Xue Jiang*, Xuebing Zhou, and Jens Grossklags

Comprehensive Analysis of Privacy Leakage in Vertical Federated Learning During Prediction

Abstract: Vertical federated learning (VFL), a variant of federated learning, has recently attracted increasing attention. An *active party* having the true labels jointly trains a model with other parties (referred to as *passive parties*) in order to use more features to achieve higher model accuracy. During the prediction phase, all the parties collaboratively compute the predicted confidence scores of each target record and the results will be finally returned to the active party. However, a recent study by Luo *et al.* [28] pointed out that the active party can use these confidence scores to reconstruct passive-party features and cause severe privacy leakage.

In this paper, we conduct a comprehensive analysis of privacy leakage in VFL frameworks during the prediction phase. Our study improves on previous work [28] regarding two aspects. We first design a general gradient-based reconstruction attack framework that can be flexibly applied to simple logistic regression models as well as multi-layer neural networks. Moreover, besides performing the attack under the white-box setting, we give the first attempt to conduct the attack under the black-box setting. Extensive experiments on a number of real-world datasets show that our proposed attack is effective under different settings and can achieve at best twice or thrice of a reduction of attack error compared to previous work [28]. We further analyze a list of potential mitigation approaches and compare their privacy-utility performances. Experimental results demonstrate that privacy leakage from the confidence scores is a *substantial* privacy risk in VFL frameworks during the prediction phase, which cannot be simply solved by crypto-based confidentiality approaches. On the other hand, processing the confidence scores with information compression and randomization approaches can provide strengthened privacy protection.

Keywords: vertical federated learning, privacy attacks

DOI 10.2478/popets-2022-0045

Received 2021-08-31; revised 2021-12-15; accepted 2021-12-16.

1 Introduction

Building artificial intelligence (AI) applications often requires a large amount of user data to train high-accuracy machine learning (ML) models. Traditional centralized learning mechanisms directly gather all relevant data from local sites for model training. This not only leads to a huge storage complexity and computational cost, but more importantly, it also suffers from serious privacy issues. To mitigate these disadvantages, federated learning (FL) [29] was proposed, where the models are jointly trained by multiple local parties coordinated by a central server. The framework achieves improvements in both computational efficiency and privacy protection, and it has been increasingly used in real-life applications, e.g., mobile keyboard prediction [16, 35], healthcare [24, 27], purchase recommendation [44, 49], and distributed synthetic data generation [3, 22, 47] systems.

Recently, vertical federated learning (VFL) [17, 38, 40, 50], a variant of FL, has gained increasing attention. In comparison to the generic FL framework, where each local party holds a different set of samples with the same feature space, the local parties in VFL systems hold different features of the same set of samples. There are usually two kinds of participants in a VFL system: the *active party* who has the real labels of a set of samples and the *passive parties* who provide additional features of the same set of samples. Usually, the active party initiates the training task and invites the passive parties to jointly train an ML model. The increase in input features helps improve the model's prediction accuracy. For instance, in a smart finance scenario, a bank wants to train a model to produce customers' credit scores and predict default risk. However, it has only a few features describing the customers' transaction history, which may not be sufficient for training a satisfactory model. Therefore, the bank wants to collaborate with an e-commerce company and use its large number of profile features such as age, gender, and purchase his-

***Corresponding Author: Xue Jiang:** Technical University of Munich; Huawei Technologies Düsseldorf GmbH, E-mail: xue.jiang@tum.de

Xuebing Zhou: Huawei Technologies Düsseldorf GmbH, E-mail: Xuebing.Zhou@huawei.com

Jens Grossklags: Technical University of Munich, E-mail: jens.grossklags@tum.de

tory to improve the model's prediction accuracy. Since both institutes cannot directly exchange real data due to privacy issues, the training task can be performed under a VFL framework. In addition to the data holders, existing VFL frameworks [12, 17, 48] usually also involve a trusted coordinator, who is responsible for coordinating the training and prediction process. During the training process, both parties train part of the model on the local side based on their data and send the intermediate results to the coordinator. Then, the coordinator aggregates the information and sends the gradients back to both parties to update their local models. Additionally, crypto-based technologies such as secure multi-party computation (SMC) [53] and homomorphic encryption (HE) [9], are applied in VFL frameworks for privacy protection. During the prediction process, each party computes the intermediate results using the trained model and the features of the new record. Then, the coordinator aggregates the results, computes the predictions (referred to as *confidence scores*), and returns the predictions to the active party.

Although the involvement of the trusted coordinator prevents potential privacy leakage from the intermediate results, a recent work by Luo *et al.* [28] shows that the active party can still use the confidence scores to reconstruct passive-party features during the prediction. The authors analyzed privacy leakage under a two-party VFL setting and respectively proposed an equation solving attack (ESA) and a generative regression network (GRN) attack on simple logistic regression (LR) models and complex multi-layer neural networks (NNs). Nevertheless, the work has the following limitations. First, although the proposed GRN attack can be applied to complex NNs, it usually requires the collection of numerous predictions to train the attack model, which may not always be feasible in practice. Second, the study only investigated privacy leakage under the white-box setting, where the trained passive-party model is revealed to the active party. However, in real-life scenarios, the passive party may not share the trained model due to privacy and intellectual property reasons. Thus, the proposed attacks in [28] would no longer be applicable.

In this paper, we extend the study of [28] with a more comprehensive analysis of the privacy leakage during the prediction phase under the two-party VFL setting. We address the limitations in [28] as follows. First, we design a generic gradient-based inversion attack (GIA) that can be applied to both simple and complex models. The algorithm can be applied flexibly and independently to any individual prediction. Second, we further conduct the attack under the black-box setting, where the active party is unaware of the passive-party model parameters. In this setting, we assume the active party has access to a small set of the passive-party data (referred to as *auxiliary data*). For instance, the active party may collude with a few data owners and gain complete knowledge about their data. Subsequently, the attacker can use these auxiliary data to build a shadow model that mimics the performance of the passive-party model and then perform the attack. In our experiments, we show that for simple models such as LR, the attacker only needs less than ten data records to achieve satisfactory attack performance. For multi-layer NNs, it is also possible to have similar performance with less than 25 data records. The results demonstrate the feasibility of our black-box attack in real-life scenarios. Our contributions can be summarized as follows:

- We propose a generic framework to perform reconstruction attacks on VFL during the prediction phase. The framework can be flexibly applied to different ML models, such as LR models and multi-layer NNs. The attack error of our proposed method can be reduced by twice or three times compared to the most relevant prior work [28].
- Besides performing the reconstruction attack under the white-box setting, we further give the first attempt to conduct the attack under the black-box setting. Experimental results demonstrate that with prior knowledge of a small set of auxiliary data, the attacker can still perform the reconstruction attack without the knowledge of the target model's real weights, or even the model structure.
- We conduct comprehensive experiments on real-world datasets to evaluate the attack performance under different settings. Experimental results show that privacy leakage during the prediction phase is a *substantial* privacy risk in VFL frameworks, which cannot be simply solved by crypto-based approaches. We further analyze possible mitigation approaches and show that it is necessary to apply additional information compression and randomization approaches during training or to the confidence scores to strengthen the privacy protection in VFL.

2 Related Work

2.1 Vertical Federated Learning

In comparison to horizontal federated learning (HFL), where data are horizontally partitioned and held by mul-

tiple local clients, VFL focuses on scenarios where the local clients hold different features of the same set of users. The goal of VFL is to jointly build AI models using the features from all parties. Therefore, simple model-averaging strategies are no longer applicable in VFL settings.

In recent decades, studies for vertically partitioned data have been widely conducted for various data mining and ML applications. For instance, Vaidya *et al.* proposed a series of privacy-preserving protocols for vertically-partitioned data covering association rule mining [38], k-means clustering [39], Bayes classifier [40], decision trees [41], and support vector machines [54] using SMC [53]. Hardy *et al.* [17] proposed a VFL framework combined with HE [9] for training LR models [9]. Yang [48] further applied the quasi-Newton method in the VFL framework to reduce the communication rounds. In addition to LR models, other works [7, 25, 46] also proposed VFL frameworks for tree-based models. In comparison to the above VFL frameworks, which are limited to two parties, Feng *et al.* [12] proposed a multi-participant multi-class VFL (MMVFL) framework that enables label sharing of its owner with other VFL participants in a privacy-preserving manner. Moreover, Yang *et al.* [51] proposed to remove the coordinator to reduce the complexity of the system. On the other hand, Hu *et al.* [19] and Chen *et al.* [6] proposed asynchronous VFL frameworks, where the models are updated by each party in an asynchronous manner and do not require feature sharing between parties. Furthermore, prior VFL frameworks primarily use crypto-based technologies such as HE and SMC to ensure secure and private learning. Recent works proposed incorporating differential privacy (DP) into the training process to provide strict privacy guarantees for local data [6, 42].

2.2 Privacy Risks in Federated Learning

Although FL achieves significant privacy benefits compared to centralized ML, a number of prior works have shown that FL still suffers from privacy attacks such as *membership inference attacks* and *reconstruction attacks*. Membership inference attacks determine whether a sample is included in the training dataset. Prior works [32, 37] conducted comprehensive analyses of membership inference attacks against FL under both white-box and black-box settings. As an extension to inferring membership, Melis *et al.* [30] further investigated a *property inference attack*, where the attackers can identify when a property appears in the data during training. On the other hand, reconstruction attacks aim to recover specific training data. Phong *et al.* [34] firstly showed that the input data could be mathematically derived from the gradients of first-layer weights and bias in fully-connected models. Hitaj *et al.* [18] and Wang *et al.* [43] utilized generative adversarial networks (GANs) to reconstruct representatives of local training data. Recently, Zhu *et al.* [56] and other follow-up studies [15, 55] proposed gradient-based attacks that can recover pixel-wise accurate original images and token-wise matching original texts.

The abovementioned attacks only focus on the HFL frameworks. Recently, Weng *et al.* [45] and Luo *et al.* [28] conducted privacy attacks against VFL. Weng *et al.* [45] analyzed privacy leakage during the training phase, whereas Luo *et al.* [28] focused on the privacy risks during the prediction phase. In this paper, we conduct a comprehensive analysis of the privacy leakage of VFL during prediction. We address the limitations of [28] regarding complex models and further extend the attack with a black-box setting.

3 Background

3.1 Machine Learning

An ML model $\mathcal{F}_{\boldsymbol{\phi}} : \mathcal{X} \mapsto \mathcal{Y}$ is a function $\mathcal{F}$ with a set of parameters $\boldsymbol{\phi}$ that maps data samples from the input (or feature) space $\mathcal{X}$ to the output space $\mathcal{Y}$. In this paper, we mainly focus on models for supervised classification tasks, including LR models and NNs.

