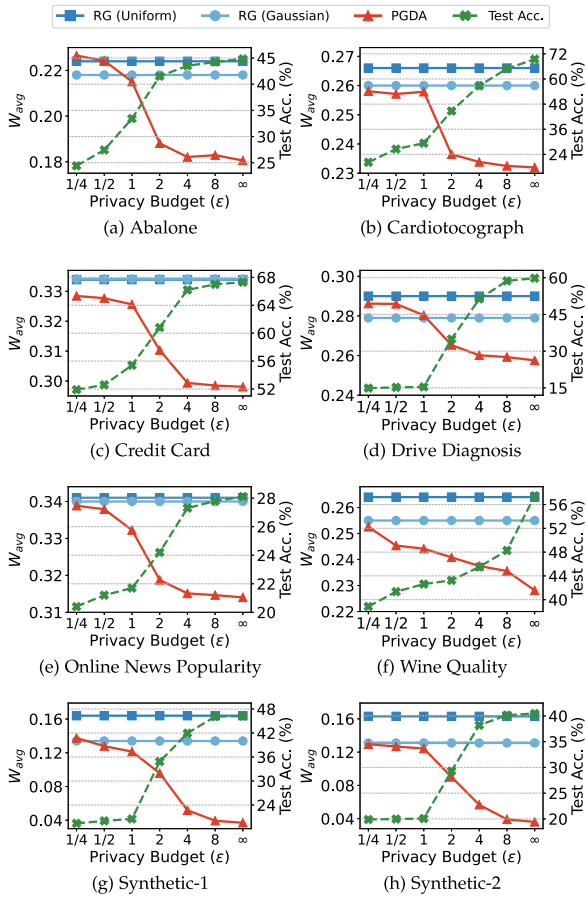


Fig. 4. Feature distribution attack on NN models with different ϵ settings.

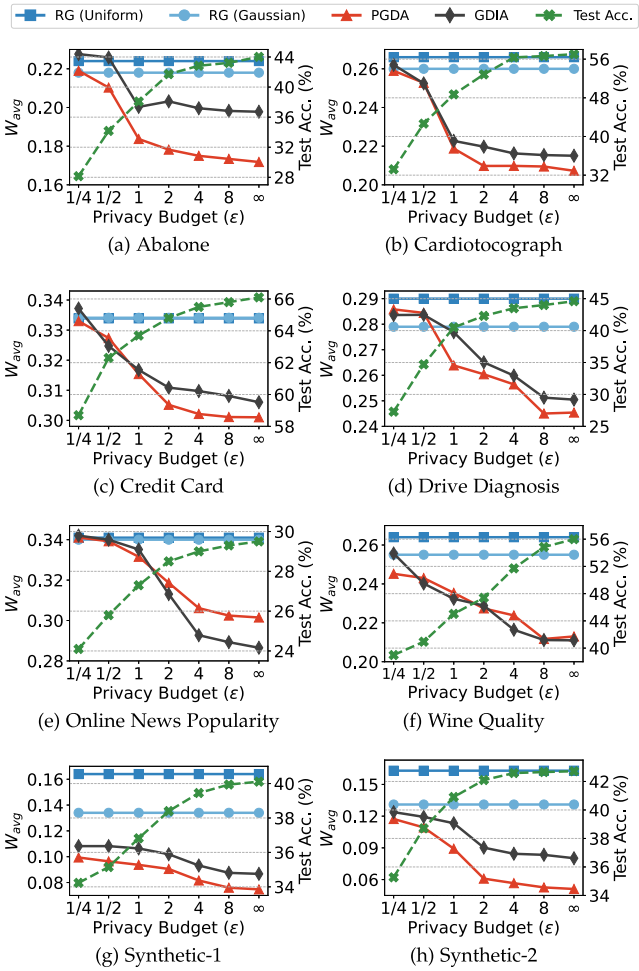


Fig. 5. Feature distribution attack on LR models with different ϵ settings.

results are significantly smaller than the two baselines when the privacy budgets $\epsilon \geq 2$. This indicates that the adversary can successfully infer feature distribution information in each party. In contrast, the attacker can obtain little useful information when privacy budgets are extremely small (e.g., $\epsilon \leq 1$). However, the model accuracy is significantly degraded in these cases due to the large DP noise. For example, in Fig. 4(d), the attack performance degenerates to random guess when ϵ is set smaller than 1, but compared to the model with $\epsilon = 4$, the test accuracy of the model with $\epsilon = 1$ decreases 40%, which would be unacceptable in practice.

GDIA Attack on Non-Differentiable Models: We now evaluate the attack performance of GDIA *w.r.t.* to different ϵs. So far, the only class of well-studied non-differentiable models in FL is tree-based models [58], [59], and to our best knowledge, no work considers differentially private tree model training in cross-silo FL. The only possible existing solution for conducting DP-tree model training in the cross-silo FL setting is using a peer-to-peer manner [60]. However, training the fully distributed tree model is not effective or efficient in practice [58], [61], and the application scenarios considered in the training algorithm design in [60] are significantly different from FL. Nevertheless, we can still demonstrate the effectiveness of GDIA on non-differentiable models by taking the differentially private LR model as a black-box model and using GDIA to infer the feature distribution without computing model gradients during the attack process.

Fig. 5 (black solid curves) shows the attack results of GDIA, which are similar to those of PGDA on LR and NN models. When $\epsilon \geq 2$, we can obtain the training data distribution with high accuracy. Meanwhile, when $\epsilon \leq 1$, the attack results are undesirable while the model accuracies are also significantly low. As a result, Fig. 5 also validates GDIA's effectiveness on non-differentiable models.

