Federated Learning for Tabular Data

Wu, Han; Zhao, Zilong; Chen, Lydia Y.; van Moorsel, Aad

DOI:
10.1109/ISSRE55969.2022.00028

Federated Learning for Tabular Data: Exploring Potential Risk to Privacy

Han Wu*
School of Computing
Newcastle University
Newcastle upon Tyne, UK
han.wu@ncl.ac.uk

Zilong Zhao*
Department of Computer Science
Delft University of Technology
Delft, Netherlands
Z.Zhao-8@tudelft.nl

Lydia Y. Chen
Department of Computer Science
Delft University of Technology
Delft, Netherlands
lydiaychen@ieee.org

Aad van Moorsel
School of Computer Science
University of Birmingham
Birmingham, UK
a.vanmoorsel@bham.ac.uk

***Abstract*—Federated Learning (FL) has emerged as a potentially powerful privacy-preserving machine learning methodology, since it avoids exchanging data between participants, but instead exchanges model parameters. FL has traditionally been applied to image, voice and similar data, but recently it has started to draw attention from domains including financial services where the data is predominantly tabular. However, the work on tabular data has not yet considered potential attacks, in particular attacks using Generative Adversarial Networks (GANs), which have been successfully applied to FL for non-tabular data. This paper is the first to explore leakage of private data in Federated Learning systems that process tabular data. We design a Generative Adversarial Networks (GANs)-based attack model which can be deployed on a malicious client to reconstruct data and its properties from other participants. As a side-effect of considering tabular data, we are able to statistically assess the efficacy of the attack (without relying on human observation such as done for FL for images). We implement our attack model in a recently developed generic FL software framework for tabular data processing. The experimental results demonstrate the effectiveness of the proposed attack model, thus suggesting that further research is required to counter GAN-based privacy attacks.**

***Index Terms*—Federated Learning, GAN, Privacy, Tabular Data**

I. Introduction

Federated Learning (FL), or collaborative learning in some literature, is an emerging paradigm for machine learning models, specifically useful to maintain privacy for sensitive personal information [1]. FL enables multiple clients (e.g., end users, companies, institutes) to cooperatively train a machine learning model without exposing their sensitive data (e.g. customer identifiable information, healthcare records) [2]. During the training process, each client iteratively trains a sub-model using its local data and exchanges only the parameters of the sub-model with a parameter server to construct a global model. Clearly, this potentially alleviates or at least reduces the privacy risks associated with traditional, centralised, machine learning that rely on data sharing. Compelling use cases of FL reported in the literature include a risk management application for small and micro enterprise loans [3], an edge computing platform for fire detection [4], and an anti-money laundering system for banks [5].

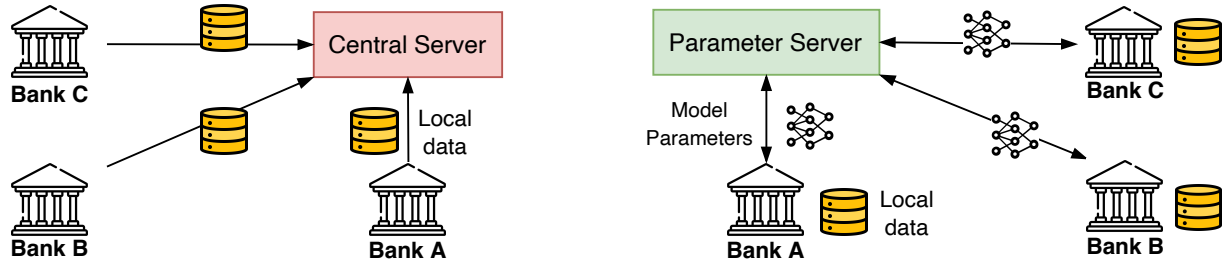
*Equal contribution

Despite the fact that in FL data itself is not exchanged, the risk to privacy is not completely eliminated. Recent attack models on FL have managed to reveal sensitive information of the training data by studying the model parameters exchanged [6]–[10]. These attacks have been performed on image processing models, for instance, the attack model runs on a malicious parameter server in [10] to reconstruct the persons' images owned by a specific client. To evaluate the efficacy of such attack, one subjectively judges whether the image generated by the attack model is close to the target one [6].

In many application areas, such as financial services, data does not come in the shape of images, but is *tabular data*, that is, data consisting of information and values structured in rows and columns (such as in spreadsheets). A typical example is a table of customer records, and in such cases, tabular data will typically contain sensitive information about individuals, such as income and marital status. In recent years, researchers have started to apply FL to tabular data, mostly focusing on improving performance [2], [11], [12]. However, given the sensitive nature of much tabular data, it is essential to consider privacy implications of FL when applied to tabular data.

In this paper, we explore if it is possible to infer information about the collective data of the various participants based solely on the exchange of machine learning model information. Particularly, our attack model assumes one of the participants to be malicious, called *adversary*, aiming to infer collective data properties about some data classes. We call this a *class property inference attack*. The adversary adopts advanced data synthesising technology, Generative Adversarial Networks (GANs) [13], to construct samples of the target class. Through these samples the adversary infers the statistical property of the target class, i.e., the distributions of some attributes.

Despite the great success that GANs have achieved in image processing [14]–[16], GANs for tabular data synthesis are still in the preliminary stage of development [17]–[19]. Therefore, attacks against FL for tabular data have not been considered yet in the literature. In this paper, we therefore propose a tabular GAN-based privacy attack approach against FL systems. Our attack approach is inspired by GAN-based attacks on image data such as studied by [20] and [9], but with



(a) Centralised Learning (b) Federated Learning

Fig. 1: Conventional Centralised Machine Learning and Federated Learning.

some differences: (i) Our attack model runs on a malicious client, while [9] assumes the parameter server to be malicious; (ii) [20] assumes that the adversary can change the architecture of the global model (e.g., number of neurons), which is not realistic and we deprecate this assumption in this paper; (iii) [20] aims to reconstruct the class of images that look similar, while our attack focuses on inferring statistic characteristics of the specified class and we use quantitative methods to evaluate the privacy risk, which is missing in [20].

To demonstrate the effectiveness of our attack, we perform experiments on datasets that are not Independent and Identically Distributed (Non-IID). Non-IID means the data distributions of participating FL clients differ from each other or are dependent. The Non-IID setting is particularly susceptible to privacy leakage, as we will see. We introduce distance metrics to quantitatively measure the statistical similarity between the synthetic samples and target ones. In this sense, tabular data allows for statistically more powerful assessment of the success of attacks than image data, which relies on subjective similarity assessment [6] using the human eye.

The results of our experiments show that an adversary is able to infer considerable information about potentially sensitive data properties. In tabular data for a finance scenario, such data leakage could for instance pertain to income and marital status of customers associated with the target class. We also compare our GAN-based attack with the use of GANs for synthetic data generation. Interestingly, an unexpected outcome of our experiments is that for certain properties associated with a data class, the data generated in our GAN-based attack is more similar to the real data than that generated by state-of-the-art synthetic tabular data generators. This is particularly the case for data features that most influence the classification.

In summary, the main contributions of this paper are:

- To the best of our knowledge, we are the first to explore privacy risks in FL for tabular data. Related GAN-based attacks have mainly focused on recovering image data and the approaches are not directly applicable for tabular data.
- We propose a class property inference attack on tabular data classification models in FL, where the adversary infers the property of the target class. Then we use similarity metrics to evaluate the seriousness of such private information leakage.
- We conduct extensive experimental evaluation to assess the efficacy of our attack. On the Bank Loan and Credit datasets, our model successfully infers private information of the target class.