3.1.1 Logistic Regression

As one of the most commonly used linear classifiers, the $\mathcal{F}_{\boldsymbol{\phi}}(\boldsymbol{x})$ of LR models can be represented as the inner product of input $\boldsymbol{x}$ and model parameters $\boldsymbol{\phi}$ followed by a nonlinear activation function. The confidence score $\boldsymbol{c}$ is calculated as follows:

$$\boldsymbol{c} = \mathcal{F}_{\boldsymbol{\phi}}(\boldsymbol{x}) = \xi(f(\boldsymbol{x}, \boldsymbol{\phi})) = \xi(\boldsymbol{\phi} \cdot \boldsymbol{x}), \tag{1}$$

where $f(\boldsymbol{x}, \boldsymbol{\phi}) = \boldsymbol{\phi} \cdot \boldsymbol{x}$ is the computation of the inner product and ξ is the output activation. Let $\boldsymbol{z} = \boldsymbol{\phi} \cdot \boldsymbol{x}$ be the intermediate result. For binary LR, $\boldsymbol{\phi}$ is a vector and $\boldsymbol{z}$ is a scalar. The activation function ξ is the *sigmoid* function defined as

$$\textit{sigmoid}: \quad \boldsymbol{c} = \xi(\boldsymbol{z}) = \frac{1}{1 + e^{-\boldsymbol{z}}}. \tag{2}$$

The confidence score $\boldsymbol{c}$ is a scalar between 0 and 1. For multi-class LR (also referred to as *softmax regression*), the model parameter matrix $\boldsymbol{\phi}$ can be vectorized as $\boldsymbol{\phi} = [\boldsymbol{\phi}_1, \cdots, \boldsymbol{\phi}_k]$, where k is the number of classes and $\boldsymbol{\phi}_m$ is the parameter vector of the m^{th} class. Thus, the intermediate result $\boldsymbol{z} = [z_1, \cdots, z_k]$ is a k-dimensional vector, where $z_m = \boldsymbol{\phi}_m \cdot \boldsymbol{x}$. Finally, the *softmax* function is applied to normalize $\boldsymbol{z}$ and the output $\boldsymbol{c} = [c_1, \cdots, c_k]$ is also a k-dimensional vector where each entry c_m is calculated as

$$softmax: \quad c_m = \xi(z_m) = \frac{e^{z_m}}{\sum_{j=1}^{k} e^{z_j}}. \tag{3}$$

3.1.2 Neural Networks

As an extension of LR, NNs have been widely used in ML and deep learning due to their capability to capture hidden features and patterns of training data. The NN model can also be generalized as $\mathcal{F}_{\boldsymbol{\phi}}(\boldsymbol{x}) = \xi(f(\boldsymbol{x}, \boldsymbol{\phi}))$, where $f(\boldsymbol{x}, \boldsymbol{\phi})$ represents computations of multiple interconnected layers. The computation on each hidden layer is similar to Equation (1). Besides the *sigmoid* function, commonly used hidden-layer functions also include *relu* and *tanh*:

$$relu: \quad \xi(\boldsymbol{z}) = \max\{0, \boldsymbol{z}\}, \tag{4}$$

$$tanh: \quad \xi(\boldsymbol{z}) = \frac{e^{\boldsymbol{z}} - e^{-\boldsymbol{z}}}{e^{\boldsymbol{z}} + e^{-\boldsymbol{z}}}. \tag{5}$$

The activation on the output layer is the same as in LR models, namely the *sigmoid* function for binary classifiers and the *softmax* function for multi-class classifiers.

3.1.3 Model Learning and Prediction

Let us assume we have a set of training samples $\{\boldsymbol{x}^1, \cdots, \boldsymbol{x}^n\}$, each with d features, and their labels $\{\boldsymbol{y}^1, \cdots, \boldsymbol{y}^n\}$ belonging to k classes. For each input $\boldsymbol{x}^i$, the confidence score $\boldsymbol{c}^i$ can be presented as $\boldsymbol{c}^i = \mathcal{F}_{\boldsymbol{\phi}}(\boldsymbol{x}^i)$. The goal of the training is to find a set of parameters $\boldsymbol{\phi}$ that achieves the minimum loss over the whole batch of inputs, which is given as

$$\boldsymbol{\phi} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\boldsymbol{y}^i, \boldsymbol{c}^i) = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\boldsymbol{y}^i, \mathcal{F}_{\boldsymbol{\phi}}(\boldsymbol{x}^i)), \tag{6}$$

where $\mathcal{L}(\cdot, \cdot)$ measures the difference between the output predictions and the true labels. Once the model $\mathcal{F}_{\boldsymbol{\phi}}$ is trained, we can use it to compute the prediction of new samples and make decisions.

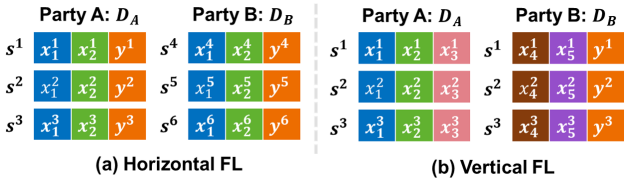


Fig. 1. Comparison of HFL and VFL frameworks, where s^i represents the i^{th} record, x_j^i and y^i represent the j^{th} feature and the label of the i^{th} record, respectively.

3.2 Vertical Federated Learning

The basic idea of FL is to collaboratively train the ML models by a number of local parties, where only model updates are shared during the training process and local data are never uploaded. According to the partition strategies of local data, existing FL frameworks can be further categorized into HFL and VFL [49]. A comparison between HFL and VFL is presented in Figure 1.

Consider FL for a two-party scenario, where party A and B respectively hold local datasets $\mathcal{D}_A$ and $\mathcal{D}_B$. Let $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{S}$ be the feature, label, and sample space. In the HFL setting, each party holds data from a different set of samples, while these data share the same feature space, which can be expressed as

$$\mathcal{X}_A = \mathcal{X}_B, \quad \mathcal{Y}_A = \mathcal{Y}_B, \quad \mathcal{S}_A \neq \mathcal{S}_B. \tag{7}$$

In contrast, in the VFL setting, each party holds different features of the same set of samples. Moreover, the labels are held by only one *active party*, while the other *passive parties* only provide inputs of additional features. In Figure 1, we have A and B as the passive and active parties, respectively. Such VFL systems can be expressed as

$$\mathcal{X}_A \neq \mathcal{X}_B, \quad \mathcal{Y}_A = \emptyset, \quad \mathcal{Y}_B = \mathcal{Y}, \quad \mathcal{S}_A = \mathcal{S}_B. \tag{8}$$

Besides the local training parties, existing VFL frameworks [12, 17, 48] usually involve an additional trusted third-party coordinator, which can be for instance a trusted execution enclave. During the inference phase, the coordinator is responsible for privately collecting and aggregating intermediate results of all local parties, computing the confidence scores, and sending them to the active party, as demonstrated in Figure 2. Later studies, such as [51], also proposed to remove the coordinator, so that the intermediate results of the passive party are directly sent to the active party for calculating the prediction results. In this paper, we mainly focus on privacy leakage under the first scenario, where the active party only uses the confidence scores to reconstruct the inputs of the passive party. Intuitively,

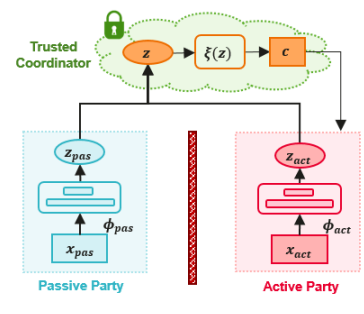


Fig. 2. Inference process of VFL systems with a coordinator.

this scenario provides higher privacy protection than the without-coordinator scenario since the third-party coordinator prevents the attacker from directly accessing the intermediate results.

4 Attack Methodologies

In this section, we present our reconstruction attack on VFL frameworks during prediction. First, we introduce the threat model, which can be divided into the *white-box* and *black-box* settings. Then, we describe the proposed attack methods for both settings in detail.

4.1 Threat Model

In this paper, we consider the scenario where two parties holding non-overlapping features of the same set of samples aim to jointly train a classification model under the VFL framework. As described in Section 3.2, the party with the real labels is called the *active party* $\mathcal{P}_{act}$, whereas the other party that only provides features is called the *passive party* $\mathcal{P}_{pas}$. Let $\boldsymbol{\phi}_{act}$ and $\boldsymbol{\phi}_{pas}$ be the models respectively held by $\mathcal{P}_{act}$ and $\mathcal{P}_{pas}$. The entire model trained under VFL is denoted as $\boldsymbol{\phi} = [\boldsymbol{\phi}_{act}, \boldsymbol{\phi}_{pas}]$. Similarly, each data record $\boldsymbol{x}$ can be presented as $\boldsymbol{x} = [\boldsymbol{x}_{act}, \boldsymbol{x}_{pas}]$, where $\boldsymbol{x}_{act}$ and $\boldsymbol{x}_{pas}$ are the inputs from $\mathcal{P}_{act}$ and $\mathcal{P}_{pas}$. The inputs $\boldsymbol{x}_{act}$ and $\boldsymbol{x}_{pas}$ have d_{act} and d_{pas} features, respectively. During the prediction phase, given a new data record $\boldsymbol{x}$, $\mathcal{P}_{act}$ and $\mathcal{P}_{pas}$ calculate the intermediate result $\boldsymbol{z}_{act} = f(\boldsymbol{x}_{act}, \boldsymbol{\phi}_{act})$ and $\boldsymbol{z}_{pas} = f(\boldsymbol{x}_{pas}, \boldsymbol{\phi}_{pas})$ and send them to the coordinator. For LR models, the intermediate results $\boldsymbol{z}_{act}, \boldsymbol{z}_{pas}$ are directly calculated as the inner product between the local inputs and the local model parameters. For multi-layer NNs, we assume that $\boldsymbol{\phi}_{act}$ and $\boldsymbol{\phi}_{pas}$ are NNs without the nonlinear activation function on the output layer. The intermediate results $\boldsymbol{z}_{act}, \boldsymbol{z}_{pas}$ are the output of each local model. After receiving $\boldsymbol{z}_{act}$ and $\boldsymbol{z}_{pas}$ from $\mathcal{P}_{act}$ and $\mathcal{P}_{pas}$, the coordinator applies the nonlinear activation function $\xi(\cdot)$ to the aggregated intermediate result and calculates the confidence score $\boldsymbol{c}$ as

$$\boldsymbol{c} = \xi(\boldsymbol{z}) = \xi(f(\boldsymbol{x}_{act}, \boldsymbol{\phi}_{act}) + f(\boldsymbol{x}_{pas}, \boldsymbol{\phi}_{pas})). \quad (9)$$

Finally, the confidence score $\boldsymbol{c}$ will be returned to $\mathcal{P}_{act}$.