Visualization of Attack Results: To provide an intuitive view of the attack results, we visualize the probability density functions *w.r.t.* the "height" feature of the Abalone dataset recovered by PGDA against LR models (trained with $\epsilon = 4$) in Fig. 6, from which we can see that our attacks can obtain a good estimation of the private feature distributions of local parties.

D. Evaluation of Label Distribution Attacks

Next, we evaluate the performance of the label distribution attack on different models. Table III reports the Wasserstein distances of the attack results on all datasets with different models and privacy budgets, and Fig. 7 visualizes the attack results of the label distribution attack on the Drive Diagnosis dataset. The

TABLE III
WASSERSTEIN DISTANCES OF THE LABEL DISTRIBUTION ATTACK RESULTS

Dataset	Model	Privacy Budget (ϵ)							Random Guess
		$\epsilon=\infty$	$\epsilon=8$	$\epsilon=4$	$\epsilon=2$	$\epsilon=1$	$\epsilon=1/2$	$\epsilon=1/4$	
Abalone	LR	**0.22**	**0.26**	**0.27**	**0.29**	**0.40**	0.63	0.77	0.54
	NN	**0.27**	**0.27**	**0.28**	**0.31**	0.69	0.78	1.01	
	GDIA	**0.18**	**0.23**	**0.23**	**0.27**	**0.36**	0.78	1.14	
Cardiotocograph	LR	**0.35**	**0.36**	**0.50**	**0.54**	**0.71**	1.36	1.67	1.16
	NN	**0.33**	**0.38**	**0.58**	1.32	1.53	2.03	2.19	
	GDIA	**0.35**	**0.37**	**0.45**	**0.46**	**0.61**	1.45	1.82	
Credit Card	LR	**0.09**	**0.09**	**0.10**	**0.12**	**0.17**	0.26	0.29	0.19
	NN	**0.14**	**0.17**	**0.18**	0.22	0.23	0.29	0.30	
	GDIA	**0.10**	**0.10**	**0.11**	**0.12**	**0.14**	0.23	0.31	
Drive Diagnosis	LR	**0.36**	**0.40**	**0.42**	**0.43**	**0.49**	**0.63**	1.31	0.93
	NN	**0.29**	**0.55**	**0.78**	**0.91**	1.05	2.19	2.91	
	GDIA	**0.24**	**0.41**	**0.41**	**0.46**	**0.58**	**0.72**	1.63	
News Popularity	LR	**0.19**	**0.20**	**0.27**	**0.27**	0.53	0.79	0.95	0.41
	NN	**0.17**	**0.19**	**0.19**	**0.27**	0.60	0.75	0.88	
	GDIA	**0.13**	**0.13**	**0.17**	**0.22**	**0.39**	0.84	1.13	
Wine Quality	LR	**0.10**	**0.16**	**0.21**	**0.34**	**0.43**	**0.61**	1.02	0.97
	NN	**0.14**	**0.37**	**0.44**	**0.57**	**0.78**	0.99	1.12	
	GDIA	**0.08**	**0.15**	**0.20**	**0.37**	**0.55**	**0.73**	1.08	
Synthetic-1	LR	**0.49**	**0.62**	**0.63**	**0.64**	**0.67**	**0.68**	**0.69**	0.88
	NN	**0.28**	**0.34**	**0.35**	**0.46**	0.88	0.90	0.91	
	GDIA	**0.54**	**0.78**	**0.82**	**0.83**	**0.84**	0.88	0.90	
Synthetic-2	LR	**0.27**	**0.36**	**0.39**	**0.40**	**0.41**	**0.41**	**0.43**	0.71
	NN	**0.26**	**0.30**	**0.32**	**0.33**	0.74	0.81	0.89	
	GDIA	**0.38**	**0.42**	**0.46**	**0.47**	**0.48**	**0.48**	**0.49**	

A smaller value indicates a better attack result.
The successful attack results (Wasserstein distance lower than random guess) are highlighted in bold.

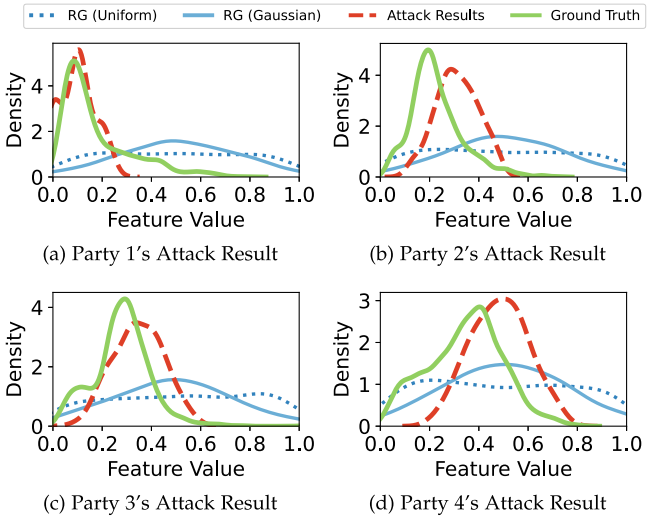


Fig. 6. Visualization of the attack results on the "height" feature of the Abalone dataset. Each subfigure represents the attack results on an FL party. All the curves are probability density functions of the feature value.

