Defending Batch-level Label Inference and Replacement Attacks in Vertical Federated Learning

Tianyuan Zou*, Yang Liu*, Yan Kang, Wenhan Liu, Yuanqin He, Zhihao Yi, Qiang Yang†, Ya-Qin Zhang†

Abstract—In a vertical federated learning (VFL) scenario where features and models are split into different parties, it has been shown that sample-level gradient information can be exploited to deduce crucial label information that should be kept secret. An immediate defense strategy is to protect sample-level messages communicated with Homomorphic Encryption (HE), exposing only batch-averaged local gradients to each party. In this paper, we show that even with HE-protected communication, private labels can still be reconstructed with high accuracy by gradient inversion attack, contrary to the common belief that batch-averaged information is safe to share under encryption. We then show that backdoor attack can also be conducted by directly replacing encrypted communicated messages without decryption. To tackle these attacks, we propose a novel defense method, Confusional AutoEncoder (termed **CAE**), which is based on autoencoder and entropy regularization to disguise true labels. To further defend attackers with sufficient prior label knowledge, we introduce DiscreteSGD-enhanced CAE (termed **DCAE**), and show that DCAE significantly boosts the main task accuracy than other known methods when defending various label inference attacks.

Index Terms—Vertical Federated Learning, Label Inference, Label Replacement, Confusional AutoEncoder, Privacy.

1 Introduction

Federated Learning (FL) [18], [27], [32] is a collaborative learning framework for training deep learning models with data privacy protection. In the original proposal of cross-device FL [27], data samples are distributed among different participants. Thus these works can be regarded as sample-partitioned FL, or horizontal FL (HFL) [32]. Along the data-silo dimension, feature-partitioned FL, or vertical FL (VFL) [5], [15], [17], [23], [25], [32] is another important scenario for many real-world applications. For example, when a bank and an E-commerce company collaboratively train a credit risk model, different features of the same group of users are partitioned among different parties. Similar to HFL, it is important to safeguard the VFL framework from attacks and data leakages.

Because FL is a collaboration system that requires parties to exchange gradient or model level information, it has been of great research interest to study the potential information leakage from gradients. In FL scenarios, curious parties might be honest but attempt to infer other parties' private data through inference attacks while malicious parties might manipulate the learning process for their own purposes through backdoor attacks. Previous works [34], [35], [36] have shown that it is possible to recover pixel-level (deep leakage) raw data from transmitted sample-level gradient information. Previous studies on label leakage of VFL [21] have made the assumption that per-sample gradient is communicated and revealed, therefore the value of gradients can be used to infer labels. However, sample-level information is not necessarily available in VFL with Homomorphic Encryption (HE) or Multi-Party Secure Computation (MPC) protection [32], where the encrypted intermediate results are communicated instead of plain-text messages resulting that both data and local model parameters are kept secret and no sample-level information is accessible to clients. Instead, batch-level local gradient is decrypted and accessible. Although this does not harm the convergence of VFL models, it does make the inference attacks more difficult to execute since they need to recover sample-level labels with only batch-level gradient information (See Figure 1).

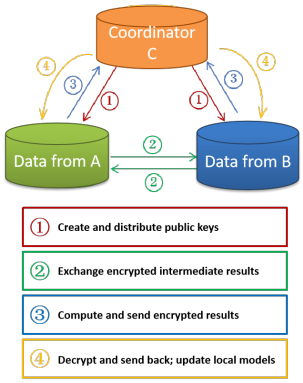


Fig. 1. Protocols of a VFL system with Homomorphic Encryption.

In this paper, we first perform theoretical analysis on batch-level label inference task under VFL scenario with HE protection to show that labels can be fully recovered when batch size is smaller than the dimension of the embedded features of the last fully connected layer. Next, we conduct experiments on multiple datasets to demonstrate that batch-level label inference attack can be successful using a gradient inversion technique under large batch size with only local batch-averaged gradients information, achieving a la-