We assume the attacker to be the active party $\mathcal{P}_{act}$, which uses the predicted confidence score $\boldsymbol{c}$ of each *individual* target record $\boldsymbol{x}$ to reconstruct the corresponding passive-party input $\boldsymbol{x}_{pas}$. We further distinguish the attacks between the *white-box setting* and the *black-box setting*:

- For the *white-box setting*, $\mathcal{P}_{act}$ has full access to the entire model $\boldsymbol{\phi} = [\boldsymbol{\phi}_{act}, \boldsymbol{\phi}_{pas}]$, the confidence scores $\boldsymbol{c}$, and its own features $\boldsymbol{x}_{act}$. The goal is to estimate $\boldsymbol{x}_{pas}$ via certain attack algorithms $\mathcal{A}_{wb}$ as follows:

$$\hat{\boldsymbol{x}}_{pas} = \mathcal{A}_{wb}(\boldsymbol{x}_{act}, \boldsymbol{\phi}_{act}, \boldsymbol{\phi}_{pas}, \boldsymbol{c}). \quad (10)$$

- For the *black-box setting*, $\boldsymbol{\phi}_{pas}$ are not revealed to $\mathcal{P}_{act}$ due to privacy and intellectual property issues. We slightly relax the setting by assuming that $\mathcal{P}_{act}$ is aware of a small set of passive-party data, which are also referred to as *auxiliary data*. These auxiliary data and their confidence scores can be denoted as $\boldsymbol{x}^{aux} = [\boldsymbol{x}^{aux}_{act}, \boldsymbol{x}^{aux}_{pas}]$ and $\boldsymbol{c}^{aux}$. The attack algorithm $\mathcal{A}_{bb}$ can be expressed as

$$\hat{\boldsymbol{x}}_{pas} = \mathcal{A}_{bb}(\boldsymbol{x}_{act}, \boldsymbol{\phi}_{act}, \boldsymbol{c}, \boldsymbol{x}^{aux}, \boldsymbol{c}^{aux}). \quad (11)$$

4.2 Baseline Attacks

We first briefly describe the baseline attacks in [28] for LR models and multi-layer NNs. Both attacks assume that the active party has full access to $\boldsymbol{\phi}_{act}$, $\boldsymbol{\phi}_{pas}$, $\boldsymbol{x}_{act}$, and $\boldsymbol{c}$ and aims to reconstruct $\boldsymbol{x}_{pas}$.

4.2.1 Equation Solving Attack

The ESA attack is proposed by [28] against LR models. For the LR model with k classes, the model parameters of $\mathcal{P}_{act}$ and $\mathcal{P}_{pas}$ can be respectively vectorized as $\boldsymbol{\phi}_{act} = [\boldsymbol{\phi}_{1_{act}}, \cdots, \boldsymbol{\phi}_{k_{act}}]$ and $\boldsymbol{\phi}_{pas} = [\boldsymbol{\phi}_{1_{pas}}, \cdots, \boldsymbol{\phi}_{k_{pas}}]$. Given a data record $\boldsymbol{x} = [\boldsymbol{x}_{act}, \boldsymbol{x}_{pas}]$, the intermediate result $\boldsymbol{z} = [z_1, \cdots, z_k]$ is a k-dimensional vector where each entry z_m is calculated as

$$z_m = \boldsymbol{\phi}_{m_{act}} \cdot \boldsymbol{x}_{act} + \boldsymbol{\phi}_{m_{pas}} \cdot \boldsymbol{x}_{pas}. \quad (12)$$

The confidence score $\boldsymbol{c}$ is calculated as $\boldsymbol{c} = \xi(\boldsymbol{z})$, where ξ is the *softmax* function (see Equation (3)). A relationship between $\boldsymbol{c}$ and $\boldsymbol{z}$ can be further derived as below:

$$\ln c_{m+1} - \ln c_m = z_{m+1} - z_m. \tag{13}$$

Let $\boldsymbol{\Phi}_m = \boldsymbol{\phi}_{m+1} - \boldsymbol{\phi}_m$, we further have

$$\boldsymbol{\Phi}_{m_{pas}} \cdot \boldsymbol{x}_{pas} = (\ln c_{m+1} - \ln c_m) - \boldsymbol{\Phi}_{m_{act}} \cdot \boldsymbol{x}_{act}. \tag{14}$$

Obviously, for the LR model with k classes, we get $k-1$ linear equations. Since $\boldsymbol{\Phi}_{m_{act}}$, $\boldsymbol{x}_{act}$, c_m and c_{m+1} are all known to the active party, we can substitute the right-hand side of Equation (14) with $\boldsymbol{\Psi}$. Thus, the equation can be rewritten as

$$\boldsymbol{\Phi}_{m_{pas}} \cdot \boldsymbol{x}_{pas} = \boldsymbol{\Psi}, \tag{15}$$

where $\boldsymbol{\Phi}_{m_{pas}}$ is a matrix of size $(k-1) \times d_{pas}$, $\boldsymbol{x}_{pas}$ and $\boldsymbol{\Psi}$ are vectors of size d_{pas} and $k-1$. It can be easily seen that when the number of unknown passive-party features is less than the number of linear equations, *i.e.*, $d_{pas} \leq k-1$, the system will have a unique solution of $\hat{\boldsymbol{x}}_{pas}$. If $d_{pas} > k-1$, there will be an infinite number of solutions to Equation (15). Nevertheless, the attacker can still estimate the target features by solving $\hat{\boldsymbol{x}}_{pas} = \boldsymbol{\Phi}_{pas}^{+} \cdot \boldsymbol{\Psi}$, where $\boldsymbol{\Phi}_{pas}^{+}$ is the pseudo-inverse matrix of $\boldsymbol{\Phi}_{m_{pas}}$. The obtained $\hat{\boldsymbol{x}}_{pas}$ minimizes $\|\boldsymbol{\Phi}_{m_{pas}} \cdot \boldsymbol{x}_{pas} - \boldsymbol{\Psi}\|_2$ and is the solution with the minimum l_2-norm, namely, $\|\hat{\boldsymbol{x}}_{pas}\|_2 \leq \|\boldsymbol{x}_{pas}\|_2$.

4.2.2 Generative Regression Network Attack

For multi-layer NNs, the activation function on each hidden layer introduces nonlinearity into the model, thereby causing the ESA attack not to be applicable. Therefore, a GRN attack was proposed [28]. The attack trains a generator model $\boldsymbol{\phi}_G$ to learn the correlation between the features and to "generate" the estimated passive-party features of the new target records. During each iteration, the generator $\boldsymbol{\phi}_G$ inputs the active-party features $\boldsymbol{x}_{act}$ and a set of random variables $\boldsymbol{\delta}_{pas}$ and produces the estimated passive-party features $\boldsymbol{x}_{pas}$. Then, $\boldsymbol{x}_{pas}$ with $\boldsymbol{x}_{act}$ will be fed into $\boldsymbol{\phi} = [\boldsymbol{\phi}_{act}, \boldsymbol{\phi}_{pas}]$ and generate the estimated confidence score $\hat{\boldsymbol{c}}$. Finally, a distance between $\hat{\boldsymbol{c}}$ and the real confidence score $\boldsymbol{c}$ will be calculated and used to update the generator model. While the GRN attack fills the gap created by the ESA attack with respect to complex NNs, training the generator model requires the collection of an adequate number of prediction results, *e.g.*, more than 1000 records as mentioned in [28]. For instance, in the scenario where a bank collaborates with an e-commerce company to jointly train a model for predicting the clients' default risk, the bank may need several weeks or months to collect the prediction results from enough clients before the attack can be applied. This may not always be practicable in real life.

4.3 Gradient-based Inversion Attack

4.3.1 GIA Under White-box Setting

To address the limitations of both baseline attacks, we propose a gradient-based inversion attack (GIA). The main idea is to search for the optimal estimation of passive-party input $\hat{\boldsymbol{x}}_{pas}$ within the feature space $\mathcal{X}_{pas}$ that produces an estimated confidence score $\hat{\boldsymbol{c}}$ close enough to the real confidence score $\boldsymbol{c}$, namely

$$\hat{\boldsymbol{x}}_{pas} = \underset{\mathcal{X}_{pas}}{\operatorname{argmin}}\, \mathcal{D}(\boldsymbol{c}, \hat{\boldsymbol{c}}), \tag{16}$$

where $\mathcal{D}(\cdot, \cdot)$ is a metric for measuring the distance between $\boldsymbol{c}$ and $\hat{\boldsymbol{c}}$. The main workflow is presented in Figure 3 and Algorithm 1. Given the real confidence score $\boldsymbol{c}$, the attacker $\mathcal{P}_{act}$ first randomly initializes the estimated passive-party input $\hat{\boldsymbol{x}}_{pas}$ and computes the estimated confidence score $\hat{\boldsymbol{c}}$ as

$$\hat{\boldsymbol{c}} = \xi(f(\boldsymbol{x}_{act}, \boldsymbol{\phi}_{act}) + f(\hat{\boldsymbol{x}}_{pas}, \boldsymbol{\phi}_{pas})), \tag{17}$$

where $\boldsymbol{\phi}_{act}$ and $\boldsymbol{\phi}_{pas}$ are model parameters of $\mathcal{P}_{act}$ and $\mathcal{P}_{pas}$, respectively, and $\boldsymbol{x}_{act}$ is the input of $\mathcal{P}_{act}$. Then, the distance between the real and estimated confidence scores $\boldsymbol{c}$ and $\hat{\boldsymbol{c}}$ is calculated. Here, we use two metrics, the mean squared error (MSE) and the KL divergence (KLD), to measure the difference between $\boldsymbol{c}$ and $\hat{\boldsymbol{c}}$, which are defined as

$$\mathcal{D}_{MSE}(\boldsymbol{c}, \hat{\boldsymbol{c}}) = \frac{1}{k} \sum_{m=1}^{m=k} (\boldsymbol{c}_m - \hat{\boldsymbol{c}}_m)^2, \tag{18}$$

$$\mathcal{D}_{KLD}(\boldsymbol{c}, \hat{\boldsymbol{c}}) = \sum_{m=1}^{m=k} \boldsymbol{c}_m \cdot \log(\frac{\hat{\boldsymbol{c}}_m}{\boldsymbol{c}_m}). \tag{19}$$

Finally, $\mathcal{D}$ is used to update $\hat{\boldsymbol{x}}_{pas}$ as

$$\hat{\boldsymbol{x}}_{pas} \leftarrow \textsc{Opt}.update(\hat{\boldsymbol{x}}_{pas}, \mathcal{D}, \lambda). \tag{20}$$

The Opt can be any common ML optimizer such as stochastic gradient descent (SGD) [4] and Adam [23], which updates the current $\hat{\boldsymbol{x}}_{pas}$ according to the distance $\mathcal{D}$ and a predefined learning rate λ. Here, we use

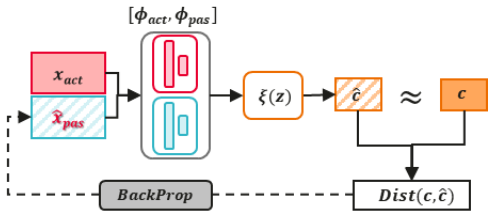


Fig. 3. Workflow of the proposed GIA attack under the white-box setting, where the entire model $\phi = [\phi_{act}, \phi_{pas}]$ is revealed to the attacker. Given the confidence score $\boldsymbol{c}$, the goal is to find an optimal estimated passive-party input $\hat{\boldsymbol{x}}_{pas}$ that produces an estimated confidence score $\hat{\boldsymbol{c}}$ close enough to $\boldsymbol{c}$.