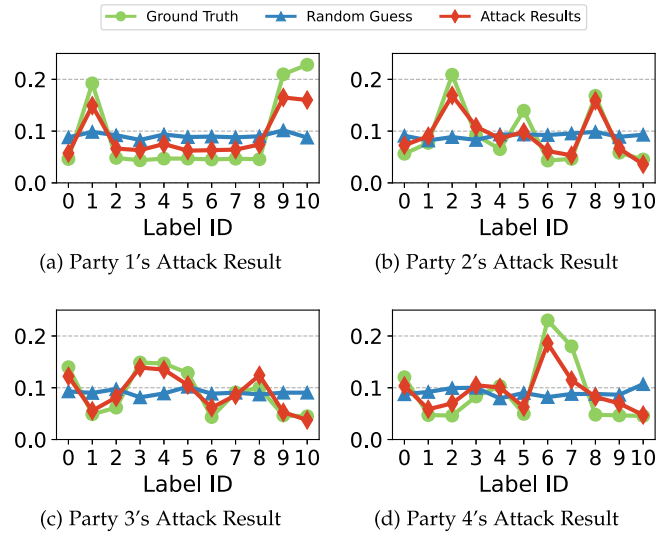


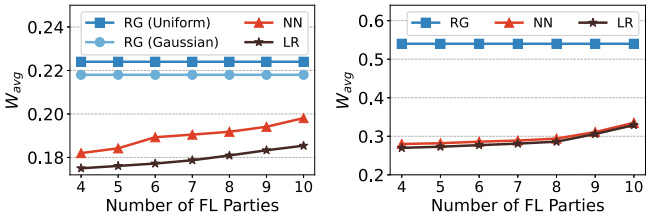
Fig. 7. Visualization of the attack results on label distribution of the Drive Diagnosis dataset. Models are trained on Drive Diagnosis dataset. Each subfigure represents the attack results on an FL party. The x-axis refers to label ID (from 0 to 10), and the y-axis refers to the proportion of each label.

label distribution attack results in Fig. 7 are obtained by first performing GDIA against LR models (trained with $\epsilon = 4$), and then launching the label distribution attack.

The results demonstrate that the proposed attack can successfully infer the label distribution of each party as long as the DP mechanism does not impair the model utility significantly (i.e., $\epsilon \geq 2$). From Table III we can also observe that, different privacy budgets ϵ have significant impacts on the results of the label distribution attack. In particular, the success rate of the label distribution attack drops rapidly as the ϵ decreases. The reason is that the label distribution attack relies on the accuracy of models. As described in Algorithm 4, we label the synthetic samples based on model outputs. Therefore, a small ϵ can damage the model accuracy and lead to inaccurate labels, which reduce the attack performance accordingly.

E. Effect of Number of Parties on the Attacks

We vary the number of FL parties from 4 to 10 and evaluate our attacks. Specifically, we divide the Abalone dataset into different numbers of local datasets, and perform the attacks with a fixed



(a) Feature Distribution Attacks (b) Label Distribution Attacks

Fig. 8. Attack results with different numbers of FL parties.

TABLE IV
FEATURE DISTRIBUTION ATTACK WITH DIFFERENT NUMBER OF USEFUL FEATURES

Number of Useful Features	5	25	50	75
LR	0.132	0.095	0.081	0.061
NN	0.117	0.080	0.059	0.057
GDIA	0.138	0.097	0.089	0.085
RG (Gaussian)	0.093	0.104	0.123	0.131
RG (Uniform)	0.162	0.163	0.163	0.164

TABLE V
LABEL DISTRIBUTION ATTACK WITH DIFFERENT NUMBER OF USEFUL FEATURES

Number of Useful Features	5	25	50	75
LR	0.778	0.704	0.446	0.364
NN	0.791	0.635	0.364	0.328
GDIA	0.782	0.716	0.513	0.431
Random Guess	0.784	0.787	0.796	0.778

privacy budget $\epsilon = 4$. Fig. 8 shows the attack results, from which we can observe that the attack performance slightly decreases as the number of parties increases. This is because the size of the local dataset becomes smaller when the number of parties goes higher. Given a fixed privacy budget, the smaller the dataset size is, the greater influence the individual data samples will impose on the final model parameters, thus the larger noise is required to achieve the same level of privacy, leading to worse attack performance.

F. Effect of Number of Useful Features on the Attacks

We evaluate the impact of the number of useful features on our attacks using synthetic datasets. Specifically, we set the number of records to 200,000, the number of classes to 5, and the number of features to 100. The number of useful features varies from $\{5, 25, 50, 75\}$. Additionally, we maintain a constant privacy budget ϵ at 4 for all experiments. The results of our feature and label distribution attacks are depicted in Tables IV and V, respectively. It is observed that the number of useful features significantly influences the success of our attack. When the number of useful features is moderate (e.g., more than 25), the attacker can effectively infer private information about feature and label distributions. However, with a very small number of useful features (e.g., only 5), the attacker obtains little useful information. This is because the non-useful features, generated randomly, do not correlate meaningfully with the labels, resulting in little impact on the model's output. Consequently, our attack methods are less effective in inferring data distributions of these non-useful features by comparing the outputs between local and global models.

TABLE VI
FEATURE DISTRIBUTION ATTACK WITH DIFFERENT NUMBER OF CLASSES

Number of Classes	2	4	6	8
LR	0.099	0.068	0.054	0.051
NN	0.103	0.062	0.056	0.043
GDIA	0.101	0.087	0.083	0.080
RG (Gaussian)	0.133	0.134	0.134	0.132
RG (Uniform)	0.163	0.164	0.161	0.164

TABLE VII
LABEL DISTRIBUTION ATTACK WITH DIFFERENT NUMBER OF CLASSES

Number of Classes	2	4	6	8
LR	1.541	0.508	0.416	0.310
NN	1.517	0.419	0.388	0.305
GDIA	1.585	0.643	0.543	0.422
Random Guess	1.937	1.030	0.831	0.752

G. Effect of Number of Classes on the Attacks

We evaluate the effect of the number of classes on our attacks using synthetic datasets. Specifically, we set the number of records to 200,000, the number of features to 100, the proportion of useful features to 70%, and varied the number of classes among $\{2, 4, 6, 8\}$. The evaluation results for feature and label distribution attacks are presented in Tables VI and VII, respectively. We observe that the attack results on binary classification datasets are less effective compared to datasets with a larger number of classes (e.g., 4 or more), yet they are still more effective than random guessing. These results indicate that our attacks reveal less information about the training data distribution in binary classification datasets compared to datasets with more classes. This is attributed to our attacks' reliance on comparing the output vectors of local and global models. In binary classification, the output vector's smaller dimension increases the likelihood of the global and local models producing similar predictions for some randomly-generated out-of-distribution features. Conversely, in datasets with more classes, the generation of such false-positive samples is less likely, leading to more accurate attacks.