II. Preliminary Knowledge

A. Federated Learning

Throughout this paper, we comprehend the machine learning model as a deterministic function $Y = f(x_1, x_2, \ldots, x_d; \theta)$ parameterised by a set of parameters θ. We work with the supervised learning models for tabular data classification. Specifically, the input is a d-dimensional feature vector $(x_1, x_2, \ldots, x_d)$ such as the profile record of a customer (e.g., age, gender, income). The output of the model Y has a finite set of labels such as the types of customer's loan status (e.g., positive or negative). The training data is a set of data records in the form of $(x_1, x_2, \ldots, x_d, y)$, in which y is the correct class label of the corresponding features. The objective of model training is finding the optimal set of parameters that fits the training data. In the training process, the model normally starts from randomly selected parameters, then the *loss function* L is computed to evaluate the distance between the model output and the actual labels. We use $L(f(x_1, x_2, \ldots, x_d), y; \theta)$ to denote the loss calculated on the data record $(x_1, x_2, \ldots, x_d, y)$ given the model parameters θ. The model adopts the *optimization function* to iteratively update its parameters, based on the loss computed on a batch of training data records. The training finishes when the parameters remain stable around certain values and the loss is close to the minimum.

Considering the concrete example illustrated in Fig. 1, in which multiple banks establish collaboration on developing a machine learning model that predicts their customers' loan status. We assume such collaboration to be necessary because each bank holds a small set of available customer data and none of the banks is able to train a usable model on its own. In the conventional machine learning approach, all banks upload their local data to a central server for training, as depicted in Fig. 1(a). The central server releases the final model to each of the banks when the training is finished. This is effective but under high privacy risk as the sensitive data is transferred from one place to another.

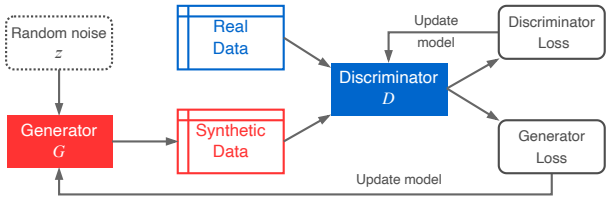


Fig. 2: The general architecture of a GAN.

Federated Learning, introduced by McMahan et al. in [11], is a distributed machine learning framework designed for privacy preservation. Compared to the conventional training methods that collect all data in one place for training, FL allows multiple clients to jointly train a model, while keeping their data stored locally. Fig. 1(b) presents the case of FL paradigm, where each bank trains a model locally and shares only the model parameters. We use θ_i to denote the parameters of the local model f_i on the i th client. As FL training starts, each client trains the local model using the loss L_i computed on its local data, and then uploads its model parameters θ_i to the parameter server. The parameter server aggregates these local models based on the model averaging function:

$$\theta^* = \sum_{i=1}^{K} \omega_i \cdot \theta_i, \tag{1}$$

where K is the total number of clients and ω_i is the *aggregation weight* assigned for the i th client. The clients download the averaged model parameters θ^* from the parameter server to update its local model, and apply it for the next round training. We use $\mathbf{x}$ to denote any input features in the shape of $(x_1, x_2, \ldots, x_d)$. The FL training finishes when the global model, denoted by $f^*(\mathbf{x}; \theta^*)$, converges and reaches a certain accuracy on all clients.

In this paper we work on Horizontal FL, where the tabular data on different clients share the same feature space but have different sample space [21]. For instance, in the tabular dataset held by the banks in Fig. 1, each row corresponds to one particular customer. These banks have different rows of data, i.e., different groups of customers, with the same personal features. The case where the clients have different feature spaces is called Vertical FL [2], [22], which is beyond the scope of this paper.

B. Generative Adversarial Networks

In 2014, Goodfellow et al., for the first time, introduce the Generative Adversarial Networks (GANs) to generate synthetic image samples indistinguishable from the real ones [13]. The training strategy of the GAN is a zero-sum game between two competing deep learning networks. The architecture of this game is depicted in Fig. 2. The generator network $\mathcal{G}$ takes random noise as input to generate synthetic samples, which are fed to the discriminator network $\mathcal{D}$ together with the real samples. $\mathcal{D}$ is trained to distinguish the synthetic samples from the real ones, while $\mathcal{G}$ is trained to fool $\mathcal{D}$. Both real data and synthetic samples are fed into $\mathcal{D}$ and the output is the predicted 'real' or 'synthetic' label of the input data, which is combined with the actual input label to compute the discriminator loss $L_{\mathcal{D}}$. The generator loss $L_{\mathcal{G}}$ is computed to evaluate the similarity between real and synthetic samples. $L_{\mathcal{D}}$ and $L_{\mathcal{G}}$ are applied to update $\mathcal{D}$ and $\mathcal{G}$ respectively.

This game ends when $\mathcal{D}$ is unable to distinguish between the samples from the real training data and the synthetic samples generated by $\mathcal{G}$. The objective of GANs can be summarised as the equation below [13]:

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \mathop{\mathbb{E}}_{x \sim \mathbb{P}_r} [\log \mathcal{D}(x)] + \mathop{\mathbb{E}}_{z \sim \mathbb{P}_z} [\log(1 - \mathcal{D}(\mathcal{G}(z)))] \tag{2}$$

where $\mathbb{P}_r$ denotes the distribution of the real training dataset x, and $\mathbb{P}_z$ is the distribution of the random input z for $\mathcal{G}$. The distribution of the generator's output $\mathcal{G}(z)$ is denoted by $\mathbb{P}_g$. Ideally, the GAN expects to obtain $\mathbb{P}_r = \mathbb{P}_g$ when the training finishes.

III. Related Work

Privacy Attacks on Federated Learning. Privacy attacks on FL can be categorised into insider and outsider attacks according to the sources of attacks [23]. Outsider attacks are those carried out by eavesdroppers on the communication network of FL system, or the users who can access the final trained FL model. In this paper, the discussion of privacy attacks on FL mainly focuses on the insider attacks, which are launched by the FL server or the clients in the FL system.

Membership inference attacks have been extensively studied [8], [24]–[26], in which the attacker aims to infer whether a given data point has been used for training the model. Melis et al. [8] first apply the membership inference attack against FL to infer the presence of exact data points in other clients' training data. The authors also managed to infer the properties of a subset of the training data by using property classifiers.

Under the assumption that the final FL model is accessible, previous reconstruction attacks on Machine Learning model, such as the Model Inversion Attack (MIA), would apply [6], [27]. MIA has been studied to reconstruct a recognisable image of a person, given only access to the trained facial recognition model and the person's name. Hitaj et al. [20] first apply GANs on the malicious client to reconstruct the class representatives of other clients in FL. In [9] the authors assume the parameter server to be the attacker and reconstruct the training data on a specific client. However, only in the special case where all class members are similar, the results of those reconstruction attacks are close to the training data [8]. For instance, all handwritten images of the digit '3' are visually similar, thus the synthetic images of '3' look similar to the real ones [20]. Additionally, the reconstruction attacks mainly focus on image processing models, and the results are just visually measured.

Tabular GANs. Beyond GAN's success in generating images [28], [29], generating realistic synthetic tabular data using GANs has only recently been introduced. For instance, medGAN is proposed in [30] to generate synthetic patient records via a combination of an autoencoder and GANs. Park

et al. [17] propose table-GAN which adopts Convolutional Neural Network (CNN) to synthesise tables in relational databases. By contrast, conditional GAN is designed to generate a specific class of data [14]. CTGAN [18] constructs a specific conditional vector combined with a mechanism *training-by-sampling*. For a chosen discrete column, CTGAN samples training data by log-frequency which largely oversamples the minor category. Zhao et al. design CTAB-GAN [19] and CTAB-GAN+ [31] which can effectively synthesize diverse data types in tabular data, including the mixed data type of continuous and discrete variables and long-tail distribution.