- *The two authors contributed equally to this paper.*
- *Corresponding author.*

bel reconstruction accuracy over 90%. Built on the success of the label inference attack, we further show that attacker can directly replace labels through gradient-replacement attack without modifying the VFL protocols. Finally, we evaluate several defense strategies. We introduce a novel technique termed confusional autoencoder (**CAE**) based on autoencoder and entropy regularization, to hide the true labels without any noticeable changes to the original protocol. We demonstrate that our technique can successfully block those two attacks we proposed without hurting as much main task accuracy as existing methods such as differential privacy (DP) and gradient sparsification(GS), making it an attractive defense strategy in VFL scenarios. To further protect the CAE decoder from being learned by attackers with prior knowledge of labels, for example in model completion (MC) attack, we introduce DiscreteSGD-enhanced CAE (**DCAE**) which reaps the benefit of both CAE and DiscreteSGD and demonstrate its robustness to the original gradient inversion and backdoor attacks as well as superior performance in defending MC attacks compared with other defense methods including DP, GS and standalone DiscreteSGD. In summary, our contributions are the following:

- In the VFL setting, we introduce a novel gradient inversion-based label inference attack that utilize only the batch-level gradients information, achieving much higher attack accuracy than existing attacks. We further propose a new backdoor attack by replacing the encrypted gradients at the party without label, achieving high accuracy on both backdoor tasks and main tasks.
- We introduce a novel defense method CAE to effectively defend these two attacks by training an autoencoder with adjustable confusion to maximally confuse the attacker without hurting main task accuracy. Furthermore, we introduce DiscreteSGD-enhanced CAE (DCAE), to improve CAE's robustness when faced with attackers having sufficient prior label knowledge. We demonstrate with extensive experiments on MNIST, NUSWIDE, CIFAR10 and CIFAR100 dataset that DCAE is effective in various kinds of attacks in VFL setting and achieves better main task accuracy and attack accuracy balance than previous defense methods.

2 Related Work

VFL frameworks [32] support various local models including trees [5], linear and logistic regression [13], [16], [19], and neural networks [17], [23]. As each sample's features are distributed in multiple parties, sample-specific updates are communicated for gradient calculation during training. Communications of sample-level information can be either raw or encrypted, where privacy-preserving techniques such as Homomorphic Encryption (HE) [1], [28] is typically applied to preserve user-level privacy [32]. Recently, label leakage and protection for vertical federated learning (VFL) framework is studied [11], [21], [26]. Liu [26] points out that label can be inferred from sign of per-sample gradients. Li's study [21] exploits the difference in the norms of per-sample gradients for positive and negative classes. Fu [11] proposed three different label inference attacks in VFL setting: passive model completion, active model completion and direct label inference attack. In model completion attacks, the attacker needs auxiliary labeled data for fine-tuning its local model. We attack a more challenging system in our work, where the communicated messages are protected by HE with only the final batch-level local model gradients accessible to attackers, a scenario that is inline with real-world VFL projects, such as the implementation of FATE [24].

In addition, previous studies have shown that FL opens doors to backdoor attacks like model poisoning. Bagdasaryan [3] introduced a backdoor attack to FL by replacing the global model with a targeted poisoning model and discussed the effectiveness of possible defense strategies. Xie [31] introduced distributed backdoor attacks to FL. Sun [29] studied backdoor and defense strategies in FL and show that norm clipping and "weak" differential privacy mitigate the attacks. However, all the works above consider the backdoor attack under HFL scenario.

On the defense side, existing defense strategies mainly focus on adding noise, such as differential privacy [8], [9], [10] and MARVELL [21], or reducing the transmitted information [11], such as discretization [7], sparsification [2] and compression [22], while a few focuses on data augmentation [12], and mutual information regularization [30]. However, these defenses usually suffer from accuracy loss in main tasks. In our approach, we focus on maintaining main task accuracy without changing the existing protocols by introducing maximally confused fake labels. Our first basic defense method is orthogonal to the above defense methods and can be used jointly, forming our enhanced defense method, to improve the overall robustness and main task accuracy in defending different attacks.