H. Evaluation of Party-Side Attacks

We evaluate party-side feature and label distribution inference attacks on LR models trained on Abalone dataset. The attack results are obtained by first performing PGDA and then launching the label distribution attack. Specifically, we examined the following two cases:

i) Party 1 attacks Party 2, 3 and 4.
ii) Party 1 colludes with Party 2 to attack Party 3 and 4.

Tables VIII and IX report the attack results, from which we observe that party-side attacks are worse than server-side attacks, as the adversary has less information than the server. Nevertheless, party-side attacks still obtain some distribution information about other parties (i.e., better than random guess), and the performance of the attack increases with the number of colluding parties.

TABLE VIII
PARTY-SIDE FEATURE DISTRIBUTION ATTACK

ϵ	1	2	4	8
Server-side	0.182	0.178	0.174	0.172
Party-side Case (i)	0.213	0.205	0.198	0.193
Party-side Case (ii)	0.209	0.201	0.190	0.187
RG (Gaussian)	0.218			
RG (Uniform)	0.224			

TABLE IX
PARTY-SIDE LABEL DISTRIBUTION ATTACK

ϵ	1	2	4	8
Server-side	0.401	0.292	0.272	0.264
Party-side Case (i)	0.494	0.403	0.389	0.371
Party-side Case (ii)	0.429	0.341	0.354	0.335
Random Guess	0.542			

TABLE X
FEATURE DISTRIBUTION ATTACK UNDER INCONSISTENT PRIVACY BUDGETS

ϵ	1	2	4	8
Server-side	0.188	0.173	0.171	0.170
Party-side	0.210	0.197	0.188	0.183
RG (Gaussian)	0.218			
RG (Uniform)	0.224			

TABLE XI
LABEL DISTRIBUTION ATTACK UNDER INCONSISTENT PRIVACY BUDGETS

ϵ	1	2	4	8
Server-side	0.385	0.287	0.277	0.271
Party-side	0.418	0.351	0.350	0.323
Random Guess	0.542			

In scenarios where multiple parties collude to attack others, the colluding parties may opt for a larger DP budget, thereby injecting less DP noise into their local models to improve the global model's utility. Since our attacks partially rely on global model predictions, this approach might enhance attack performance. We evaluated this strategy on the Abalone dataset under the second party-side attack setting. Specifically, we did not add any DP noise to the local models of Parties 1 and 2, while varying the privacy budgets of the other parties, and launched attacks from both the server's and the colluding parties' sides. Tables X and XI present the evaluation results for feature and label distribution attacks. Unfortunately, we did not observe a significant improvement in attack performance using this method (compared to the results shown in Tables VIII and IX). This is because our attacks are primarily based on the premise that the global model generalizes local models. When the privacy budget of the attacked parties is small, DP noise significantly distorts the local model parameters, causing the local model to behave randomly, which leads to poorer attack results. On the other hand, when the privacy budgets of the attacked parties are large, the global model already effectively generalizes the local models. Therefore, in both cases, reducing the DP noise in the attackers' models offers little benefit to the attacks.

I. Summary of Results

We summarize the key findings from our experiments as follows:

TABLE XII
WASSERSTEIN DISTANCES OF ATTACK RESULTS ON THE NN MODEL WITH DROPOUT

Dropout p	**PGDA**	**Label Attack**	**Test Accuracy (%)**
0.0	0.184	0.263	44.6
0.1	0.186	0.270	44.2
0.2	0.194	0.281	44.2
0.3	0.199	0.278	42.9
0.4	0.201	0.294	42.1
0.5	0.210	0.295	40.2

- Our attacks are consistently capable of inferring both feature and label distributions of FL parties from their differentially private local models, as long as the model utility is practical.
- The experiments demonstrated that the attack performance slightly decreased as the number of FL parties increased. Moreover, the number of useful features and the number of classes in datasets significantly influenced attack effectiveness. A moderate proportion of useful features (e.g., $> 25\%$) and larger numbers of classes (e.g., ≥ 4) led to more accurate attacks.
- In scenarios where FL parties collaborated to attack others, it was observed that colluding parties could successfully infer data distribution information about other parties.

These findings underscore the necessity of developing advanced defense techniques to protect the training data distribution in cross-silo FL.

VI. POTENTIAL DEFENSES

In this section, we discuss several potential defense mechanisms that may mitigate the distribution inference attacks.

Dropout for Neural Networks: Dropout [55] is a widely used regularization method in deep learning for mitigating the overfitting problem. Several recent studies demonstrate that dropout can mitigate the membership and feature inference attacks [16], [41], [62] because the effectiveness of these inference attacks is partially based on the overfitting nature of ML models. To evaluate the impact of dropout on the distribution inference attacks, we first fix the privacy budget $\epsilon = 8$, then vary the dropout probability p by $\{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$ and train different NN models on the Abalone dataset. Table XII reports the attack results (measured by (5)), from which we observe that the attack performance on both feature and label distribution is barely impacted by the dropout probability p. The reason is that dropout aims to improve the generalization ability by preventing the model from memorizing any specific training records instead of the underlying data distribution, thus making little impact on our distribution inference attacks.