IV. Scenarios and Attack Model

A. Federated Learning Scenario

Our FL scenario follows the framework described in Section II-A, additionally includes some details. We assume that $K (K \geq 2)$ clients agree on a common learning objective and collaboratively train a deep neural network model. The clients reach a consensus on the structure of the neural network model before training starts. We use $\mathcal{T}_i = \{\mathbf{x}_i, \mathbf{y}_i\} (1 \leq i \leq K)$ to denote the tabular data stored locally on client i. The feature $\mathbf{x}_i$ consists of N_c columns with data from continuous variables, namely *continuous columns*, and N_d columns with discrete-valued data, called *discrete columns*. The target column, $\mathbf{y}_i$, is a discrete column that contains the class labels of the rows.

In each round of FL, the client i trains its model locally using $\mathcal{T}_i$ and uploads the model parameters θ_i to the parameter server, which aggregates these parameters according to Equation (1). To simplify our experiments, we assume that the parameter server should collect parameters from all clients before aggregation, while in some work only a fraction of clients is used [21].

B. Non-IID Data

We use $\mathbf{x}_i = \{C_1^i, \ldots, C_{N_c}^i, D_1^i, \ldots, D_{N_d}^i\}$ to denote the features of tabular data on the i th client, where $\{C_1^i, \ldots, C_{N_c}^i\}$ are the continuous columns and $\{D_1^i, \ldots, D_{N_d}^i\}$ are the discrete columns. The values in these columns are considered as random variables that follow an unknown joint distribution $\mathbb{P}_{\mathcal{T}_i} = \{\mathbb{P}(\mathbf{x}_i), \mathbb{P}(\mathbf{y}_i)\}$. We study FL scenarios with data that is not Independent and Identically Distributed (Non-IID), which means $\mathbb{P}_{\mathcal{T}_i}$ differs from client to client [21]. This is close to the real world cases that none of the clients knows the distribution of the overall dataset. Thus collaboration via FL is necessary in order to obtain a usable prediction model.

Particularly, the Non-IID data in our FL scenario is label skewed, that is, the distribution of labels, denoted by $\mathbb{P}(\mathbf{y}_i)$ is imbalanced across clients. For instance, in Section VI we design the case in which one client holds the dataset with 99% negative class and 1% positive class, while the other client holds 90% and 10% respectively.

Studies have shown that compared to centralised machine learning, the performance degradation is almost inevitable for FL processing Non-IID data [11], [32]. In our FL framework, we design a similarity-based aggregation algorithm to compute the aggregation weights assigned for the clients, which mitigates the impact of Non-IID data. We discuss this further in Section VI-B.

C. Malicious Client

Our attack model is actively conducted by a malicious client, the *adversary*, in the FL scenario. Throughout the FL process, the adversary pretends to be an honest client but aims to extract the private information of a specific class, which the adversary is not supposed to know. Note that there can be two cases about the target class: (i) the adversary does not have the data of the target class; (ii) the adversary has only a small amount of the target class records, so the distribution cannot represent the property of the class in the overall dataset. Both cases are studied in Section VII.

Following the FL protocol, the adversary uploads its local model parameters to the server and downloads the aggregated results in each round. This way it behaves like a normal client that collaborates with other clients to train the classification model. To conduct the attack, the adversary runs the GAN model locally, and manipulates its local dataset using the generated samples. This 'infects' the model parameters that the adversary uploads to the parameter server and affect the aggregated model parameters based on an aggregation function. Subsequently, the other clients are 'infected' and their sub-models become 'too' good at distinguishing the target class, thus the parameters uploaded leak more information of the target class. The details of the attack procedure is introduced in the next section.

V. Proposed Privacy Attack

In this section, we introduce the workflow of the proposed class property inference attack.

A. Attack Target and Outline

Our class property inference attack is not targeted at reconstructing the actual rows in the real tabular data, e.g., the records of some specific customers. Instead the adversary aims to infer only the properties that characterise the target class. Let class a be the target class of our attack, then the *class property* refer to the distributions of the features in class a data, i.e., $\mathbb{P}(\mathbf{x} \mid \mathbf{y} = a)$. Particularly, we focus on evaluating the efficacy of our attack model to infer the distributions of sensitive columns in the targeted real data. In our work, the sensitive columns are selected based on the following criteria:

- The content of the column contains personal information that needs to be protected from public view. The typical examples include age, income and marital status information of customers.
- The properties of the column values, e.g., range and distribution, can be potentially exploited by scammers or competitors.
- The column should have a certain effect on the classification model's prediction. In other words, the selected column has a correlation with the prediction target. Otherwise in real FL scenarios, it can problematic to use irrelevant private features for model training.

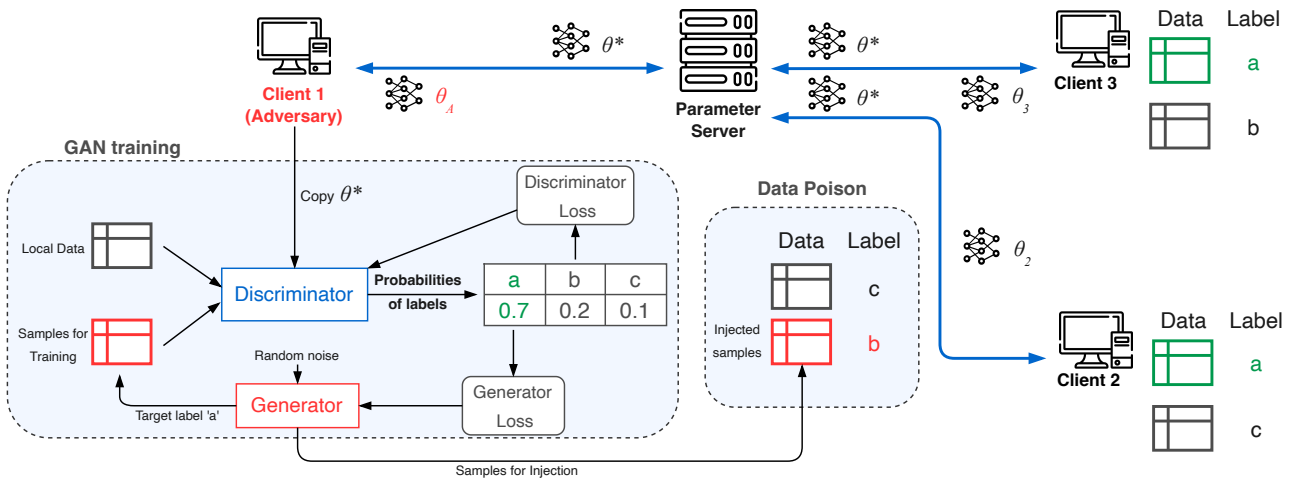


Fig. 3: GAN-based Attack on Federated Learning.

To conduct the attack, the adversary trains a GAN locally to generate synthetic samples of class a. Specifically, the network architecture of the generator $\mathcal{G}$ follows the tabular GAN structure proposed in [18]. A GAN discriminator $\mathcal{D}$ requires both the real and synthetic samples as input, as illustrated in Fig. 2. However, in our attack, the GAN runs locally on the adversary client and thus real samples of the class a are not available. The solution is to let the adversary employ the global model $f^*(\mathbf{x};\theta^*)$ as its GAN's discriminator. This is possible because the global model is an aggregation of the classification models trained on all clients, some of which use the class a data as input. The adversary exploits this attribute to learn the distribution of the target class data without directly accessing class a data.

Following the idea of image GAN-based attack in [20], we employ a data poison method that surreptitiously influences the FL process into leaking more information about the target class a. However, [20] changes the global model output dimension from M to $M+1$ (the additional one for judging fake/real data), given the fact that there are only M classes in the dataset. This is not realistic in our case because it is suspicious for the adversary to change the output dimension. In our approach, the adversary injects a number of synthetic samples into its local training dataset but changes the labels of those injected samples to class b. When the adversary trains the local classification model with the poisoned training dataset, the model sees a number of samples whose distribution is similar to class a but are actually labeled as b. Consequently, the FL system needs to work harder in order to distinguish these injected samples from the real class a data. The discriminator finally benefits from this impact as the global model becomes better at classifying class a.