3 A Vertical Federated Learning Framework with HE

In a typical VFL system [32] which is often applied in cross-organizational collaborative learning scenarios, K data owners collaboratively train a machine learning model based on a set of data $\{\mathbf{x}_i, y_i\}_{i=1}^N$ with only one party owns labels $\{y_i\}_{i=1}^N$. This is a reasonable assumption because in reality, labels (such as users' credit scores, patients' diagnosis etc.) are expensive to obtain and only exist in one or few organizations. Under this scenario, the feature vector $\mathbf{x}_i$ is often decomposed into K blocks $\{\mathbf{x}_i^k\}_{k=1}^K$ with each block belongs to one owner. Without loss of generality, we can assume that the labels are located in party K. Then the collaborative training problem can be formulated as:

$$o_i^K = f(\theta_1, \ldots, \theta_K; \mathbf{x}_i^1, \ldots, \mathbf{x}_i^K) \tag{1}$$

$$\min_{\Theta} \mathcal{L}(\Theta; \mathcal{D}) \triangleq \frac{1}{N} \sum_{i=1}^{N} \ell(o_i^K, y_i^K) + \beta \sum_{k=1}^{K} \gamma(\theta_k) \tag{2}$$

where θ_k denotes the training parameters of the k^{th} party; $\Theta = [\theta_1; \ldots; \theta_K]$; N denotes the total number of training samples; $f(\cdot)$ and $\gamma(\cdot)$ denote the prediction function and regularizer with β being the hyperparameter. Following previous work, we assume each party adopts a sub-model G_k which generates local predictions, also local latent representations, with $H_i^k = G_k(\theta_k, \mathbf{x}_i^k)$. The final prediction is

Algorithm 1 A VFL framework with Homomorphic Encryption (HE)

Input: learning rate η
Output: Model parameters $\theta_1, \theta_2 \ldots \theta_K$

1: Party 1,2,. . . ,K, initialize $\theta_1, \theta_2, \ldots \theta_K$.
2: TTP creates encryption pairs, send public key to each party;
3: **for** each iteration j=1,2, ... **do**
4: Randomly sample $S \subset [N]$
5: **for** each passive party ($k \neq K$) in parallel **do**
6: $\boldsymbol{k}$ computes, encrypts and sends $\{[[H_i^k]]\}_{i \in S}$ to party K;
7: **end for**
8: active party K computes and sends $\{[[\frac{\partial \ell}{\partial H_i}]]\}_{i \in S}$ to all other parties;
9: **for** each party k=1,2,. . . ,K in parallel **do**
10: $\boldsymbol{k}$ computes $[[\nabla_k \ell]]$ with Eq.(4), sends them with random mask to TTP for decryption;
11: $\boldsymbol{k}$ receive and unmask $\nabla_k \ell$ and update $\theta_k^{j+1} = \theta_k^j - \eta \nabla_k \ell$;
12: **end for**
13: **end for**

made by merging H_i^k with a nonlinear operation $S(\cdot)$, such as softmax function. That is,

$$\ell(\theta_1, \ldots, \theta_K; \mathcal{D}_i) = \ell(S(\sum_{k=1}^{K} H_i^k), y_i^K) \tag{3}$$

where G_k can adopt a wide range of models such as linear or logistic regression, support vector machines, neural networks etc. Let $H_i = \sum_{k=1}^{K} H_i^k$, then the gradient function has the form:

$$\nabla_k \ell(\theta_1, \ldots, \theta_K; \mathcal{D}_i) = \frac{\partial \ell}{\partial H_i} \frac{\partial H_i^k}{\partial \theta_k} \tag{4}$$

We refer the party having the labels as **active party**, and the rest as **passive parties**. In a basic VFL protocol where no encryption is applied, each passive party sends raw $\{H_i^k\}$ to active party. Then the active party calculates $\{\frac{\partial \ell}{\partial H_i}\}$ and sends the raw result back to passive parties for gradient update. To further protect sample-level information leakage from intermediate results, Homomorphic Encryption(HE), denoted as $[[\cdot]]$, is applied to each sample's communicated updates, $\{H_i^k\}$, and the gradient calculations are performed under encryption. Note that although the coordinator is trusted for decryption, VFL participants do not trust it for anything other than that, such as having access to parties' local data. Specifically, as Figure 1 illustrated, the following steps are taken: 1) Coordinator creates encryption pairs, sends public key to each party; 2) Parties encrypt and communicate the intermediate results for gradient calculations; 3) Parties compute encrypted gradients and add additional mask respectively and then send encrypted values to coordinator; 4) Coordinator decrypts and sends the decrypted gradients back to each party; 5) Parties unmask the gradients, update the model parameters accordingly. Note the additional mask is applied to ensure coordinator server does not learn gradients from parties, see [32]. This process ensures that only local batch-averaged gradients rather than sample-level gradients are available to passive party. See also Algorithm 1 and Figure 2(A).