Non-Uniform Sampling for Stochastic Gradient Descent: Stochastic gradient descent (SGD) algorithm samples training records uniformly at random for the gradient computation, thus each sample has an equal chance of contributing gradients to the model update, which will cause the model parameters to be affected by the training data distribution. Therefore, a possible defense mechanism against the proposed attacks is to select data records non-uniformly such that the distribution of the

TABLE XIII
ATTACK RESULTS UNDER OVERSAMPLING

	Uniform sampling	Oversampling
LR Model	0.36	0.76
NN Model	0.29	0.88
Random Guess	0.93	

used training records is different from the distribution of the original dataset. In this way, the attacker can only infer a skewed distribution instead of the ground truth distribution.

Oversampling is a practical approach to achieving the non-uniform sampling, which is initially used to address the problem of label bias in the training dataset via increasing the sampling probability of training records with minority labels. To evaluate the impact of oversampling on the proposed attacks, we modify the sampling probability of training records with minority labels such that all labels are evenly distributed in the sampled training records for SGD. We perform the label distribution inference attack on the LR and NN models that are trained with oversampling on the Drive Diagnosis dataset. Table XIII reports the experimental results. Compared with the original training method, the attack result (measured by (5)) increases from 0.22 to 0.76 for LR model; for NN model, the attack result increases from 0.27 to 0.88. The evaluation results indicate that oversampling can greatly mitigate the label distribution attack.

Nevertheless, there are three issues when employing oversampling in practical FL applications. First, parties' local datasets may contain only partial labels [1], [3], [6]. In this case, parties cannot increase the proportion of training records with labels that they do not have; thus, the adversary may still infer whether these parties have the records with specific labels or not. Second, oversampling may cause model overfitting since the sampling rate of records with minority labels is increased. Third, the existing solutions for differentially private SGD are based on the sampled Gaussian mechanism [19], [63], [64] with a basic theoretical assumption that each training record is sampled with the same probability by Poisson sampling. If the training records are sampled non-uniformly, then the theoretical privacy guarantee of differentially private SGD does not hold anymore. Therefore, if we are to adopt oversampling as a defense method, we need to design advanced DP mechanisms that are compatible with oversampling.

Hiding Local Models: In the FL systems with MPC [43], [59], [65], [66], the local models are first encrypted and then shared with the server, and the server can only obtain a plain-text version of the aggregated global model. These systems may reasonably mitigate our attacks but fail to provide the protection of DP, and are thus vulnerable to other attacks, e.g., model inversion and membership inference. Naturally, our attacks, which reveal the privacy risks in DP-based FL solutions, provide a strong motivation to *incorporate both DP and MPC into FL.*

Nevertheless, deploying MPC into FL systems still has two practical issues. The first issue is that the systems may lack support for or require extra computational and communication costs for important mechanisms, such as fairness [67], incentive [68], and robustness [38], [44], [45], [46]. Notably, many existing poisoning attack defense mechanisms require the FL server to access and analyze each party's local models [38], [44], [45], [46]. Without plain-text local models, the FL system can hardly verify the honesty of the parties, e.g., whether they provide meaningless datasets or conduct active attacks (e.g., backdoor or poisoning attacks [69]). The second issue is that efficiently incorporating DP into FL when using MPC is a highly non-trivial task. Existing MPC protocols primarily support finite field operations because modular arithmetic is used as a basic cryptographic primitive [43], [70]. Consequently, MPC protocols are incompatible with popular DP model training algorithms that require sampling real-valued DP noises from the continuous Gaussian distribution (e.g., DP-SGD [19]). Note that the aggregated global models can still leak membership information about training records [3], thus it is still necessary to train local models by DP algorithms even when MPC is used for global model aggregation. However, only a few works [20], [71], [72] have so far explored this issue, and designing efficient DP model training mechanisms compatible with MPC remains a challenging and under-explored problem.

In conclusion, MPC is a powerful defense method against our attacks, as it conceals local model parameters. Developing advanced privacy-preserving FL frameworks that (i) can efficiently combine DP with MPC, and (ii) support important mechanisms in FL (e.g., fairness, incentive, and robustness) in an efficient way would be an interesting and promising future direction.

VII. RELATED WORK

Privacy Leakage in FL: It has been reported that ML model parameters can memorize sensitive information of the training datasets [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16]. Therefore, it is possible for an adversary to infer sensitive training data from each FL party's local model. In the FL setting, recent studies have pointed to the potential privacy risks from publishing local models. Hitaj et al. [6] demonstrate the class representation leakage in FL. Their attack method enables a malicious party to infer the representation of an unseen class. Melis et al. [13] propose a property inference attack in FL that allows a malicious party to infer certain properties of other parties' training data. Gao et al. [34] propose a category inference attack that allows the FL server to infer which classes of data a FL party has. In addition, gradient inversion attacks [8], [9] demonstrate that the training data can be recovered precisely from the parties' submitted gradients.