B. Class Property Inference Attack

The procedure of our class property inference attack is depicted in Fig. 3. For simplicity, we consider the case in which three clients (client 1, 2, and 3) collaboratively train a classification model. Overall, there are three types of labels to be predicted in the training data, class a, b, and c. To better elaborate our idea, the example in Fig. 3 follows the label-skewed Non-IID data scenario discussed in Section IV-B. Specifically, each client is assumed to own the data of only two different classes, i.e., Client 1 has classes (b,c), Client 2 has classes (a,c), Client 3 has classes (a,b). As illustrated in the figure, Client 1 is assumed to be the adversary in the FL system, which aims to infer the properties of the class a data (highlighted in green). The steps of the attack are summarised as follows:

(i) The clients, including the adversary, establish a consensus on the architecture of the classification model. The parameter server computes the aggregation weights based on the statistic information collected from the clients. This initialisation process is explained in Section VI-B.

(ii) The FL process runs for a number of rounds, following the protocol introduced in Section IV-A. In each round the parameter server aggregates the parameters of models uploaded by all clients and distributes the global model to each of them.

(iii) Specifically, within the above step, the normal participants Client 2 and Client 3 follow the steps below:

1) The client downloads the global model parameters θ^* from the parameter server to update its local model.
2) The client trains the updated model for a few epochs using its local data, i.e., Client 2 with class (a,c) data, Client 3 with class (a,b) data.
3) The parameters of the trained local model θ_i is uploaded to the parameter server.

(iv) Meanwhile, the adversary uploads and downloads model parameters in the same way as normal clients, but with different training methods:

1) The adversary downloads the global model parameters θ^* to update its local model, and makes a copy of $f^*(\mathbf{x};\theta^*)$ to be the discriminator $\mathcal{D}$.

2) The generator $\mathcal{G}$ takes random noise as input, and generates samples to emulate class a data. Note that the output dimension of $\mathcal{G}$ is identical to the feature dimension of the training data, since $\mathcal{G}$ aims to generate one particular type of data.
3) The adversary trains $\mathcal{D}$ with both its local real data and the samples generated by $\mathcal{G}$. Here we need to train with the real data because the performance of $\mathcal{D}$ is unstable in the early stage of FL training. The output of $\mathcal{D}$ is a multinomial probability distribution of being classified into the three classes.
4) The discriminator loss and generator loss are computed and used to update $\mathcal{D}$ and $\mathcal{G}$ respectively.
5) The adversary generates samples from the $\mathcal{G}$ and assigns label b to these samples.
6) The real training data is mixed with the generated samples.
7) The adversary trains its local model on the poisoned dataset.
8) The parameters of the adversary's local model, θ_A is uploaded to the parameter server.

(v) The FL system finishes training when the global model $f^*(\mathbf{x}; \theta^*)$ converges and reaches a predefined accuracy on all clients.

C. Quantitative Analysis of Privacy Leakage

In our work, we evaluate the efficacy of the proposed attack via similarity analysis. The more similar the synthetic samples are to the targeted real data, the more serious the privacy leakage is, i.e., the more effective the attack model is.

In the attacks on image data, the similarity between generated images and the target ones is normally evaluated by people's subjective opinions. For instance, in order to quantify the efficacy of their attack on facial recognition models, Fredrikson et al. perform experiments using Amazon's Mechanical Turk to see if human can use their generated facial images to correctly pick the target person from a list [6]. The authors take the accuracy of human judgement as the evaluation metric of similarity. Such evaluation is not applicable for tabular data since the class of the generated records can not be simply judged by observation.

In this paper, two metrics are used to measure the similarity between synthetic tabular samples and the targeted real tabular data: the Jensen-Shannon Divergence (JSD) and the Wasserstein Distance (WD). Specifically, we use JSD to calculate the similarity distance between two discrete columns, and WD for the distance between two continuous columns. The JSD between two probability vectors p and q is defined mathematically as:

$$JSD(p,q) = \sqrt{\frac{KL(p||m) + KL(q||m)}{2}} \tag{3}$$

where m is the point-wise mean of p and q, and KL is the Kullback-Leibler divergence [33]. The JSD distance metric is symmetric and bounded between 0 and 1 which makes it easier to interpret the result. But one limitation of JSD is that it is impossible to calculate JSD distance if two distributions have no overlapping. In practice, the calculation demands the vectors p and q have the same length, which makes it not suitable for continuous columns.

The WD between two distributions u and v is defined as:

$$WD(u,v) = inf_{\pi \in \Gamma(u,v)} \int_{\mathbb{R}\times\mathbb{R}} |x-y| d\pi(x,y) \tag{4}$$

where $\Gamma(u,v)$ is the set of probability distributions on $\mathbb{R}\times\mathbb{R}$ whose marginals are u and v on the first and second factors, respectively. It can be interpreted as the minimum cost to transform one distribution into another where the cost is given by amount of distribution to shift times the distance it must be shifted. JSD and WD are also applied in the aggregation function of our FL framework introduced in Section VI-B.

We use $\mathcal{T}^\circ = \{\mathbf{x}^\circ, \mathbf{y}^\circ\}(y^\circ = a)$ to denote the targeted tabular data. The sensitive columns in $\mathcal{T}^\circ$ consist of n continuous and m discrete features, denoted by $\mathbf{x}_p^\circ = \{C_1^\circ, \ldots, C_n^\circ, D_1^\circ, \ldots, D_m^\circ\}$. Let $\mathcal{T}' = \{\mathbf{x}', \mathbf{y}'\}$ be the synthetic tabular samples generated by the attack model, and $\mathbf{x}'_p = \{C_1', \ldots, C_n', D_1', \ldots, D_m'\}$ be the sensitive columns, then the similarity between $\mathcal{T}^\circ$ and $\mathcal{T}'$ is quantitatively measured by $JSD(D_i^\circ, D_i')(i \in [1,m])$ and $WD(C_i^\circ, C_i')(i \in [1,n])$.

TABLE I: Dataset used in our experiments.

Dataset	**Bank Loan**	**Income Type**
Number of Records	5,000	10,000
Target column	Personal loan	Income type
Sensitive columns	Age, income, family members, mortgage, credit card usage	Gender, family members, income, marital status, number of children, age, education type
Accuracy, AUC on CL	0.9850, 0.9951	0.9840, 0.9780
Accuracy, AUC on FL	0.9770, 0.9849	0.9725, 0.9921

VI. Experimental Setup

In our experiments, we focus on the scenarios of digital finance, where the data processed are financial data and the sensitive columns are considered commercial confidentiality. We note that the associated code is available upon request.

A. Datasets

Bank Loan dataset. The dataset contains the records of 5,000 customers from Thera Bank[1]. The prediction target is a binary category that indicates whether the individual has applied for the personal loan. After removing the irrelevant features, the dataset has 11 features, as listed in Table I. In the five selected sensitive columns, four are continuous columns while *family members* is a discrete column.

Income Type dataset. The dataset is sampled from the Credit Card dataset[2] and contains the records of 10,000 credit card applicants. Instead of the previous binary prediction

[1]https://www.kaggle.com/datasets/itsmesunil/bank-loan-modelling

[2]https://www.kaggle.com/datasets/rikdifos/credit-card-approval-prediction

target, we choose the attribute *income type* as our target column, which has three different classes: *working*, *commercial associate*, and *state servant*. There are 15 features in the dataset, including seven sensitive columns, which consist of five discrete columns and two continuous columns (*age* and *income*), as depicted in Table I.