Algorithm 1: GIA attack

Input: ϕ_{act}, ϕ_{pas}: active- and passive-party models; $\boldsymbol{x}_{act}$: active-party input; $\boldsymbol{c}$: confidence score; OPT: optimizer for updating the estimated input; T: rounds of optimization; λ: learning rate of optimization

Output: $\hat{\boldsymbol{x}}_{pas}$: estimated passive-party input

Initialize estimated input $\hat{\boldsymbol{x}}_{pas} = [0, \cdots, 0]$
for round $t = 1, \cdots, T$ **do**
  $\hat{\boldsymbol{c}} = \xi(f(\boldsymbol{x}_{act}, \phi_{act}) + f(\hat{\boldsymbol{x}}_{pas}, \phi_{pas}))$
  $\mathcal{D} = \mathcal{D}_{MSE}(\boldsymbol{c}, \hat{\boldsymbol{c}})$ or $\mathcal{D} = \mathcal{D}_{KLD}(\boldsymbol{c}, \hat{\boldsymbol{c}})$
  $\hat{\boldsymbol{x}}_{pas} \leftarrow$ OPT.$update(\hat{\boldsymbol{x}}_{pas}, \mathcal{D}, \lambda)$
end for
Return $\hat{\boldsymbol{x}}_{pas}$

the Adam optimizer because of its fast convergence performance and use a default learning rate $\lambda = 0.001$. We repeat the optimization for T rounds until $\hat{\boldsymbol{c}}$ is close enough to $\boldsymbol{c}$, *i.e.*, until the distance $\mathcal{D}$ approaches 0.

Our GIA attack achieves the following advantages compared to both baseline attacks. First, the attack is model-agnostic and can be applied to both simple LR models and complex multi-layer NNs. Second, the attack adopts a gradient-based method to iteratively estimate $\boldsymbol{x}_{pas}$, which can achieve a closer approximation of the real input values compared to the ESA attack. Moreover, the GIA attack eliminates the requirement for collecting a large number of predictions in the GRN attack. Thus, it can be applied independently and flexibly to any individual records.

4.3.2 GIA Under Black-box Setting

In real-life scenarios, ϕ_{pas} may not always be revealed to $\mathcal{P}_{act}$ because of intellectual property and privacy issues. Therefore, we further investigate the attack under a black-box setting. We slightly relax the setting by assuming that $\mathcal{P}_{act}$ has prior knowledge of a small set of auxiliary data $\boldsymbol{x}^{aux} = [\boldsymbol{x}^{aux}_{act}, \boldsymbol{x}^{aux}_{pas}]$ and their confidence scores $\boldsymbol{c}^{aux}$. For instance, $\mathcal{P}_{act}$ can collude with a few internal employees (whose data are saved by both parties) and use their data as auxiliary data. Moreover, $\mathcal{P}_{act}$ can manually construct a few fake data, instead of using the real user data, to perform the attack. These auxiliary data can then be used to build a shadow model $\hat{\phi}_{pas}$ that imitates the performance of the real ϕ_{pas}. The main workflow is presented in Algorithm 2. Given a set of auxiliary data and corresponding confidence scores $(\boldsymbol{x}^{aux}, \boldsymbol{c}^{aux})$, we first initialize the shadow model $\hat{\phi}_{pas}$. Then, we compute the confidence scores predicted by the current shadow model $\hat{\boldsymbol{c}}^{aux}$ and calculate $\mathcal{D}^{aux}$, which measures the distance between $\boldsymbol{c}^{aux}$ and $\hat{\boldsymbol{c}}^{aux}$. The distance $\mathcal{D}^{aux}$ will then be used to update the shadow model. The goal of the optimization is to estimate the real passive-party model parameters by minimizing the difference between $\boldsymbol{c}^{aux}$ and $\hat{\boldsymbol{c}}^{aux}$. Once the shadow model $\hat{\phi}_{pas}$ is obtained, $\mathcal{P}_{act}$ can use the model to perform Algorithm 1 for the reconstruction attack based on new prediction records as in the white-box setting.

The next question is, "how many auxiliary data are enough for constructing a satisfying shadow model?". We start the analysis with an LR model. Let d_{pas} be the number of passive-party features and k be the number of classes. The total number of unknown model parameters is thus $d_{pas} \cdot k$. Assuming $\mathcal{P}_{act}$ is aware of n auxiliary data records, he/she can use $n \cdot k$ predicted values to solve the linear equation system. To ensure that the equation system has a unique solution, it should satisfy

$$n \cdot k \geq k \cdot d_{pas} \quad \Rightarrow \quad n \geq d_{pas} = \tau. \tag{21}$$

Here, the threshold τ is the lower bound of the number of auxiliary data n to obtain an attack performance similar to that of the white-box setting. Note that the black-box attack can still be conducted with fewer auxiliary data, but it may result in a larger reconstruction error. Thus, τ can be used as a metric that estimates the auxiliary data needed to obtain satisfying attack performance. Moreover, it can be easily observed that an increasing number of unknown passive-party features d_{pas} leads to a larger τ.

For complex models such as multi-layer NNs, although we cannot derive the exact threshold for n, we can follow Equation (21) to derive an approximated threshold. Let $|\phi_{pas}|$ be the number of unknown pa-

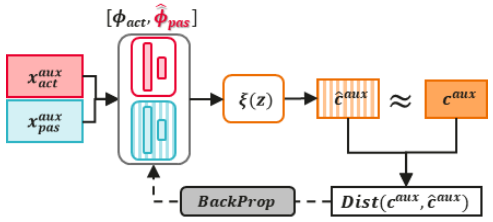


Fig. 4. Workflow of training a shadow model $\hat{\phi}_{pas}$ using the auxiliary data $(\boldsymbol{x}^{aux}, \boldsymbol{c}^{aux})$, which mimics the performance of the real passive-party model ϕ_{pas}.

Algorithm 2: Training the shadow model

Input: ϕ_{act}: active-party model; $\boldsymbol{x}^{aux} = [\boldsymbol{x}_{act}^{aux}, \boldsymbol{x}_{pas}^{aux}]$: auxiliary data; $\boldsymbol{c}^{aux}$: confidence score of $\boldsymbol{x}^{aux}$; OPT: optimizer for updating the shadow model; T: rounds of optimization; λ: learning rate of optimization

Output: $\hat{\phi}_{pas}$: shadow passive-party model

1: Initialize shadow model $\hat{\phi}_{pas}$
2: **for** round $t = 1, \cdots, T$ **do**
3: $\quad \hat{\boldsymbol{c}}^{aux} = \xi(f(x_{act}^{aux}, \phi_{act}) + f(x_{pas}^{aux}, \hat{\phi}_{pas}))$
4: $\quad \mathcal{D}^{aux} = \mathcal{D}_{MSE}(\boldsymbol{c}^{aux}, \hat{\boldsymbol{c}}^{aux})$ or $\mathcal{D}^{aux} = \mathcal{D}_{KLD}(\boldsymbol{c}^{aux}, \hat{\boldsymbol{c}}^{aux})$
5: $\quad \hat{\phi}_{pas} \leftarrow \text{OPT}.update(\hat{\phi}_{pas}, \mathcal{D}^{aux}, \lambda)$
6: **end for**
7: **Return** $\hat{\phi}_{pas}$

rameters of the passive-party model. Then, we have

$$n \cdot k \geq |\phi_{pas}| \quad \Rightarrow \quad n \geq \frac{|\phi_{pas}|}{k} = \tau. \tag{22}$$

However, since the activation functions in multi-layer NNs introduce nonlinearity to the model, the real required number of auxiliary data might be larger.

5 Experiments

5.1 Experimental Setup

5.1.1 Models and Datasets

Here, we investigate the performance of our proposed attack against LR models and multi-layer NNs. For the experiments with LR models, each party holds the model weights of the corresponding local features. For the experiments with NNs, each party holds a multi-layer model. The model takes the number of local features as input size and the number of classes as output size. In the experiments, we compare the attack performance against local models with respectively one, two, and three hidden layers, where each layer has eight neurons and uses *sigmoid* (Equation (2)) as the activation function. After the model is trained, we conduct the proposed reconstruction attacks during the inference phase.

We perform comprehensive privacy-utility evaluations over four widely-used public datasets, which are used for binary and multi-class classification tasks:

- **Bank** [31]: The dataset contains 45211 records of contact history of a Portuguese banking institution, which are used to predict if customers will subscribe to a term deposit by telemarketing. Each data record consists of 20 features representing customers' personal information, contact details, *etc.*
- **Robot** [14]: The dataset contains records of a robot navigating through a room. Each record has 24 attributes, representing the data collected by 24 ultrasound sensors. The records are categorized into four classes.
- **Satellite** [10]: The dataset consists of the coded multi-spectral values of satellite images, which are categorized into six classes. The goal is to use the multi-spectral values to predict the type of land.
- **Drive** [10]: The dataset contains 58,509 records. Each record has 48 features that are extracted from the electric current drive signals, which are classified into 11 different conditions.

Table 1. Details of datasets

Dataset	#Feature	#Class	#Records	Accuracy	
				LR	NN
Bank	20	2	41,188	0.9109	0.9506
Robot	24	4	5,456	0.6710	0.8946
Satellite	36	6	6,430	0.8152	0.8275
Drive	48	11	58,509	0.8037	0.9241

Details of each dataset are shown in Table 1, including the number of features, classes, and records. Moreover, we normalize the feature values in each dataset into [0,1]. We further present the accuracy of LR models and two-layer NNs evaluated on each dataset in Table 1.

5.1.2 Baselines

In our experiments, we use the ESA attack and the GRN attack proposed in [28] as baselines for the LR models and multi-layer NNs. For the GRN attack, we use the same generator model as in [28]. Moreover, since both

baseline methods assume access to real trained models, we will not include them in our comparison for the attacks under the black-box setting.

5.1.3 Evaluation Metrics

We evaluate the performance of the proposed attack in terms of *attack error* and *attack accuracy*.

For *attack error*, we use the MSE to measure the distance between the original passive-party features $\boldsymbol{x}_{pas}$ and the reconstructed features $\hat{\boldsymbol{x}}_{pas}$, as in [28]. More specifically, given a batch of b records, we use the averaged MSE to measure the overall attack error in different settings, as shown below

$$\text{Error} = \frac{1}{b \cdot d_{pas}} \sum_{i=1}^{b} \sum_{j=1}^{d_{pas}} (\hat{x}_{pas}^{i,j} - x_{pas}^{i,j})^2, \tag{23}$$

where d_{pas} is the number of passive-party features, $x_{pas}^{i,j}$ and $\hat{x}_{pas}^{i,j}$ are respectively the original and reconstructed values of the j^{th} entry of the i^{th} records. Obviously, the smaller the MSE, the better the attack performance. An MSE $= 0$ means that the reconstructed input has the same values as the original input.