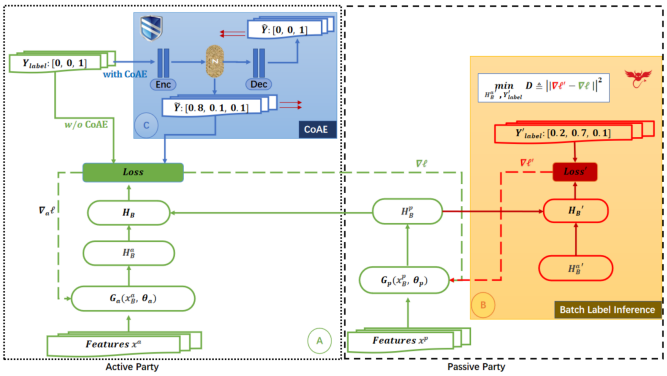


Fig. 2. The gradient-inversion-based batch-level label inference attack and defense in VFL. (A) Basic VFL framework. (B) Gradient-inversion-based batch-level label inference attack. (C) CAE protection. All the symbols with B as subscripts in this figure denote the batch level arguments. a, p stands for **active** and **passive** party respectively.

Attack	Require sample-level gradients	Require auxiliary labeled samples	Number of classes supported
Norm-based Scoring [21]	✓	×	$= 2$
Direction-based Scoring [21]	✓	×	$= 2$
Passive Model Completion [11]	×	✓	≥ 2
Active Model Completion [11]	×	✓	≥ 2
Direct Label Inference [11]	✓	×	≥ 2
Batch-level Label Inference (this work)	×	×	≥ 2

TABLE 1
Comparison of different label inference attacks in VFL setting.

4 Batch-level Label Inference Attack

4.1 Threat Model

Based on Algorithm 1, we consider the following attack model: (i) The attacker has access to the training data of one passive party and controls the training procedure and the local model of that passive party; (ii) The attacker only receives batched-averaged local gradients in plain text and encrypted per-sample local gradients; (iii) The attacker does not modify local updates such as training weights and gradients in the VFL protocol; (iv) The attacker does not control any benign party's training nor does the attacker controls the global communication and aggregation protocols. The differences between our model and the VFL framework considered in [11], [21] is that our threat model further assumes that the communicated messages between parties are encrypted, so that per-sample gradients are not available for inference. Therefore our attack is on the **batch** or **population** level. In addition, we do not assume attacker has balanced auxiliary labeled samples, which can be difficult or impossible to obtain in real-world applications. We compare our

Algorithm 2 Batch-level label inference attack in VFL

Input: learning rate η; Batch $\mathcal{B}$
Output: the label recovered from the gradients $\{y_i'\}_{i\in\mathcal{B}}$
Passive party do:

receive decrypted local gradients $\nabla\ell$;
for i in $\mathcal{B}$ **do**
  Initialize $H_i^{a\prime} \leftarrow \mathcal{N}(0,1), y_i{}' \leftarrow \mathcal{N}(0,1)$
end for
for $j \leftarrow 1$ **to** $iterations$ **do**
  compute local recovery loss with Eq.(10);
  for i in $\mathcal{B}$ **do**
    $y_i{}' \leftarrow y_i{}' - \eta \cdot \partial D/\partial y_i{}', H_i^{a\prime} \leftarrow H_i^{a\prime} - \eta \cdot \partial D/\partial H_i^{a\prime}$
  end for
end for
return $\{y_i'\}_{i\in\mathcal{B}}$

approach with other recent works in Table 1. In this table, norm-based and direction-based scoring attacks [21] can be used for binary classifier task only, model completion is the method based on adding and fine-tuning a "completion layer" at the passive party with balanced auxiliary data to infer active party's private label and direct label inference is a sample-level label inference attack that utilize the sign of gradients. All these methods are designed for label inference in VFL setting.