Bank Loan dataset is for binary classification scenario, in which the records are classified into one of two classes, normally positive or negative. The case of classifying records into one of three or more classes is called multi-class classification, which we study using the Income Type dataset. As a benchmark, we first train centralised machine learning models with the datasets. On each dataset, we use a fully connected deep neural network, known as multi-layer perceptron to predict the target labels. Each dataset is split into training set for model training and test set for evaluation. As listed in Table I, the centralised learning (CL) prediction accuracy reaches 0.9850 on the Bank Loan test set and 0.9840 on the Income Type test set.

In the FL experiments we split the dataset into $(K+1)$ subsets, where K is the number of clients in the FL system. Each client owns one of the subsets, and the remaining subset is used as the held-out test set. In the case of FL with Non-IID data, the prediction accuracy of the classification model is generally lower than in the centralised learning. We mitigate this deterioration in accuracy by using a similarity based aggregation algorithm in our FL system, which will be introduced in the next section. Benefiting from this similarity based aggregation algorithm, the FL prediction accuracy with Non-IID data remains above 0.97.

B. Federated Learning Framework for Tabular Data

In previous FL work such as [8], [21], the authors compute the aggregation weights based only on the size of local data, which is not comprehensive, especially when it comes to non-IID data. [34] proposes a way to calculate aggregation weights based on class distribution, but it only works for single label data. Since tabular data contains multiple columns, each column can have different distribution and data type (e.g., discrete and continuous). Therefore, the previous weighting algorithm cannot be directly applied. [12] designs a mechanism to calculate aggregation weights in FL for tabular data based on (i) data similarity between local and global and (ii) size of data. When calculating data similarity, [12] evaluates the distance between local and global distribution column by column.

Discrete columns use the Jensen-Shannon Divergence (JSD) [35] to calculate the distance between local and global class distribution. For each discrete column j and client i, [12] computes the similarity distance JSD_{ij} between local and global class distribution according to Eq. (3). Concretely, class distribution is represented by a probability vector (i.e., p and q in Eq. (3)) based on image class frequency. Local and global vectors have the same length (i.e., the number of all classes in the group) and corresponding bit in all vectors should represent same class.

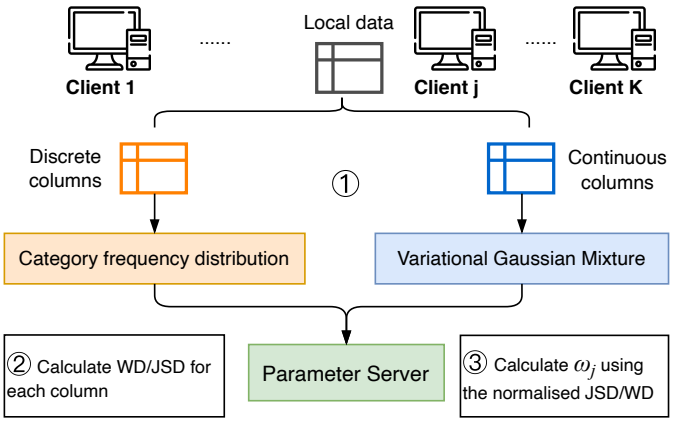


Fig. 4: The initialisation process before FL training.

Continuous columns use the Wasserstein Distance (WD) [36]. For client i, it first estimates a Variational Gaussian Mixture (VGM) for their continuous column j and sends the VGM_{ij} to server. Server samples the continuous column C_{ij} using VGM_{ij}, sampling size is the same as the local data size of client i. Server gathers all the samples: C_j = $\{C_{1j}$, C_{2j}, ... $C_{Kj}\}$ where K is the number of clients, and uses C_j as an approximation of global distribution of column j. Then the distance between local and global distribution – WD_{ij} is calculated by Eq. (4) between $\mathcal{T}_{ij}$ and $\mathcal{T}_j$ for each client i of continuous column j.

Once each client calculates the distances for all the columns, a normalization process is applied on these distances combined with size of local data to calculate the final aggregation weights in Eq. (1). The above initialisation process is summarised in Fig. 4. The implication of this aggregation algorithm is, the more similar the client's local data is to the overall dataset, the higher weight it gets in the model aggregation.

Our FL framework is implemented using the Pytorch RPC framework. This choice makes it easy to control the flow of the training steps from the server. Clients just need to join the group, then wait to be initialized and assigned work. To parallelize the training across all clients, RPC provides a function *rpc_async()* which allows the server to make nonblocking RPC calls to run functions at a client.

One drawback of current RPC framework from Pytorch v1.8.1 is that it does not support the transmission of tensors directly on GPU through RPC call. This means that each time when we collect or update the model weights we need to pay an extra time cost to detach the weights from GPU to CPU or reload the weights from CPU to GPU. In this work we ignore the communication cost and focus only on the privacy concerns.

C. Model Setup

For all the experiments we conduct preprocessing on the dataset to speed up the training process. The continuous columns are scaled into the $[-1, 1]$ range while the discrete

TABLE II: The Network Architectures in FL Experiments

Bank Loan dataset	Classifier/ Discriminator	$LL(20,32) \xrightarrow{ReLU} LL(32,64) \xrightarrow{ReLU}$ $LL(64,128) \xrightarrow{ReLU} LL(128,256) \xrightarrow{ReLU}$ $LL(256,128) \xrightarrow{ReLU} LL(128,64) \xrightarrow{ReLU}$ $LL(64,32) \xrightarrow{ReLU} LL(32,2) \rightarrow Sigmoid()$
	Generator	$LL(128,256) \rightarrow BN(256) \xrightarrow{ReLU}$ $LL(384,256) \rightarrow BN(256) \xrightarrow{ReLU}$ $LL(640,20) \rightarrow Tanh()$
Income Type dataset	Classifier/ Discriminator	$LL(62,64) \xrightarrow{ReLU} LL(64,128) \xrightarrow{ReLU}$ $LL(128,256) \xrightarrow{ReLU} LL(256,128) \xrightarrow{ReLU}$ $LL(128,64) \xrightarrow{ReLU} LL(64,3)$
	Generator	$LL(128,256) \rightarrow BN(256) \xrightarrow{ReLU}$ $LL(384,256) \rightarrow BN(256) \xrightarrow{ReLU}$ $LL(640,62) \rightarrow Tanh()$

columns are one-hot encoded. As introduced in Section V, the FL system with an adversary involves three neural network models: (i) the classification model (classifier), denoted by f^*, (ii) the discriminator $\mathcal{D}$ which shares the same architecture with the classification model, (iii) the generator $\mathcal{G}$. The network architectures for the Bank Loan and Income Type datasets are depicted in Table II. *LL* represents the Linear Layer and *ReLU*, arrows denote the links between the layers, *Sigmoid* and *Tanh* are the activation functions used. Batch normalisation, denoted by *BN*, is adopted at the intermediate layers of the generator. The architecture of $\mathcal{G}$ refers to the work in [18].

In the FL training process, we adopt Adam optimizer in f^* with the learning rate of 0.0006 for the Bank Loan dataset, and 0.001 for the Income Type dataset. In the experiments where the adversary is enabled, we use the Adam optimizer in $\mathcal{D}$ with the learning rate of 0.0002 and the weight decay of 1e-6. SGD optimizer is used in the $\mathcal{G}$ and the learning rate is set to 0.0002 with the momentum of 0.9. We arrived at these values based on our experience running the experiments with the two tabular datasets. In both normal and attack experiments, each client trains f^* for 10 epochs before uploading the model parameters to the parameter server. We finish the FL training and save the models for evaluation when the performance of f^* stops improving on each client.

In the experiment with an adversary client, we do the normal FL training for the first few rounds and run the attack model when the accuracy of f^* reaches a specified threshold (e.g. 0.85) on the adversary's local data. This makes the training of the GANs more efficient as $\mathcal{D}$, whose parameters are copied from f^*, starts from a considerable accuracy. This schema is reasonable in realistic as the adversary stays inactive until it observes that f^* reaches a functional level on its local data. This threshold also works for the data poison process, which means the adversary will not inject new samples to its local dataset until f^* achieves a certain accuracy to classify the poisoned dataset.