Nevertheless, the assumptions of these attacks are relatively strong and impractical in the cross-silo FL setting. *First*, the adversary in model inversion attacks [15] requires the model output of a certain input record to reconstruct the corresponding record representation, which is not a realistic assumption in FL. *Second*, the success rate of membership inference attacks [16], [73] can be theoretically bounded by differential privacy (DP) [16], [17], [18], [19], [31]. *Third*, the existing property [13], [74], [75], category [34], and class representation attacks [6] rely heavily on sensitive auxiliary datasets, which can hardly be collected by the adversary in advance [33]. In addition, these attacks only focus on *coarse-grained* distribution information such as binary

properties [13], [74] or the presence of specific labels [34], which discloses less privacy information compared to the attacks we propose. *Fourth*, the gradient inversion attacks [8], [9] make strong assumptions about the FL training process, i.e., (i) the FL parties submit the gradients computed by a batch of training records instead of the local model parameters trained by multiple local iterations, (ii) the gradients are computed based on a small batch of training records, and (iii) the corresponding private labels are available to the adversary. However, these assumptions do not hold for the FedAvg algorithm. A recent study [76] points out that the performance of these attacks drops significantly once these impractical assumptions are removed.

On the contrary, we consider a more stringent and realistic setting where no auxiliary datasets or prior knowledge of training data are known to the adversary. Moreover, our attacks can infer *fine-grained* distribution information of *both features and labels*, revealing a much more serious privacy leakage issue in DP-based cross-silo FL solutions.

Privacy-Preserving FL: To further enhance the privacy protection level of FL, differential privacy (DP) has been incorporated into the FL training to perturb the local model parameters. The DP used in FL can be further classified into *record-level* DP and *party-level* DP. In the cross-silo FL setting, record-level DP [14], [17], [18], [19], [20], which, by definition, aims to hide the information of whether a data record (membership) participates in the model training, is commonly used to protect data privacy. On the other hand, party-level DP, which aims to hide the information of whether a party participates in the FL training, is mostly considered in the cross-device FL setting [77]. It is worth noting that party-level DP is unsuitable for cross-silo FL because it requires thousands of parties [13], [77] to participate in the training process to converge the global model, due to the large magnitude of party-level DP noise.

Another approach to protect data privacy in FL is concealing local model parameters from the server during the training process. Trusted hardware such as Intel SGX [78] safeguards private data in FL through secure enclaves [79], but it is vulnerable to side-channel attacks [80]. MPC offers a strong privacy guarantee [43], [59], [65], [66], [70], [81] and is therefore a powerful defense method against our attacks. Nevertheless, as shown in Section VI, deploying MPC into FL systems has several practical issues. Our work provides a strong motivation to develop advanced privacy-preserving frameworks that (i) support important FL mechanisms such as defense mechanisms against poisoning attacks, and (ii) effectively incorporate both DP and MPC into FL.

VIII. Conclusion

In this article, we propose a set of data distribution inference attacks against cross-silo FL, which reveals a new privacy risk in FL that cannot be mitigated by DP effectively. Our attacks demonstrate that an honest-but-curious server can infer both the feature and label distribution of each party's training data without using any auxiliary dataset. To the best of our knowledge, this is the first work that studies the problem of data distribution leakage in the context of FL and/or DP. Extensive experiments verify the effectiveness of the proposed attacks. We further discuss and evaluate several potential defenses against the proposed attacks. Our work highlights the necessity of developing more advanced privacy-preserving frameworks for protecting data distribution in real-world applications.

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 *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[2] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019.

[3] P. Kairouz et al., "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, no. 1/2, pp. 1–210, 2019.

[4] Q. Li et al., "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 3347–3366, Apr. 2023. [Online]. Available: http://sites.computer.org/debull/A23mar/p9.pdf

[5] Y. Tong et al., "Federated computing: Query, learning, and beyond," *IEEE Data Eng. Bull.*, vol. 46, no. 1, pp. 9–26, 2023.

[6] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: Information leakage from collaborative deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 603–618.

[7] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 2512–2520.

[8] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 14774–14784.

[9] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients - how easy is it to break privacy in federated learning?," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 16937–16947.

[10] Z. Zhang, Q. Liu, Z. Huang, H. Wang, C.-K. Lee, and E. Chen, "Model inversion attacks against graph neural networks," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 9, pp. 8729–8741, Sep. 2023.

[11] S. Zhang, W. Yuan, and H. Yin, "Comprehensive privacy analysis on federated recommender system against attribute inference attacks," *IEEE Trans. Knowl. Data Eng.*, early access, Jul. 14, 2023, doi: 10.1109/TKDE.2023.3295601.

[12] Y. Yang, Z. Ma, B. Xiao, Y. Liu, T. Li, and J. Zhang, "Reveal your images: Gradient leakage attack against unbiased sampling-based secure aggregation," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 12, pp. 12958–12971, Dec. 2023.

[13] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 691–706.

[14] F. Wang, E. Hugh, and B. Li, "More than enough is too much: Adaptive defenses against gradient leakage in production federated learning," in *Proc. IEEE Conf. Comput. Commun.*, 2023, pp. 1–10.

[15] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1322–1333.

[16] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 3–18.

[17] S. Truex, L. Liu, K.-H. Chow, M. E. Gursoy, and W. Wei, "LDP-FED: Federated learning with local differential privacy," in *Proc. 3rd ACM Int. Workshop Edge Syst. Anal. Netw.*2020, pp. 61–66.

[18] N. Wu, F. Farokhi, D. Smith, and M. A. Kaafar, "The value of collaboration in convex machine learning with differential privacy," in *Proc. IEEE Symp. Secur. Privacy*, 2020, pp. 304–317.

[19] M. Abadi et al., "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 308–318.

[20] E. Bao et al., "Skellam mixture mechanism: A novel approach to federated learning with differential privacy," *Proc. VLDB Endowment*, vol. 15, 2022, pp. 2348–2360.

[21] M. Naseri, J. Hayes, and E. De Cristofaro, "Local and central differential privacy for robustness and privacy in federated learning," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2022, pp. 1–18.