4.2 Methodology

In this section, we first perform analysis on the computed batch-averaged local gradients and its connections with the labels and then show how the success of label inference attack depends on batch size. Next we introduce our attack method, which is based on gradient inversion.

Lemma 1. Direct Label Inference. *In Eq.(3), if $\mathcal{S}(\cdot)$ represents a softmax function and ℓ represents a cross-entropy loss, then inferring $\frac{\partial \ell}{\partial H_i}$ is equivalent to inferring labels.*

Proof. $\frac{\partial \ell}{\partial H_i}$ is a N-dimension vector where N is the number of classes with the j^{th} element being:

$$\frac{\partial \ell}{\partial H_{i,j}} = \begin{cases} S_j, & j \neq y \\ S_j - 1, & j = y \end{cases} \tag{5}$$

where S_j is the softmax function $S_j = \frac{e^{h_j}}{\sum_v e^{h_v}}$ over H_i. Here we abuse the notation y to denote the index of the true label. If $\frac{\partial \ell}{\partial H_i}$ is revealed and known by attacker, the label information is known because the y^{th} element of $\frac{\partial \ell}{\partial H_i}$ will be only element having opposite sign compared to others.

As Eq.(4) shows, for party k in VFL system, H_i^k is the logits of the local model of party k for sample i and is a C-dimention vector (C is the number of classes for the classification task). Let B denotes the size of a batch $\mathcal{B}$. Then the total loss is $\ell_{\mathcal{B}} = \frac{1}{B}\sum_{i=1}^{B} \ell_i$. Denote the parameters of last fully connected layer o on client k by $\theta_{k,o}$, which is a $C \times M$ matrix, where M is the dimension of the feature embedding. Then the local gradients at the last layer are:

$$\nabla_{\theta_{k,o}} \ell_{\mathcal{B}} = \frac{\partial \ell_{\mathcal{B}}}{\partial H_{\mathcal{B}}} \frac{\partial H_{\mathcal{B}}^k}{\partial \theta_{k,o}} \tag{6}$$

where $\nabla_{\theta_{k,o}} \ell_{\mathcal{B}} \in \mathbb{R}^{C\times M}$, $\frac{\partial \ell_{\mathcal{B}}}{\partial H_B} \in \mathbb{R}^{C\times B}$ and $\frac{\partial H_{\mathcal{B}}^k}{\partial \theta_{k,o}} \in \mathbb{R}^{B\times M}$.

Theorem 1. Denote $\nabla_{\theta_{k,o}} \ell_{\mathcal{B}} = \frac{\partial \ell_{\mathcal{B}}}{\partial H_{\mathcal{B}}} \frac{\partial H_{\mathcal{B}}^k}{\partial \theta_{k,o}}$ by $Q = UA$. Then, when $B \leq rank(A) \leq M$, labels can be 100% fully recovered.

Proof. Because $A = \frac{\partial H_{\mathcal{B}}^k}{\partial \theta_{k,o}}$ is the matrix of local computed back-propagation gradients at party k and $Q = \nabla_{\theta_{k,o}} \ell_{\mathcal{B}}$ is the averaged gradients transformed to party k thus is available to party k, so the unknowns are $U = \frac{\partial \ell_{\mathcal{B}}}{\partial H_B}$. Therefore the problem is reduced to solving the linear equations described in Eq.(6) with $C \times B$ unknown variables and $C \times M$ equations in the linear equations. As $Q^T = A^T U^T$, the matrix form of it is:

$$\begin{bmatrix} q_{11} & \dots & q_{C1} \\ \vdots & \ddots & \vdots \\ q_{1M} & \dots & q_{CM} \end{bmatrix} = \begin{bmatrix} a_{11} & \dots & a_{B1} \\ \vdots & \ddots & \vdots \\ a_{1M} & \dots & a_{BM} \end{bmatrix} \begin{bmatrix} u_{11} & \dots & u_{C1} \\ \vdots & \ddots & \vdots \\ u_{1B} & \dots & u_{CB} \end{bmatrix} \tag{7}$$

where $\{u_{cb}\}_{c=1,2,..,C,b=1,2,..,B}$ are unknown arguments.