VII. Experimental Results

In this section we evaluate the efficacy of the proposed attack by comparing the synthetic samples with the targeted real data. We design use cases for both binary and multi-class classification scenarios. We run two clients in the experiments with the Bank Loan dataset, and three clients with the Income Type dataset. One of the clients is selected to play the role of adversary, and we report on the synthetic samples created in each round. All experiments are performed on a workstation running Ubuntu 20.04 LTS equipped with a 3.9 GHz CPU Intel Xeon W-2245, 16 cores, 128GB RAM and an Nvidia Quadro RTX6000 GPU card. In our experiments, each client represents a finance company that joins the FL system. The computing and communication resources are assumed to be sufficient and stable since the hardware devices and network are supposed to be deployed at enterprise level. Therefore, we do not consider the case with heterogeneous clients, which is commonly considered in FL system based on smartphone devices [37].

A. Class Property Inference

Binary classification. We use the Bank Loan dataset to study a binary classification model f_b^*. The prediction target *personal loan* has two categories: *negative* indicates the customer has never accepted a personal loan offer from the bank, while *positive* indicate the customer has previously taken such a loan. We select the *positive* class as the target class of our inference attack. In our experiments, the adversary owns 50% of the overall dataset, the other client holds 40%, and the remaining 10% is used as the test set for evaluating f_b^*.

To demonstrate privacy leakage most effectively, the adversary is designed to own only a small number of *positive* records, i.e., 1% of its local data. Hence, the adversary has relatively little knowledge about the properties of the *positive* class due to the insufficient sample size. By contrast, the other client owns most of the *positive* records in the overall dataset.

Fig. 5 illustrates the inference results of the positive customers' *income* distribution during the FL training process. In the figures, the green (right-most) curve represents the *income* distribution of all *positive* customers in the overall dataset. The curve filled with blue (shaded) area is the *income* distribution of the synthetic samples generated by the adversary's GAN in different FL training rounds. For comparison, we also plot the positive customers' income distribution that the adversary observes from its local data, represented by the red (left-most) curve. The figure displays the results for increasing number of training rounds, from 200 to 1000. The FL training finishes at round 1000 when the classification model f_b^* reaches convergence.

An intuitive, indicative way to assess the efficacy of our attack is to visually check how well the distribution of synthetic samples (the curve filled with blue area) fits the distribution of all target data (the green curve). (Note, a formal assessment using distance measures is carried out in the next section.) We observe that the synthetic samples are not yet able to emulate the actual *income* distribution of *positive* customers at earlier

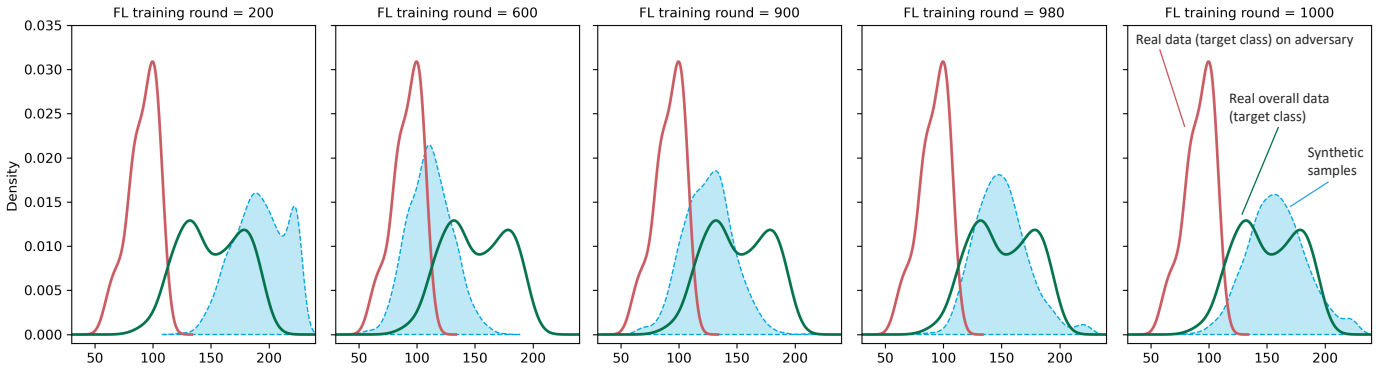


Fig. 5: The distribution of positive customers' income, for increasing training rounds, using the Bank Loan dataset. The red (left-most) curve is for the adversary's local data; The green (right-most) curve is for the overall dataset; The curve filled with blue (shaded) area is from the synthetic samples generated by the adversary's GAN. For increasing FL training rounds, the generated distribution increasingly matches the overall data.

rounds (round 200 to 600). As the FL training progresses, the distribution of synthetic samples converges and stabilises within the same range of the actual distribution (round 900 to 1000). This is because the classification model f_b^* becomes better at distinguishing *positive* from *negative* records, and consequently the GAN benefits from the discriminator $\mathcal{D}$, whose parameters are copied from f_b^*.

The results of the class property inference attack illustrates how private information is leaked. The adversary constructs a distribution that reasonably closely matches the distribution the real data. More precisely, where the adversary initially would conclude from its own data that *positive* customers have an income level between about 50 and 150 (the red curve in Fig. 5), by using the proposed attack, the adversary manages to capture the information that it is not supposed to know: *positive* customers actually have a income level approximately between 100 and 200 (the blue shaded area). Such privacy violation defeats the reliability of distributed privacy-preserving learning.

Multi-class Classification. For FL for multi-class classification we us the Income Type dataset. The overall dataset consists of three classes: *Working* (52.4%), *Commercial associate* (33.6%) and *State servant* (14%). The number of participants (clients in FL context) is three, and each client owns just two classes of data. Specifically, the adversary has data about classes (*Working*, *Commercial associate*), while the other two hold (*Working*, *State servant*) and (*Commercial associate*, *State servant*) respectively. The target class of our attack is the properties of the *State servant* data, e.g., marital status, secondary education, etc.

Fig. 6 illustrates the inferred distribution of *State servants'* marital status over FL training rounds. Marital status is a discrete column including five categories. In early rounds, the GAN is still generating relatively arbitrary outputs, as illustrated by the red bars in round 200 and 240 (the red bar is the right of the two bars for each value on the horizontal axis). After 500 rounds of FL training, the attack model captures the private information that most *State servants* are married. Finally, the inferred distribution stabilises and approaches the actual distribution, as observed from the figures in round 710 and 720. In addition to the example depicted in Fig. 6, the adversary can infer other private properties from the synthetic samples. For instance, in the synthetic samples, 64% of the *State servants* have secondary education and 34% have higher education. This observation is close to the characteristic in the real overall dataset, where the proportions are 58% and 39% respectively.

B. Similarity Analysis

As introduced in Section V-C, we use the distance measures JSD and WD to quantify the similarity between synthetic tabular samples $\mathcal{T}'$ and the targeted real tabular data $\mathcal{T}^\circ$. We calculate JSD between $\mathcal{T}'$ and $\mathcal{T}^\circ$'s discrete columns, and WD between continuous columns. A smaller value of JSD/WD indicates that the values in the two columns are more closely distributed (specifically, JSD/WD for two identical columns equals to 0).

Fig. 7 depicts the WD of *income* between the $\mathcal{T}^\circ$ and $\mathcal{T}'$, for increasing number of FL training rounds, for the binary classification problem (the Bank Loan dataset). The WD distance metric version of the results visualized in Fig. 5 for the *income* class property, is given in Fig. 7, the curve 'WD with data poison'. The WD/JSD counterpart of the results for multi-class classification after 720 rounds, visualized in the right-most chart of Fig. 6, is in Fig. 9 (the left most column for each of the properties given on the horizontal axis).