[22] L. Zhang, T. Zhu, P. Xiong, W. Zhou, and P. Yu, "A robust game-theoretical federated learning framework with joint differential privacy," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 3333–3346, Apr. 2023.

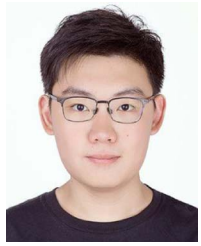
[23] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. Theory Cryptogr. Conf.*, 2006, pp. 265–284.
[24] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation," in *Proc. 24th Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2006, pp. 486–503.
[25] X. Xiao, G. Wang, and J. Gehrke, "Differential privacy via wavelet transforms," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 8, pp. 1200–1214, Aug. 2011.
[26] T. Zhu, G. Li, W. Zhou, and S. Y. Philip, "Differentially private data publishing and analysis: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 8, pp. 1619–1638, Aug. 2017.
[27] H. Shin, S. Kim, J. Shin, and X. Xiao, "Privacy enhanced matrix factorization for recommendation with local differential privacy," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 9, pp. 1770–1782, Sep. 2018.
[28] D. Kifer and A. Machanavajjhala, "Pufferfish: A framework for mathematical privacy definitions," *ACM Trans. Database Syst.*, vol. 39, no. 1, pp. 1–36, 2014.
[29] G. Cormode, "Personal privacy vs population privacy: Learning to attack anonymization," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2011, pp. 1253–1261.
[30] J. Zhou, Y. Chen, C. Shen, and Y. Zhang, "Property inference attacks against GANs," in *Proc. 30th Netw. Distrib. Syst. Secur. Symp.*, 2022, pp. 1–17.
[31] B. Balle, G. Cherubin, and J. Hayes, "Reconstructing training data with informed adversaries," in *Proc. IEEE Symp. Secur. Privacy*, 2022, pp. 1138–1156.
[32] C. Guo, B. Karrer, K. Chaudhuri, and L. van der Maaten, "Bounding training data reconstruction in private (deep) learning," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 8056–8071.
[33] Z. Li and Y. Zhang, "Membership leakage in label-only exposures," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2021, pp. 880–895.
[34] J. Gao et al., "Secure aggregation is insecure: Category inference attack on federated learning," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 1, pp. 147–160, Jan./Feb. 2023.
[35] A. A. Goldstein, "Convex programming in Hilbert space," *Bull. Amer. Math. Soc.*, vol. 70, no. 5, pp. 709–710, 1964.
[36] S. J. Russell and P. Norvig, *Artificial Intelligence—A Modern Approach*, 3rd ed. London, U.K.: Pearson Education, 2010.
[37] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–26.
[38] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2021, pp. 1–18.
[39] C. Dwork et al., "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, no. 3/4, pp. 211–407, 2014.
[40] X. Luo, X. Xiao, Y. Wu, J. Liu, and B. C. Ooi, "A fusion-denoising attack on instahide with data augmentation," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 1899–1907.
[41] X. Luo, Y. Wu, X. Xiao, and B. C. Ooi, "Feature inference attack on model predictions in vertical federated learning," in *Proc. Int. Conf. Data Eng.*, 2021, pp. 181–192.
[42] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 2938–2948.
[43] K. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1175–1191.
[44] X. Cao, J. Jia, Z. Zhang, and N. Z. Gong, "Fedrecover: Recovering from poisoning attacks in federated learning using historical information," in *Proc. IEEE Symp. Secur. Privacy*, 2023, pp. 326–343.
[45] Z. Zhang, X. Cao, J. Jia, and N. Z. Gong, "FLDetector: Defending federated learning against model poisoning attacks via detecting malicious clients," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2022, pp. 2545–2555.
[46] Y. Jiang, Y. Zhou, D. Wu, C. Li, and Y. Wang, "On the detection of shilling attacks in federated collaborative filtering," in *Proc. Int. Symp. Reliable Distrib. Syst.*, 2020, pp. 185–194.
[47] C. Fu et al., "Label inference attacks against vertical federated learning," in *Proc. USENIX Secur. Symp.*, 2022, pp. 1397–1414.
[48] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml
[49] I.-C. Yeh and C.-h. Lien, "The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients," *Expert Syst. Appl.*, vol. 36, no. 2, pp. 2473–2480, 2009.
[50] K. Fernandes, P. Vinagre, and P. Cortez, "A proactive intelligent decision support system for predicting the popularity of online news," in *Proc. Portuguese Conf. Artif. Intell.*, 2015, pp. 535–546.
[51] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decis. Support Syst.*, vol. 47, no. 4, pp. 547–553, 2009.
[52] X. Luo, Y. Jiang, and X. Xiao, "Feature inference attack on shapley values," in *Proc. Conf. Comput. Commun. Secur.*, 2022, pp. 2233–2247.
[53] Y. Wang, Y. Tong, D. Shi, and K. Xu, "An efficient approach for cross-silo federated learning to rank," in *Proc. IEEE 37th Int. Conf. Data Eng.*, 2021, pp. 1128–1139.
[54] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-IID data silos: An experimental study," in *Proc. IEEE 37th Int. Conf. Data Eng.*, 2022, pp. 965–978.
[55] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[56] C. Villani, *Optimal Transport: Old and New*. Berlin, Germany: Springer, 2009.
[57] B. Jayaraman and D. Evans, "Evaluating differentially private machine learning in practice," in *Proc. USENIX Secur.*, 2019, pp. 1895–1912.
[58] Q. Li, Z. Wen, and B. He, "Practical federated gradient boosting decision trees," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 4642–4649.
[59] Y. Wu, S. Cai, X. Xiao, G. Chen, and B. C. Ooi, "Privacy preserving vertical federated learning for tree-based models," *Proc. VLDB Endowment*, vol. 13, pp. 2090–2103, 2020.
[60] L. Zhao et al., "Inprivate digging: Enabling tree-based distributed data mining with differential privacy," in *Proc. Conf. Comput. Commun.*, 2018, pp. 2087–2095.
[61] Q. Li, Z. Wu, Z. Wen, and B. He, "Privacy-preserving gradient boosting decision trees," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 784–791.
[62] A. Salem, Y. Zhang, M. Humbert, M. Fritz, and M. Backes, "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2019, pp. 1–15.
[63] I. Mironov, K. Talwar, and L. Zhang, "Rényi differential privacy of the sampled Gaussian mechanism," 2019, *arXiv:1908.10530*.
[64] Y. Zhu and Y.-X. Wang, "Poission subsampled rényi differential privacy," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7634–7642.
[65] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning," in *Proc. USENIX Annu. Tech. Conf.*, 2020, pp. 493–506.
[66] Y. Zhang et al., "Efficient and secure skyline queries over vertical data federation," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 9, pp. 9269–9280, Sep. 2023.
[67] T. Wang, J. Rausch, C. Zhang, R. Jia, and D. Song, "A principled approach to data valuation for federated learning," in *Federated Learning*. Berlin, Germany: Springer, 2020, pp. 153–167.
[68] H. Yu et al., "A fairness-aware incentive scheme for federated learning," in *Proc. AAAI/ACM Conf. AI, Ethics, Soc.*, 2020, pp. 393–399.
[69] M. Fang, X. Cao, J. Jia, and N. Z. Gong, "Local model poisoning attacks to Byzantine-robust federated learning," in *Proc. USENIX Secur.*, 2020, pp. 1623–1640.
[70] A. C. Yao, "Protocols for secure computations," in *Proc. 23rd Annu. Symp. Found. Comput. Sci.*, 1982, pp. 160–164.
[71] P. Kairouz, Z. Liu, and T. Steinke, "The distributed discrete gaussian mechanism for federated learning with secure aggregation," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 5201–5212.
[72] N. Agarwal, P. Kairouz, and Z. Liu, "The skellam mechanism for differentially private federated learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 5052–5064.
[73] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *Proc. IEEE 31st Comput. Secur. Found. Symp.*, 2018, pp. 268–282.
[74] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov, "Property inference attacks on fully connected neural networks using permutation invariant representations," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 619–633.
[75] A. Suri and D. Evans, "Formalizing and estimating distribution inference risks," in *Proc. Privacy Enhancing Technol. Symp.*, 2022, pp. 528–551.
[76] Y. Huang, S. Gupta, Z. Song, K. Li, and S. Arora, "Evaluating gradient inversion attacks and defenses in federated learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 7232–7241.