For *attack accuracy*, we analyze the prediction results of the reconstructed passive-party features in a classification task. To this end, we first train an evaluation classifier $\mathcal{M}$ using a set of *held-out* passive-party features and the ground truth labels. Then, we test the classifier with the original target features $\boldsymbol{x}_{pas}$ and the reconstructed features $\hat{\boldsymbol{x}}_{pas}$ and analyze whether $\boldsymbol{x}_{pas}$ and $\hat{\boldsymbol{x}}_{pas}$ have the same predicted labels. Given a batch of b records, the attack accuracy is calculated as

$$\text{Acc} = \frac{1}{b} \sum_{i}^{b} = \mathbb{1}(\operatorname{argmax} \mathcal{M}(\boldsymbol{x}_{pas}^{i}), \operatorname{argmax} \mathcal{M}(\hat{\boldsymbol{x}}_{pas}^{i})). \tag{24}$$

where $\operatorname{argmax} \mathcal{M}(\boldsymbol{x}_{pas}^{i})$ and $\operatorname{argmax} \mathcal{M}(\hat{\boldsymbol{x}}_{pas}^{i})$ are the predicted labels of the i^{th} real and reconstructed features. Intuitively, the higher the attack accuracy, the closer the reconstructed features to the original features. Here, we use the SGD classifier provided in the *scikit-learn* library [33] for analyzing the attack accuracy.

We define $r_{pas} = d_{pas}/(d_{act} + d_{pas})$ as the ratio of the passive-party features and analyze the attack performance under different r_{pas}. For each r_{pas}, we independently perform the attack against $b = 100$ records and calculate the average attack error and accuracy.

5.2 Results of White-box Attacks

We first present the performance of our reconstruction attack under the white-box setting. We conduct the attack on LR models and multi-layer NNs, and evaluate the proposed GIA attack on different datasets. For each experiment, we vary r_{pas} and analyze the corresponding attack performance.

5.2.1 White-box Attacks on LR Models

We first analyze the attack performance on LR models. For each dataset, we compare the attack error of our proposed GIA attack with the results of random guesses and the baseline ESA attack. The results are shown in the upper row of Figure 5. For the GIA attack, we analyze the attack performance by using the MSE and KLD as distance metrics, which is referred to as *GIA-MSE* and *GIA-KLD*, respectively. As discussed in Section 4.2.1, given the total number of class k, the passive-party input x_{pas} can be fully recovered if $d_{pas} \leq k - 1$. It can be seen from the results that, for both attacks, the attack error is 0 when the requirement holds. When d_{pas} increases, the attack error of both methods rises and gradually approaches the random guess. However, the attack error of our method is mostly smaller than that of the ESA attack. In particular, for the **Drive** and **Satellite** datasets, when $r_{pas} = 0.9$, the attack error of our method is reduced by twice or three times compared with that of the ESA attack. Finally, we observe that for our attack, using the MSE or KLD as the distance metric does not have much influence on attack performance. Thus, we use the MSE for the remainder of the experiments.

We further analyze the attack accuracy of the proposed attack. For each dataset, we compare the attack accuracy of our GIA attack with the baseline methods under different r_{pas}. The results are presented in the bottom row of Figure 5. Intuitively, the higher the attack accuracy, the closer the reconstructed features are to the real features. For all the datasets, the attack accuracy is 1 when $d_{pas} \leq k - 1$, thereby indicating that the passive-party features are fully reconstructed. Moreover, the attack accuracy of each dataset decreases with an increase in r_{pas}, which implies that the reconstruction gets more difficult with an increase of unknown features. Additionally, it can be observed that our proposed attack achieves higher attack accuracy compared to the baseline method, which also demonstrates the better performance of our method.

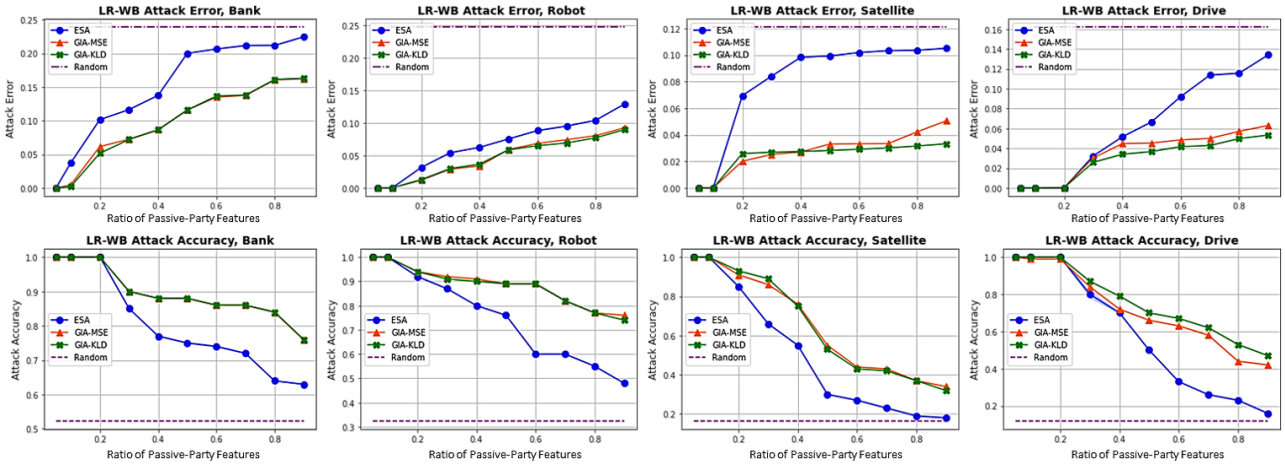
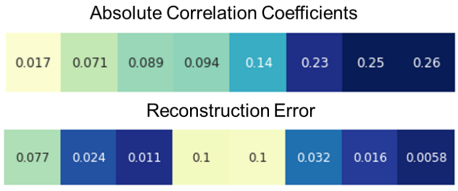
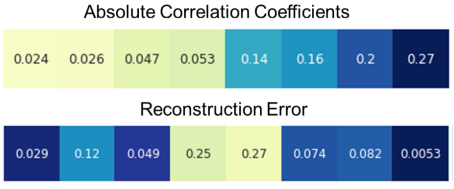


Fig. 5. Attack error (upper row) and attack accuracy (bottom row) of the proposed *GIA* attack on LR models under the white-box setting, compared with random guess and the baseline *ESA* attack.



(a) Comparison result of the **Bank** dataset.



(b) Comparison result of the **Drive** dataset.

Fig. 6. Impact of the correlation between the passive-party and active-party features on attack performance. Each cell corresponds to the absolute correlation or reconstruction error of one passive-party feature.

5.2.1.1 Attack Performance: Feature Correlations

In real-world datasets, features may be correlated with each other. For instance, in the **Bank** dataset, each feature collected by one sensor may be related to the other features. Therefore, we further analyze the impact of the feature correlations on attack performance. More specifically, for each passive-party feature $x^i_{pas} \in \boldsymbol{x}_{pas}$, we compute the average absolute correlation of x^i_{pas} to the active-party features $\boldsymbol{x}_{act}$ as

$$Corr(x^i_{pas}, \boldsymbol{x}_{act}) = \frac{1}{d_{act}} \sum_{j=1}^{d_{act}} abs(\rho(x^i_{pas}, x^j_{act})), \quad (25)$$

where x^j_{act} is the j^{th} active-party feature, $\rho(a, b)$ is the Pearson correlation coefficient between a and b, $abs(\cdot)$ is the absolute function and d_{act} is the number of active-party features. We conduct experiments on the **Bank** and **Drive** datasets with $r_{pas} = 0.3$ and visualize the absolute correlation and attack error of the first eight features, as presented in Figure 6. It can be seen that for both datasets, the passive-party features with higher correlation usually achieve a relatively lower attack error. The results indicate that the reconstruction attack can be more successful on the passive-party features that are correlated with the features owned by the attacker, which can lead to an increased risk of privacy leakage.

5.2.2 White-box Attacks on NNs

In addition to LR models, we also analyze the performance of the white-box attack on multi-layer NNs. First, we compare the attack error of our GIA attack with the baseline GRN attack. Here, we assume both the active- and passive-party models are NNs with two hidden layers. Note that the GRN attack requires enough prediction records for training the attacker model. Therefore, we use 100 prediction records in each experiment for a fair comparison. The results are shown in Figure 7. It can be observed that the attack error is close to 0 when $r_{pas} \leq 0.1$, and it increases when r_{pas} gets larger. Moreover, the attack error of the GIA attack is consistently lower than that of the GRN attack, thereby indicating the advantage of our attack in comparison to the baseline. We further present the

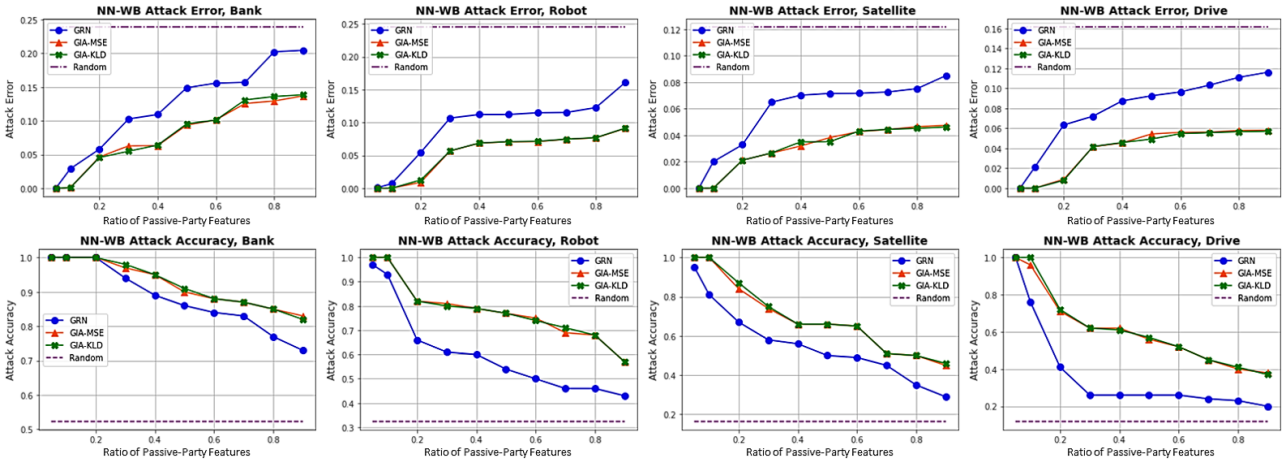


Fig. 7. Attack error (upper row) and attack accuracy (bottom row) of the proposed *GIA* attack against NNs under the white-box setting, compared with random sampling and the baseline *GRN* attack.

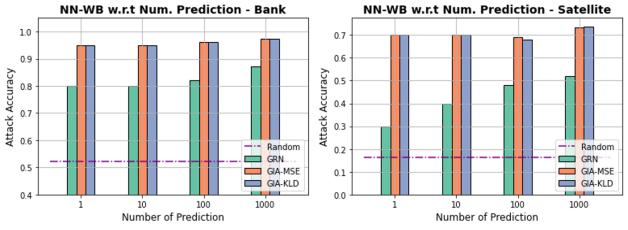


Fig. 8. Attack accuracy of the baseline GRN attack and the proposed GIA attack using different number of prediction records. The attack is conducted under $r_{pas} = 0.5$.