Data Poisoning. For the setting of data poisoning, it is recommended that the amounts of poisoning data should not exceed 5% of the adversary's local training data size. Otherwise it is difficult for the FL global model to converge. Through our experimental results, it turns out that the use of data poisoning (see Section V-A) is important for better convergence of the distance metrics. To show this, we display in Fig. 7 results with data poisoning enabled and with data poisoning disabled, respectively, while all other settings remain the same. Fig. 7 indicates that without data poisoning (the

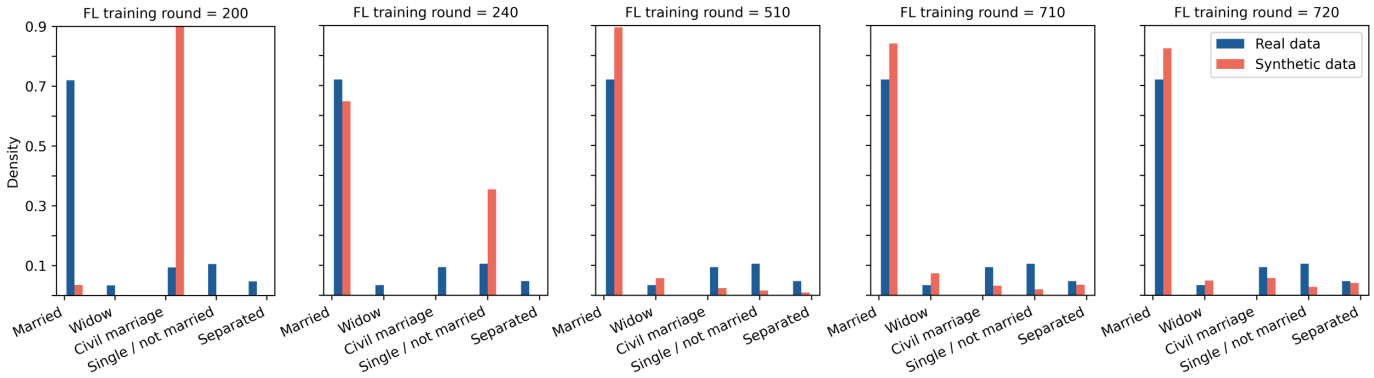


Fig. 6: The distribution of State servants' marital status, for increasing training rounds, using the Income Type dataset. Each bar denotes the density/proportion of records with the specific marital status. For increasing training rounds, the generated distribution increasingly matches the overall data.

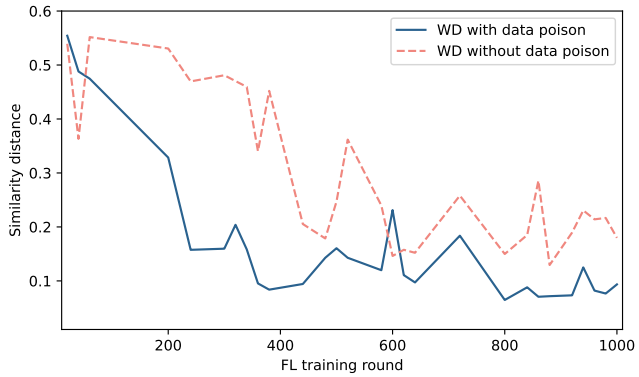


Fig. 7: The WD similarity distance between synthetic samples and the targeted real data, for increasing FL training rounds, using the Bank Loan dataset. Results are shown with and without data poison, indicating that using data poison achieves better convergence.

dashed curve), WD converges less than with data poisoning enabled. It is observed that the WD fluctuates before the FL system finishes training. This is due to the fact that every time the adversary copies the parameters from the global model f^*, it builds a new $\mathcal{D}$ which requires several rounds of training before the GAN is stable.

C. Comparison with Tabular GANs for Synthetic Data

To the best of our knowledge, this paper is the first to use tabular GANs for privacy attacks in FL, and therefore there is no work we can directly compare our approach to. However, a comparison is possible with GANs used for the generation of synthetic tabular data, in particular CTGAN [18], CTAB-GAN [19] and CTAB-GAN+ [31].

We train the advanced tabular GANs until convergence (300 epochs in our experiments) and calculate the JSDs and WDs between $\mathcal{T}^\circ$ and the synthetic samples generated by these advanced tabular GANs. Note that these advanced tabular GANs are trained in a single process independent of the FL system, and they all require the targeted real dataset $\mathcal{T}^\circ$ as input. The synthetic samples of our approach are generated by the adversary's GAN in the last round of FL training, round 1000 for the Bank Loan dataset (last figure in Fig. 5) and round 720 for the Income Type dataset (last figure in Fig. 6).

The similarity results for the targeted class in Bank Loan dataset are depicted in Fig. 8. Note that in Fig. 5, we only showed the results for *income*, here we present the results for other class properties as well. One would have expected that the synthetic samples in our approach have a larger similarity distance than the advanced tabular GANs' synthetic samples, since our GAN is not trained with the targeted real data. However, we can observe that the WD of our approach even outperforms CTGAN and CTAB-GAN in terms of the *age* and *income* class properties. We speculate that the reason for this is that these two columns have significant impacts on the FL classification model and consequently it becomes easier for our GAN to capture the distributions. By contrast, the *mortgage* and *family members* distributions show a larger WD/JSD between the adversary's constructed data and the targeted real data. This is because the distributions are difficult to emulate (*mortgage* is a long tail distribution and most values are 0) and the features contribute less to the global classification model.

The similarity results obtained for the multi-class classification experiments are given in Fig. 9. With respect to the class properties, only *income* and *age* are continuous values and the rest are discrete. One can observe that the similarity of our synthetic samples outperforms the advanced tabular GANs in terms of the *number of children*, *income* and *education type* features. In the targeted real dataset, the gender ratio of the *State servant* class is approximately 0.3 (male/female). Our approach fails to infer this property and generates samples with the gender ratio of 0.84. The JSD/WDs of other sensitive columns in our synthetic data are comparable with the results obtained from the advanced tabular GANs.

D. Discussion

Our class property inference attack aims at inferring the macro-level tabular data property of the target class. While

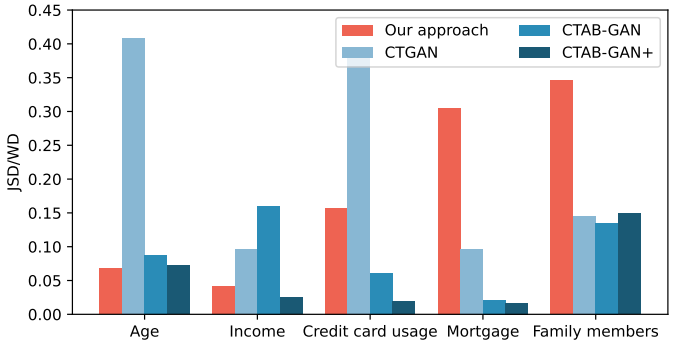


Fig. 8: The similarity distance between the targeted real data (*positive* class of Bank Loan dataset) and synthetic samples.

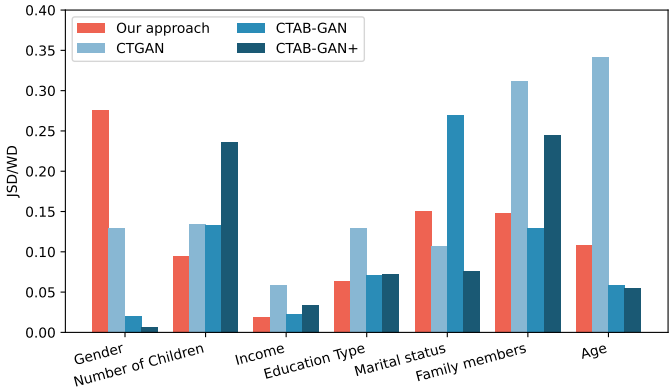


Fig. 9: The similarity distance between the targeted real data (*State servant* class of Income Type dataset) and synthetic samples.

such information leakage does not reveal the actual private data of any individual, it can be detrimental to the FL clients, e.g., financial companies in a real-world setting. Moreover, it is difficult to detect our attack from either the client or parameter server perspective. Unlike attack models that rely on white-box access to the global model [9], [22], [27], our adversary only requires black-box access and does not modify the global model directly. Thus our attack's impact on the final trained global model is negligible: throughout our experiments, the accuracy of the global model is above 0.97.