[77] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–14.
[78] F. McKeen et al., "Innovative instructions and software model for isolated execution," *HASP*, vol. 10, no. 1, 2013, pp. 1–8.
[79] O. Ohrimenko et al., "Oblivious multi-party machine learning on trusted processors," in *Proc. USENIX Secur.*, 2016, pp. 10–12.
[80] Y. Xu, W. Cui, and M. Peinado, "Controlled-channel attacks: Deterministic side channels for untrusted operating systems," in *Proc. IEEE Symp. Secur. Privacy*, 2015, pp. 640–656.
[81] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, "Secure single-server aggregation with (poly) logarithmic overhead," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2020, pp. 1253–1269.

Yangfan Jiang received the BEng degree in computer science from Sun Yat-sen University, Guangzhou, China, in 2021. He is currently working toward the PhD degree with the Department of Computer Science, National University of Singapore, Singapore. His research interests include data privacy and machine learning security.

Xinjian Luo received the MS degree in computer science from Shanghai Jiao Tong University, Shanghai, China, in 2019. He is currently working toward the PhD degree with the Department of Computer Science, National University of Singapore, Singapore. His research focuses on privacy and security issues in machine learning applications.

Yuncheng Wu received the BSc degree from the Southwest University of Science and Technology, China, in 2010, the MSc degree from the Nanjing University of Science and Technology, Nanjing, China, in 2013, and the PhD degree from the Renmin University of China, Beijing, China, in 2018. He is currently a research assistant professor with the School of Computing, National University of Singapore, Singapore. His research interests include data security and privacy, federated learning, and data collaboration.



Xiaochen Zhu received the BComp degree in computer science and the BS degree in mathematics from the National University of Singapore, Singapore, in 2023. He is currently a research assistant with the National University of Singapore. His research interests include data privacy and machine learning.

Xiaokui Xiao (Fellow, IEEE) received the PhD degree in computer science from the Chinese University of Hong Kong. He is currently a professor with the School of Computing, National University of Singapore (NUS), Singapore. Before joining NUS, he was an associate professor with the Nanyang Technological University, Singapore. His research interests include data management, especially on data privacy and algorithms for large data. He is an ACM Distinguished Member. He was the recipient of the Best Research Paper Award in VLDB 2021, and the ACM SIGMOD Research Highlight Award in 2022.

Beng Chin Ooi (Fellow, IEEE) is currently the Lee Kong Chian Centennial professor and an NGS faculty member with the National University of Singapore, Singapore. His research interests include database system architectures, performance issues, indexing techniques, and query processing, in the context of multimedia, spatio-temporal, distributed, parallel, blockchain, and in-memory systems. He was an editor-in-chief of *IEEE Transactions on Knowledge and Data Engineering* during 2009–2012, a trustee board member, and the president of the VLDB Endowment during 2014–2017. He is a fellow of ACM and Singapore National Academy of Science.