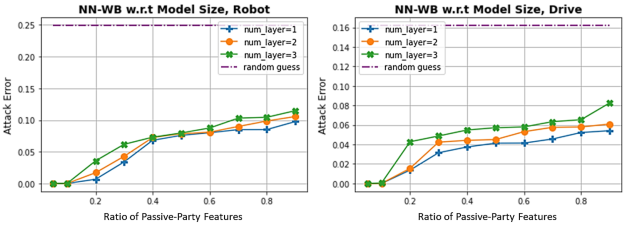


Fig. 9. Attack performance with respect to model size. We respectively compare the attack performance against NNs with one, two and three hidden layers.

comparison of the attack accuracy of different attack methods. We observe that the GIA attacks can precisely reconstruct target features with small r_{pas}. Even for larger r_{pas}, our attack can still achieve higher attack accuracy compared to the baseline method.

5.2.2.1 Attack Performance: Number of Prediction Records

As mentioned earlier, the GRN attack should collect a batch of prediction records for satisfactory attack performance. In contrast, our GIA attack can be applied to any single prediction. Thus, we also analyze how both attacks perform with a different number of prediction records. To this end, we conduct both attacks using 1, 10, 100, 1000 prediction records under $r_{pas} = 0.5$ and compare the change in attack accuracy. We present the comparison results for the **Bank** and **Satellite** datasets, as shown in Figure 8. It can be seen that the GRN attack obtains relatively lower attack accuracy with fewer prediction records, whereas our proposed attack can still achieve satisfactory performance. The results indicate that our proposed attack relaxes the multi-prediction requirement of the baseline attack against NN models and effectively applies to scenarios where only few prediction records are available.

5.2.2.2 Attack Performance: Model Size

Next, we conduct experiments to analyze the impact of model size on attack performance. For each dataset, we compare the attack error against local models with one, two, and three hidden layers. For each model, we use *sigmoid* as the activation function. The results are presented in Figure 9. It can be seen that for the same r_{pas}, the error gets larger with an increase in hidden layers. The result can be attributed to the fact that the activation functions introduce nonlinearity to the model. Thus, the more hidden layers, the more difficult it is to perform the reconstruction attack. Even in this case, it can still be observed that the attack error of our proposed attack is distinctively smaller than random guessing, which indicates that our attack poses significant privacy risks against different model sizes.

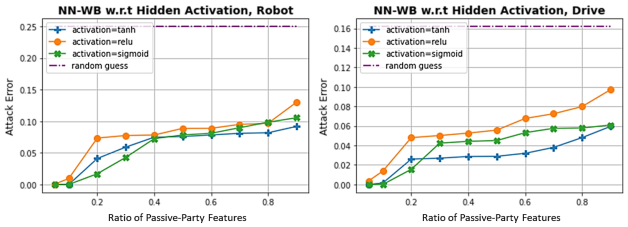
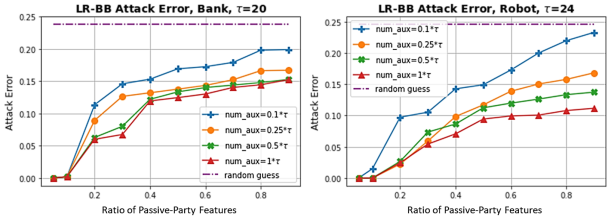
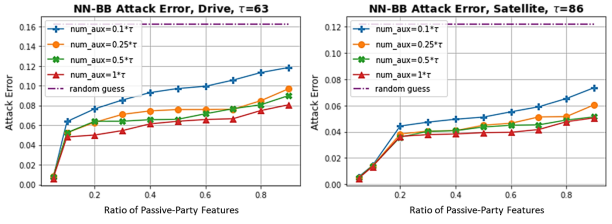


Fig. 10. Attack performance against NNs with respect to different activation functions (*relu*, *sigmoid* and *tanh*).



(a) Attack error against LR models under the black-box setting.



(b) Attack error against NNs under the black-box setting.

Fig. 11. Attack error on LR models and NNs under the black-box setting with $n \in \{0.1\tau, 0.25\tau, 0.5\tau, \tau\}$, where τ is the threshold of the auxiliary records and n is the actual number of auxiliary records used during the attacks.

5.2.2.3 Attack Performance: Activation Functions

Apart from using *sigmoid* as the hidden-layer activation function, one can also use the *relu* and *tanh* functions. As such, we also investigate whether the choice of the activation function has a distinctive impact on attack performance. To this end, we apply the reconstruction attack on two-hidden-layer NNs with *sigmoid*, *tanh*, and *relu* as the activation functions and compare the results, as shown in Figure 10. It can be seen that the attack shows similar performance under different activation functions. Moreover, for most of the datasets, the attack error for NNs with *relu* is slightly larger than that with the *sigmoid* and *tanh* functions. The difference in the attack error may be because the *relu* function only outputs the input directly if it is positive; otherwise, it outputs zero. This may cause a certain amount of information loss during the inference phase, thereby making it more difficult to reconstruct the input features from the output confidence scores.

5.3 Results of Black-box Attacks

Next, we evaluate the attack performance under the black-box setting. We assume that the attacker is aware of a few auxiliary records in the prediction set, and he/she will use these data to build a shadow model for conducting the reconstruction attack. We evaluate the performance of the black-box attack on LR models and NNs, and different datasets. As before, for each experiment, we vary r_{pas} and compare the attack error.

We first analyze the performance of the black-box attack on LR models. For each dataset, we first calculate the threshold τ, *i.e.*, the required number of auxiliary records needed to achieve similar attack performance as in the white-box setting. As described in Section 4.3.2, the increase in d_{pas} (equivalently, r_{pas}) results in an increase in τ. Thus, in the experiments, we calculate τ with $r_{pas} = 1$, which is sufficient for building shadow models with any $r_{pas} \in (0, 1)$. Then, we conduct experiments respectively with $n \in \{0.1\tau, 0.25\tau, 0.5\tau, \tau\}$, where n is the actual number of auxiliary records used for the attack. In Figure 11a, we present the results for the **Bank** and **Robot** datasets. It can be seen that for each dataset, the attack error decreases with an increase in n. When $n = \tau$, the attack error is almost the same as for the white-box attack. Nevertheless, we can already achieve satisfactory attack performance when $n \geq 0.25\tau$, which is less than ten data records for both datasets.

We also analyze the attack performance against NNs with two hidden layers. We first use Equation (22) to estimate the threshold τ of NNs with $r_{pas} = 1$. Then, we conduct experiments with $n \in \{0.1\tau, 0.25\tau, 0.5\tau, \tau\}$. The results for the **Satellite** and **Drive** datasets are shown in Figure 11b. We observe that the attack performance improves with more auxiliary records. Further, the attack error is close to the results under the white-box setting when $n \geq 0.25\tau$, which is less than 25 records for both datasets. The experimental results indicate that even without access to the real passive-party model, the attacker can still conduct reconstruction attacks with limited prior knowledge.

5.3.1 Attack Performance: Poorly-selected Auxiliary Records

In previous experiments, we assume that the auxiliary records share a similar distribution as the target prediction records. We now investigate whether poorly-selected auxiliary records, such as records that are randomly sampled from different distributions, can affect

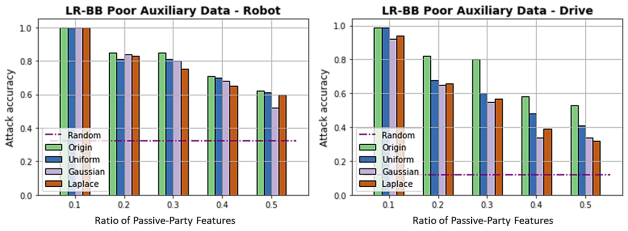


Fig. 12. Attack accuracy on LR models under the black-box setting using the auxiliary records sampled from different distributions, including the original, Uniform, standard Gaussian, and standard Laplace distribution.

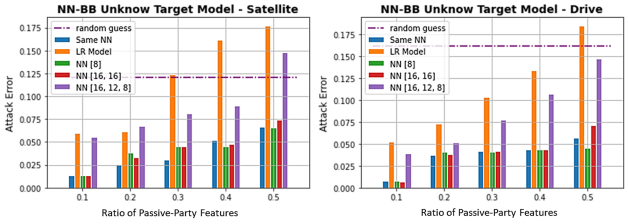


Fig. 13. Attack error under the black-box setting when the structure of the passive-party model is unknown. In the experiments, the real models are two-hidden-layer NNs. We respectively use the same NNs, LR models and some other multi-layer NNs as the guessed model structure and analyze the attack performance.

the attack performance. To this end, we manually construct a set of random auxiliary records, whose feature values are independently sampled from the uniform, standard Gaussian, and standard Laplace distributions and are not related to the real data. We use these random auxiliary records and their corresponding confidence scores to build the shadow model and apply the attack. We then compare the attack accuracy of using the auxiliary records sampled from the original and random distributions and present the results in Figure 12. It can be observed that using randomly sampled auxiliary records can decrease the attack accuracy. Nonetheless, the attack accuracy is still distinctively higher than the random guess. When r_{pas} is small, the attack performance is even close to using auxiliary data from the original distribution. As such, the results indicate that even when using randomly sampled auxiliary records under the black-box setting, the attacker may still be able to infer certain sensitive information about the passive-party features and cause potential privacy leakage.

5.3.2 Attack Performance: Prior Knowledge of Passive-party Model Structure

Finally, we investigate whether the knowledge of the passive-party model structure is essential for performing the reconstruction attack under the black-box setting. We let the *real* passive-party model be NNs with two hidden layers. Then, we build a list of shadow models with different structures, including LR models and NNs with different numbers of hidden layers and neurons. For each dataset, we conduct the black-box reconstruction attack on different shadow models with the same number of auxiliary records, *i.e.*, $n = \tau$. Our goal is to explore whether changing structures of shadow models will have a distinctive impact on attack performance. The results are presented in Figure 13. It can be seen that using LR models as shadow models usually causes a higher attack error. This may be because the activation functions applied in NNs' hidden layers result in nonlinear mapping between the input and output, which cannot be well-preserved in linear LR models. On the other hand, for the experiments that also use NNs as shadow models, we observe a similar attack performance when the shadow model has similar structures as the real model. More specifically, in our experiments, the real passive-party models are NNs with two hidden layers, each with eight neurons. For the same r_{pas}, we can achieve the smallest attack error when the shadow model is of the same structure as the real model. We can still reach a similar attack performance using shadow models with one and two hidden layers. However, the attack performance can degrade when the shadow model is much more complex than the real model, *e.g.*, when using three-hidden-layer NNs. This is because complex models increase the number of unknown model parameters and require more auxiliary data to achieve a satisfactory simulation of the real model.

6 Defenses Against the Attack

In this section, we conduct comprehensive experiments to investigate the capability of several potential defense techniques against reconstruction attacks. We will first briefly describe each defense technique and then show their performance with privacy-utility trade-off analyses.

6.1 Defense Techniques

6.1.1 Private Set Intersection

Private Set Intersection (PSI) [8, 13, 20] is a cryptography-based technique that allows two parties to compute the intersected elements in a private manner. The technique has been recently used in VFL for privately identifying the intersection of training samples from all parties [2, 26]. Here, we investigate whether PSI can help prevent the attack during the inference phase. To this end, we use the open-source framework PyVertical [36] to train models under the two-party VFL setting. Then, we simulate the attack during the inference phase and analyze the attack performance.