One limitation of our attack is that the inference result depends on the correlations between the features and the target class. Therefore the distributions of 'weak' features are not likely to be reconstructed by the generator. In practice, the FL clients are suggested to include as many 'weak' features as they can to mitigate such attack.

Existing record-level defenses such as Differential Privacy are proven to be less effective on property inference attacks [20], [38]. For other counter measures, we provide suggestions from a software engineering perspective. FL is an emerging technology and its Software Development Life Cycle is still being explored. Our work encourages software engineers to advance this life cycle and improve FL's security. In the requirement analysis phase, engineers will pay more attention to the clients with highly imbalanced training data since these clients have the motivation to conduct such attack. Counter measures at the architecture design level may mitigate the impact of our attack. For instance, if the clients' training data can be kept unchangeable throughout the FL process, then data poison is prevented. Our work intends to point software and system engineers to this potential threat and to inspire research in effective detection methods.

VIII. Conclusion

In this paper, we propose, implement and evaluate a GAN-based privacy attack against Federated Learning that processes tabular data. The attack enables a malicious client to infer properties that characterise a specific class, without actually having access to the data. The malicious client runs a tabular GAN locally, exchanges model parameters per the usual FL protocol, and utilises the global model as its discriminator. Synthetic samples generated by our GAN reveal potentially sensitive properties of the target class. We use similarity metrics to evaluate the seriousness of this privacy risk in our experiment. The results show that our GAN-based attack manages to infer the distributions of continuous and discrete properties exhibited by the target class data with increasing accuracy for more rounds of model updates. Interestingly, our approach to generate synthetic samples for a privacy attack at times outperforms state-of-the-art GAN-based synthetic data generators, which are trained with the actual targeted data. This is especially the case for data properties that heavily influence the classification outcome and it will be worth investigating our approach for the generation of synthetic data. We finally note that our attack is difficult to detect since the adversary behaves like a normal client. In future work, we aim to investigate counter measurements against this attack, including client-level differential privacy.

IX. Acknowledgments

This work is supported by the UK Engineering and Physical Sciences Research Council for the projects titled "Fintrust: Trust Engineering for the Financial Industry" (EP/R033595/1).

References

[1] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[2] Y. Wu, S. Cai, X. Xiao, G. Chen, and B. C. Ooi, "Privacy preserving vertical federated learning for tree-based models," in *The 46th International Conference on Very Large Data Bases (VLDB)*, 2020, pp. 2090–2103.

[3] Y. Cheng, Y. Liu, T. Chen, and Q. Yang, "Federated learning for privacy-preserving ai," *Communications of the ACM*, vol. 63, no. 12, pp. 33–36, 2020.

[4] Y. Liu, A. Huang, Y. Luo, H. Huang, Y. Liu, Y. Chen, L. Feng, T. Chen, H. Yu, and Q. Yang, "Fedvision: An online visual object detection platform powered by federated learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 08, 2020, pp. 13 172–13 179.

[5] M. Alazab, S. P. RM, M. Parimala, P. Reddy, T. R. Gadekallu, and Q.-V. Pham, "Federated learning for cybersecurity: concepts, challenges and future directions," *IEEE Transactions on Industrial Informatics*, 2021.

[6] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1322–1333.

[7] N. Agarwal, A. T. Suresh, F. X. X. Yu, S. Kumar, and B. McMahan, "cpsgd: Communication-efficient and differentially-private distributed sgd," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[8] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 691–706.

[9] 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 *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 2512–2520.

[10] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[11] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[12] Z. Zhao, R. Birke, A. Kunar, and L. Y. Chen, "Fed-tgan: Federated learning framework for synthesizing tabular data," *arXiv preprint arXiv:2108.07927*, 2021.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[14] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[15] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[16] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *Advances in neural information processing systems*, vol. 29, 2016.

[17] N. Park, M. Mohammadi, K. Gorde, S. Jajodia, H. Park, and Y. Kim, "Data synthesis based on generative adversarial networks," in *The 44th International Conference on Very Large Data Bases (VLDB)*, 2018.

[18] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional gan," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[19] Z. Zhao, A. Kunar, R. Birke, and L. Y. Chen, "Ctab-gan: Effective table data synthesizing," in *Asian Conference on Machine Learning*. PMLR, 2021, pp. 97–112.

[20] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 603–618.

[21] H. Zhu, J. Xu, S. Liu, and Y. Jin, "Federated learning on non-iid data: A survey," *Neurocomputing*, vol. 465, pp. 371–390, 2021.

[22] X. Luo, Y. Wu, X. Xiao, and B. C. Ooi, "Feature inference attack on model predictions in vertical federated learning," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 181–192.

[23] L. Lyu, H. Yu, and Q. Yang, "Threats to federated learning: A survey," *arXiv preprint arXiv:2003.02133*, 2020.

[24] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 3–18.

[25] Y. Long, V. Bindschaedler, L. Wang, D. Bu, X. Wang, H. Tang, C. A. Gunter, and K. Chen, "Understanding membership inferences on well-generalized learning models," *arXiv preprint arXiv:1802.04889*, 2018.

[26] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *2019 IEEE symposium on security and privacy (SP)*. IEEE, 2019, pp. 739–753.

[27] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An {End-to-End} case study of personalized warfarin dosing," in *23rd USENIX Security Symposium (USENIX Security 14)*, 2014, pp. 17–32.

[28] S. Kazeminia, C. Baur, A. Kuijper, B. van Ginneken, N. Navab, S. Albarqouni, and A. Mukhopadhyay, "Gans for medical image analysis," *Artificial Intelligence in Medicine*, vol. 109, p. 101938, 2020.

[29] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[30] E. Choi, S. Biswal, B. Malin, J. Duke, W. F. Stewart, and J. Sun, "Generating multi-label discrete patient records using generative adversarial networks," in *Machine learning for healthcare conference*. PMLR, 2017, pp. 286–305.

[31] Z. Zhao, A. Kunar, R. Birke, and L. Y. Chen, "Ctab-gan+: Enhancing tabular data synthesis," *arXiv preprint arXiv:2204.00401*, 2022.

[32] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[33] J. M. Joyce, *Kullback-Leibler Divergence*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 720–722. [Online]. Available: https://doi.org/10.1007/978-3-642-04898-2_327

[34] R. Guerraoui, A. Guirguis, A.-M. Kermarrec, and E. L. Merrer, "Fegan: Scaling distributed gans," in *Proceedings of the 21st International Middleware Conference*, ser. Middleware '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 193–206. [Online]. Available: https://doi.org/10.1145/3423211.3425688

[35] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 145–151, 1991.

[36] A. Ramdas, N. G. Trillos, and M. Cuturi, "On wasserstein two-sample testing and related families of nonparametric tests," *Entropy*, vol. 19, no. 2, 2017. [Online]. Available: https://www.mdpi.com/1099-4300/19/2/47

[37] C. Yang, Q. Wang, M. Xu, Z. Chen, K. Bian, Y. Liu, and X. Liu, "Characterizing impacts of heterogeneity in federated learning upon large-scale smartphone data," in *Proceedings of the Web Conference 2021*, 2021, pp. 935–946.

[38] M. Naseri, J. Hayes, and E. De Cristofaro, "Local and central differential privacy for robustness and privacy in federated learning," in *29th Network and Distributed System Security Symposium (NDSS 2022)*, 2022.