Eq.(7) can be separated into C parts that are foreign from one another:

$$\begin{bmatrix} a_{11} & \dots & a_{B1} \\ \vdots & \ddots & \vdots \\ a_{1M} & \dots & a_{BM} \end{bmatrix} \begin{bmatrix} u_{c1} \\ \vdots \\ u_{cB} \end{bmatrix} = \begin{bmatrix} q_{c1} \\ \vdots \\ q_{cM} \end{bmatrix}, \ c = 1, 2, ..., C \tag{8}$$

In Eq.(8), there are M equations and B unknown variables. Therefore, when $B \leq rank(A) \leq M$, all the C groups of Eq.(8) has only one group of solution. According to Lemma 1, the recovery rate of label should be 1.0. That is, when the batch size is no larger than the size of embedded features, labels are leaked with 100%.

When $B > rank(A)$, there are more than one group of solution. However, additional constraints can be exploited. First, each column in $\frac{\partial \ell_{\mathcal{B}}}{\partial H_B}$ represents a sample i's gradient with respect to H_i, and it should have only one element that is less than zero with all the other elements should be in the range of $[0, 1]$. Also, from Eq.(5), the sum of the elements in each column should be zero. What's more, there is no need to find the exact value of each element of the unknown matrix. Instead, the attacker only needs to find out which element is negative. In this case, multiple solutions may lead to the same conclusion about the labels.

Inspired by the attack performed by Zhu [36], we adopt a deep learning approach to accomplish this batch-level label inference attack, see Figure 2(A, B) and Algorithm 2. Specifically, the passive party (attacker) p sets up an internal model which tries to guess the labels and communicated intermediate results $\{H_i^a\}_{i\in\mathcal{B}}$ for every sample i in a batch $\mathcal{B}$ so that the simulated local gradients match the observed ones. First, for each sample i in each batch $\mathcal{B}$, it randomly initializes y_i' and $H_i^{a\prime}$, then computes its gradients using:

$$\nabla\ell' = \sum_{i\in\mathcal{B}} \partial \ell \left(\text{softmax}(H_i^p + H_i^{a\prime}), y_i{}'\right) / \partial \theta_p \tag{9}$$

Then the final objective is to match the simulated gradients with the observed gradients:

$$\min_{H_a', y'} \mathcal{D} \triangleq \|\nabla\ell' - \nabla\ell\|^2 \tag{10}$$

We name our attack gradient-inversion-based **batch-level label inference attack** and implement the optimization with gradient descend in Algorithm 2, in which y_i' and $H_i^{a\prime}$ are updated in order to match $\nabla\ell'$ with $\nabla\ell$. We show experimentally that this reconstruction can be quite

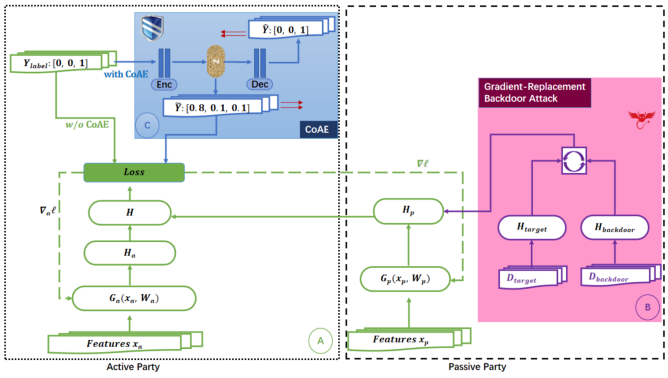


Fig. 3. The label replacement backdoor attack and defense in VFL. (A) Basic VFL framework. (B) Label replacement backdoor attack. (C) CAE protection.

powerful achieving a label reconstruction rate over 90% for large batch size. So, we conclude that batch averaging is not sufficient to protect the label information in HE-protected VFL setting. In Section6, we propose a novel defense technique to protect labels.

5 Label Replacement Backdoor Attacks

In previous sections, we focus on label inference as the adversarial objective. In this section, we further consider how the adversary is going to leverage the label information for its own benefit in the VFL framework. Specifically, we consider a backdoor attack where an adversary replaces a small number of labels with the target labels, which is commonly studied in HFL scenarios [3], [4], [31].