6.1.2 Differentially-private Training

DP [11], as a strong mathematical formalization of privacy, has recently shown its effectiveness in preventing membership and property inference attacks [5, 21]. Inspired by the success of DP in defending against inference attacks, we wonder whether the technique can also effectively prevent privacy leakage during inference. To this end, we train the model using the differentially-private stochastic gradient descent (DPSGD) algorithm [1]. During each training iteration, the algorithm clips the real gradients with a predefined $l2$-clipping bound γ and perturbs the clipped gradients with Gaussian noise. The noise scale is based on the clipping bound γ and the privacy budget ϵ. A smaller ϵ leads to a larger noise scale. In our experiments, we set $\gamma = 1$ and train DP models with $\epsilon \in \{0.1, 1, 10\}$. We then apply the reconstruction attack on all DP models and compare their defense capabilities.

6.1.3 Processing the Confidence Scores

6.1.3.1 Rounding Defense

Besides the protection techniques during the training phase, some other defenses can be directly applied to the predicted confidence scores before revealing them to the active party. The rounding defense is to approximate confidence scores to a limited number of decimals. In our experiments, we apply the attack on the confidence scores rounded to one or two decimal places as well as to integers (*i.e.*, only revealing the predicted label) and compare the attack performance with the unprotected results.

6.1.3.2 Noising Defense

In addition to the rounding defense, we investigate the performance of the noising defense, where a certain amount of random noise is added to the confidence scores. In our experiments, we perturb the confidence scores with the random noise sampled from a Gaussian distribution $\mathcal{N}(0, \sigma^2)$, where the noise scale σ equals 1, 0.1, and 0.01.

6.1.3.3 Purification Defense

Finally, we analyze the performance of the purification defense [52], which has recently been proposed against data inference attacks. The main idea is to reduce the dispersion of the predicted confidence scores using a pre-trained autoencoder, which is also referred to as a *purifier*. More specifically, the autoencoder can learn compressed representations of the inputs and helps the decoded outputs belonging to the same class to be tighter and indistinguishable from one another. Let $\mathcal{G}$ be the autoencoder, $\boldsymbol{c}$ and $\mathcal{G}(\boldsymbol{c})$ be the input and decoded confidence scores. In the original work [52], the autoencoder $\mathcal{G}$ is trained to minimize the following objective function:

$$\mathcal{L} = \mathcal{L}_1(\boldsymbol{c}, \mathcal{G}(\boldsymbol{c})) + \lambda \cdot \mathcal{L}_2(\operatorname{argmax} \boldsymbol{c}, \operatorname{argmax} \mathcal{G}(\boldsymbol{c})), \quad (26)$$

where $\mathcal{L}_1$ measures the l_2 distance between the input and the decoded output, $\mathcal{L}_2$ is the cross-entropy between the two predicted labels, and λ is a balancing coefficient.

In our experiments, we use autoencoders with one hidden layer for each dataset. The size of the hidden layer is half the input size. In addition, we find that using only $\mathcal{L}_1$ is already good enough to achieve lower attack accuracy while preserving the model accuracy. Note that the autoencoder in [52] is trained on a set of *real* confidence scores. However, in realistic settings, the autoencoder should be trained before the VFL training process, when the real confidence scores are unavailable. Therefore, we also construct a random training set, where the confidence scores for training the autoencoder are randomly sampled from the standard Gaussian distribution. We respectively compare the performance of the autoencoders trained on the real and random training set.

6.2 Privacy-utility Analysis

We implement the abovementioned defense techniques, and evaluate the privacy-utility performance of all datasets on LR models and NNs. For each dataset and

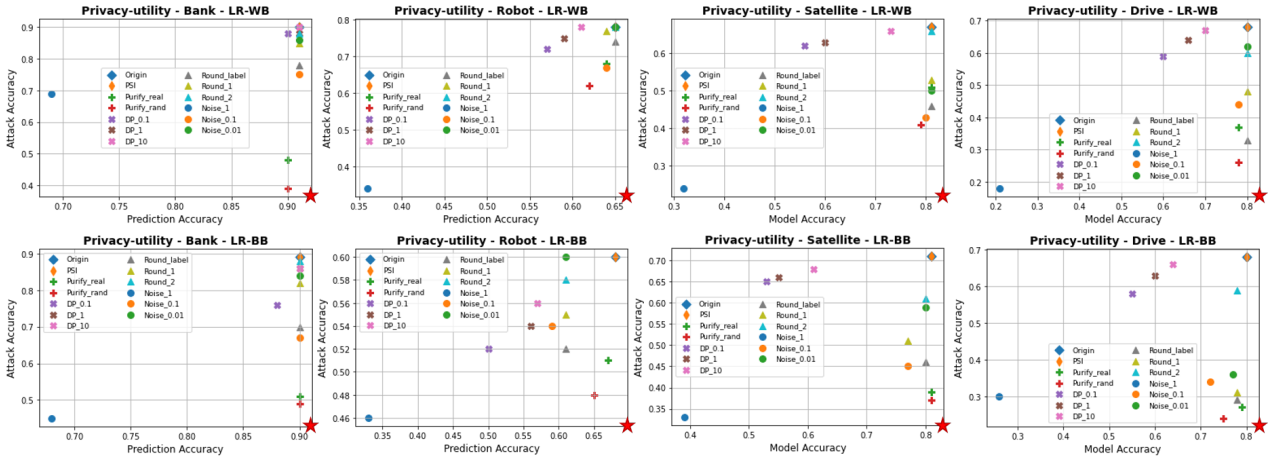


Fig. 14. Privacy-utility analysis on LR models under white-box (upper row) and black-box (lower row) setting with $r_{pas} = 0.5$. The ★ marker represents the ideal privacy-utility trade-off, where the model accuracy is high and attack accuracy is low.

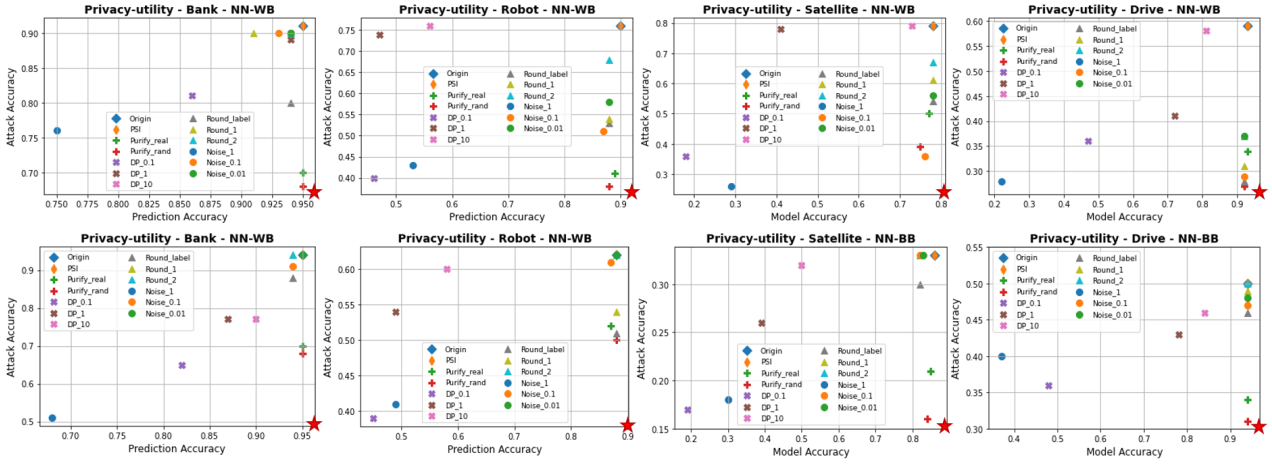


Fig. 15. Privacy-utility analysis on NNs under white-box (upper row) and black-box (lower row) setting with $r_{pas} = 0.5$. The ★ marker represents the ideal privacy-utility trade-off, where the model accuracy is high and attack accuracy is low.

model type, we compare the model and attack accuracy of all techniques under both white-box and black-box settings. Intuitively, an ideal defense technique should have distinctively lower attack accuracy while having no effects on model accuracy.

In Figure 14 and Figure 15, we present the privacy-utility analysis of the defense techniques against LR models and NNs. Here, we show the evaluation results with $r_{pas} = 0.5$. It can be seen that PSI is unable to prevent reconstruction attacks. This is because the goal of the protocol is to privately align the data instances that are held by both parties using crypto-based algorithms. However, during the prediction phase, the active party is still aware of the correspondence between the returned confidence scores and the input features. Thus, the attack can still take effect. Additionally, training the model with DP also fails to achieve a satisfactory privacy-utility balance against the proposed attack. It can be observed that the defense cannot effectively prevent the attack when the privacy budget is large, e.g., when $\epsilon = 10$. Although the attack accuracy reduces with a smaller ϵ, the model accuracy is unfavorably affected. This is because using DPSGD during training aims to protect the privacy of the training dataset. As such, the attack cannot easily infer any sensitive information of the training data from the model parameters or outputs. Since we focus on reconstructing the test data during the prediction phase, the technique is unsuitable for the proposed attack.

On the other hand, some defenses that directly process the confidence scores can also enhance privacy protection. Regarding the noising defense, adding noise with $\sigma = 0.1$ achieves a relatively good privacy-utility balance on both datasets. However, too much noise may

lead to significant utility degradation. As for the rounding defense, only returning the label instead of confidence scores achieves a distinctive reduction in attack accuracy, and it does not affect the model accuracy. Finally, we observe that purifying the confidence scores could also provide satisfactory privacy protection while maintaining negligible utility loss. Additionally, using randomly-sampled data to train the purifier can achieve an even better privacy-utility trade-off in comparison to using real data. This further relaxes the requirements in the original algorithm and makes the technique more practical in real-life scenarios.

7 Limitations and Future Work

Here, we will discuss the limitations of this work and potential future research directions.

In this paper, we use the attack error and attack accuracy to evaluate attack performance. Nevertheless, there are still limitations for both metrics. On the one hand, although the MSE is a common metric to measure the overall distance between the real and reconstructed features, it lacks the similarity information of individual features. On the other hand, the attack accuracy might depend on feature correlations. For the extreme case where the passive-party features are independent of the label, the evaluation classifier cannot effectively capture the correlation between the feature values and the predicted labels. In this case, the attack accuracy will be close to random guessing regardless of the reconstructed values, which may cause a lower estimation of privacy risks. Based on the limitations of both metrics, a future task is to explore better metrics to evaluate the privacy risks of reconstruction attacks. Moreover, even though we have shown that the proposed attack can lead to increased privacy vulnerability compared with the baseline algorithms and random guesses, the results do not indicate a complete defeat of privacy in VFL frameworks. Privacy leakage in real-life applications should still be carefully evaluated based on the semantic meaning and sensitivity level of the features.

Furthermore, besides the basic two-party VFL setting, there are other variants of VFL frameworks, such as VFL without a trusted coordinator and multi-party VFL, as discussed in Section 2.1. These variants may highlight different aspects and levels of privacy leakage. For instance, for the multi-party VFL, the existence of multiple passive parties may increase the rate of passive-party features, but the attacker may also collude with some of the passive-parties to increase the privacy risk. Thus, one of the essential future works is to systematically compare the potential privacy leakage of different VFL frameworks and analyze the capability of privacy-enhancing techniques regarding these variants. In addition, another direction is to analyze the performance of the reconstruction attack on deep models. As shown in Section 5.2.2, an increase in model size and a different choice of activation function may impact attack performance. Moreover, the use of different types of layers, *e.g.*, convolution or recurrent layers, may also increase attack difficulty. As such, studying enhanced attack algorithms against these deep models under the VFL setting is also meaningful future work.

Finally, compared to the HFL setting, there are still only few studies conducting privacy and security analyses under the VFL setting. For instance, a malicious attacker can also apply the reconstruction attack during training [45] or stage a backdoor attack. In addition, during the prediction phase, the attacker may use the predicted confidence vectors to steal the passive-party model. As such, further explorations of security and privacy attacks under VFL should be considered fruitful research directions.

8 Conclusion

In this paper, we investigate the privacy leakage in two-party VFL frameworks during the prediction phase. We first present a generic attack framework. In comparison to previous work, our attack is model-agnostic and can be flexibly applied to different ML models. Furthermore, we conduct a comprehensive privacy risk analysis under both white-box and black-box settings. Extensive experiments on a number of public datasets demonstrate that the reconstruction attack during the prediction phase is a substantial privacy risk in VFL frameworks, which cannot be simply avoided by crypto-based confidentiality approaches. Additional information compression and randomization approaches should be applied to the confidence scores to provide strengthened privacy protection.

Acknowledgements

We thank the anonymous reviewers for their constructive comments for improving this paper. In particular,

we thank our shepherd, Soteris Demetriou, for his valuable suggestions during the revision process.

References

[1] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM, 2016.

[2] Nick Angelou, Ayoub Benaissa, Bogdan Cebere, William Clark, Adam James Hall, Michael A. Hoeh, Daniel Liu, Pavlos Papadopoulos, Robin Roehm, Robert Sandmann, Phillipp Schoppmann, and Tom Titcombe. Asymmetric private set intersection with applications to contact tracing and private vertical federated machine learning. *CoRR*, abs/2011.09350, 2020.

[3] Sean Augenstein, H. Brendan McMahan, Daniel Ramage, Swaroop Ramaswamy, Peter Kairouz, Mingqing Chen, Rajiv Mathews, and Blaise Agüera y Arcas. Generative models for effective ML on private, decentralized datasets. In *8th International Conference on Learning Representations*. OpenReview.net, 2020.

[4] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *19th International Conference on Computational Statistics*, pages 177–186, 2010.

[5] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium*, pages 267–284, 2019.

[6] Tianyi Chen, Xiao Jin, Yuejiao Sun, and Wotao Yin. VAFL: A method of vertical asynchronous federated learning. *CoRR*, abs/2007.06081, 2020.

[7] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, Dimitrios Papadopoulos, and Qiang Yang. Secureboost: A lossless federated learning framework. *IEEE Intelligent Systems*, 2021.

[8] Emiliano De Cristofaro and Gene Tsudik. Practical private set intersection protocols with linear complexity. In *14th International Conference on Financial Cryptography and Data Security*, volume 6052 of *Lecture Notes in Computer Science*, pages 143–159. Springer, 2010.

[9] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In Kwangjo Kim, editor, *Public Key Cryptography, 4th International Workshop on Practice and Theory in Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136. Springer, 2001.

[10] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. http://archive.ics.uci.edu/ml.

[11] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[12] Siwei Feng and Han Yu. Multi-participant multi-class vertical federated learning. *CoRR*, abs/2001.11154, 2020.

[13] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *International Conference on the Theory and Applications of Cryptographic Techniques*, volume 3027 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2004.

[14] Ananda L. Freire, Guilherme A. Barreto, Marcus Veloso, and Antonio T. Varela. Short-term memory mechanisms in neural network learning of robot navigation tasks: A case study. In *6th Latin American Robotics Symposium*, pages 1–6. IEEE, 2009.

[15] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients - How easy is it to break privacy in federated learning? In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*. Curran Associates Inc., 2020.

[16] Andrew Hard, Kanishka Rao, Rajiv Mathews, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *CoRR*, abs/1811.03604, 2018.

[17] Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *CoRR*, abs/1711.10677, 2017.

[18] Briland Hitaj, Giuseppe Ateniese, and Fernando Pérez-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*, pages 603–618. ACM, 2017.

[19] Yaochen Hu, Di Niu, Jianming Yang, and Shengping Zhou. FDML: A collaborative machine learning framework for distributed features. In Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis, editors, *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2232–2240, 2019.

[20] Yan Huang, David Evans, and Jonathan Katz. Private set intersection: Are garbled circuits better than custom protocols? In *19th Annual Network and Distributed System Security Symposium*. The Internet Society, 2012.

[21] Bargav Jayaraman and David Evans. Evaluating differentially private machine learning in practice. In *28th USENIX Security Symposium*, pages 1895–1912. USENIX Association, 2019.

[22] Xue Jiang, Xuebing Zhou, and Jens Grossklags. Privacy-preserving high-dimensional data collection with federated generative autoencoder. *Proceedings on Privacy Enhancing Technologies*, 2022(1):481–500, 2022.

[23] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings*, 2015.

[24] Wenqi Li, Fausto Milletarì, Daguang Xu, Nicola Rieke, Jonny Hancox, Wentao Zhu, Maximilian Baust, Yan Cheng, Sébastien Ourselin, M. Jorge Cardoso, and Andrew Feng. Privacy-preserving federated brain tumour segmentation. In *10th International Workshop on Machine Learning in Medical Imaging*, volume 11861 of *Lecture Notes in Computer Science*, pages 133–141. Springer, 2019.

[25] Yang Liu, Yingting Liu, Zhijie Liu, Yuxuan Liang, Chuishi Meng, Junbo Zhang, and Yu Zheng. Federated forest. *IEEE*

Transactions on Big Data, 2020.

[26] Linpeng Lu and Ning Ding. Multi-party private set intersection in vertical federated learning. In *19th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pages 707–714. IEEE, 2020.

[27] Songtao Lu, Yawen Zhang, and Yunlong Wang. Decentralized federated learning for electronic health records. In *54th Annual Conference on Information Sciences and Systems*, pages 1–5. IEEE, 2020.

[28] Xinjian Luo, Yuncheng Wu, Xiaokui Xiao, and Beng Chin Ooi. Feature inference attack on model predictions in vertical federated learning. In *37th IEEE International Conference on Data Engineering*, pages 181–192. IEEE, 2021.

[29] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282, 2017.

[30] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy*, pages 691–706. IEEE, 2019.

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

[32] Milad Nasr, Reza Shokri, and Amir 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*, pages 739–753. IEEE, 2019.

[33] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

[34] Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5):1333–1345, 2017.

[35] Swaroop Ramaswamy, Rajiv Mathews, Kanishka Rao, and Françoise Beaufays. Federated learning for emoji prediction in a mobile keyboard. *CoRR*, abs/1906.04329, 2019.

[36] Daniele Romanini, Adam James Hall, Pavlos Papadopoulos, Tom Titcombe, Abbas Ismail, Tudor Cebere, Robert Sandmann, Robin Roehm, and Michael A. Hoeh. PyVertical: A vertical federated learning framework for multi-headed splitNN. *CoRR*, abs/2104.00489, 2021.

[37] Stacey Truex, Ling Liu, Mehmet Emre Gursoy, Lei Yu, and Wenqi Wei. Demystifying membership inference attacks in machine learning as a service. *IEEE Transactions on Services Computing*, 2019.

[38] Jaideep Vaidya and Chris Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 639–644. ACM, 2002.

[39] Jaideep Vaidya and Chris Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 206–215. ACM, 2003.

[40] Jaideep Vaidya and Chris Clifton. Privacy preserving naive Bayes classifier for vertically partitioned data. In *Proceedings of the 2004 SIAM International Conference on Data Mining*, pages 522–526. SIAM, 2004.

[41] Jaideep Vaidya, Chris Clifton, Murat Kantarcioglu, and Scott Patterson. Privacy-preserving decision trees over vertically partitioned data. *ACM Transactions on Knowledge Discovery from Data*, 2(3):1–27, 2008.

[42] Chang Wang, Jian Liang, Mingkai Huang, Bing Bai, Kun Bai, and Hao Li. Hybrid differentially private federated learning on vertically partitioned data. *CoRR*, abs/2009.02763, 2020.

[43] Qian Wang, Minxin Du, Xiuying Chen, Yanjiao Chen, Pan Zhou, Xiaofeng Chen, and Xinyi Huang. Privacy-preserving collaborative model learning: The case of word vector training. *IEEE Transactions on Knowledge and Data Engineering*, 30(12):2381–2393, 2018.

[44] Yichuan Wang, Yuying Tian, Xinyue Yin, and Xinhong Hei. A trusted recommendation scheme for privacy protection based on federated learning. *CCF Transactions on Networking*, 3(3-4):218–228, 2020.

[45] Haiqin Weng, Juntao Zhang, Feng Xue, Tao Wei, Shouling Ji, and Zhiyuan Zong. Privacy leakage of real-world vertical federated learning. *CoRR*, abs/2011.09290, 2020.

[46] Yuncheng Wu, Shaofeng Cai, Xiaokui Xiao, Gang Chen, and Beng Chin Ooi. Privacy preserving vertical federated learning for tree-based models. *Proceedings of the VLDB Endowment*, 13(11):2090–2103, 2020.

[47] Bangzhou Xin, Wei Yang, Yangyang Geng, Sheng Chen, Shaowei Wang, and Liusheng Huang. Private fl-gan: Differential privacy synthetic data generation based on federated learning. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2927–2931. IEEE, 2020.

[48] Kai Yang, Tao Fan, Tianjian Chen, Yuanming Shi, and Qiang Yang. A quasi-Newton method based vertical federated learning framework for logistic regression. *CoRR*, abs/1912.00513, 2019.

[49] Liu Yang, Ben Tan, Vincent W. Zheng, Kai Chen, and Qiang Yang. Federated recommendation systems. In *Federated Learning - Privacy and Incentive*, volume 12500 of *Lecture Notes in Computer Science*, pages 225–239. Springer, 2020.

[50] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology*, 10(2):12:1–12:19, 2019.

[51] Shengwen Yang, Bing Ren, Xuhui Zhou, and Liping Liu. Parallel distributed logistic regression for vertical federated learning without third-party coordinator. *CoRR*, abs/1911.09824, 2019.

[52] Ziqi Yang, Bin Shao, Bohan Xuan, Ee-Chien Chang, and Fan Zhang. Defending model inversion and membership inference attacks via prediction purification. *CoRR*, abs/2005.03915, 2020.

[53] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Founda-*

tions of Computer Science, pages 160–164. IEEE Computer Society, 1982.

[54] Hwanjo Yu, Jaideep Vaidya, and Xiaoqian Jiang. Privacy-preserving SVM classification on vertically partitioned data. In *10th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, volume 3918 of *Lecture Notes in Computer Science*, pages 647–656. Springer, 2006.

[55] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. iDLG: Improved deep leakage from gradients. *CoRR*, abs/2001.02610, 2020.

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