In our setting, the backdoor task is to assign an attacker-chosen label to input data with a specific pattern (i.e. a trigger), as shown in Figure 4. Also, the attacker aims to train a model which achieves high performance on both the original task and the backdoor task. The adversarial objective can be written as:

$$\min_{\Theta} \mathcal{L}^b(\Theta;\mathcal{D}) \triangleq \frac{1}{N_{cln}} \sum_{i\in\mathcal{D}_{cln}} \ell(o_i^K, y_i^K) + \frac{1}{N_{poi}} \sum_{i\in\mathcal{D}_{poi}} \ell(o_i^K, \tau^K) \tag{11}$$

where o_i^K is defined in Eq.(1), $\mathcal{D}_{cln}$,N_{cln},$\mathcal{D}_{poi}$,N_{poi} denote the clean dataset, number of samples in clean dataset, poisoned dataset and number of samples in poisoned dataset, respectively. τ^K denotes the target label. From algorithm 1, the only information the passive parties obtain is the batch-averaged gradients. Therefore the main challenge is for the passive parties to assign targeted labels to specific input data with trigger features **without direct access to labels**.

This backdoor task is built on the label inference attack discussed previously since once a passive party learns the corresponding true labels of the each sample, it can carefully design the encrypted communicated messages to replace the true label with a target label. Following Eq.(5), after label inference, $\frac{\partial \ell}{\partial H_i}$ is obtained and the y^{th} element of $\frac{\partial \ell}{\partial H_i}$ has

Fig. 4. Backdoor task settings.

Algorithm 3 Label replacement backdoor attack in VFL

Input:
D_{target}: clean target dataset;
$D_{backdoor}$: poisoned dataset;
$\mathcal{X}$: current batch of training dataset;
γ: amplify rate

Forward Propagation:
Passive party computes $\{H_i\}_{i\in\mathcal{X}}$;
empty pair set $\mathcal{P}$;
for x_i in $\mathcal{X}$ **do**
  if x_i belongs to $\mathcal{D}_{backdoor}$ **then**
    random select j from D_{target};
    $\mathcal{P} += < i, j >$;
    replace H_j with H_i;
  end if
end for
Passive party encrypts and sends $[[H_{\mathcal{X}}]]$ to active party;
Backward Propagation:
for $< i, j >$ in $\mathcal{P}$ **do**
  replace $[[\frac{\partial \ell}{\partial H_i}]]$ with $\gamma[[\frac{\partial \ell}{\partial H_j}]]$;
end for

opposite sign as compared to others. To inject the backdoor attack, the passive party only needs to replace $\frac{\partial \ell}{\partial H_i}$ with

$$\left(\frac{\partial \ell}{\partial H_{i,j}}\right)^b = \begin{cases} S_j, & j \neq \tau \\ S_j - 1, & j = \tau \end{cases} \tag{12}$$

where τ is the target label. Note that under HE-protected VFL scenario, this injection should be and can be performed under HE without knowing the plain-text value of S_j due to the additive property of HE, that is:

$$\left[\left[\left(\frac{\partial \ell}{\partial H_{i,j}}\right)^b\right]\right] = \begin{cases} [[S_j]], & j \neq \tau, j \neq y \\ [[S_j]] - [[1]], & j = \tau \\ [[S_j]] + [[1]], & j = y \end{cases} \tag{13}$$

Therefore we can exploit the recovered labels using label inference attack to perform backdoor attacks.

Such a backdoor attack requires the label inference step as a prerequisite, which requires running and observing the VFL process separately first. In fact, for this backdoor attack, we can also eliminate the label inference step completely and directly replace labels via replacing encrypted gradients. We assume that the passive attacker knows a few clean samples with the targeted label, marked as $\mathcal{D}_{target}$. Note that, these clean samples are from the original local dataset of the attacker and as few as 1 clean sample from target class is needed. With this assumption, we propose to inject a label replacement backdoor to the learning process directly by replacing the gradient as follows:

Algorithm 4 Training CAE for Disguising Labels

Input:
learning rate η; number of classes $\boldsymbol{c}$; batch size $\boldsymbol{N}$; maximum number of epochs $\boldsymbol{E}$;
Output: trained W_e, W_d

for $i \leftarrow 1$ to E **do**
  Randomly generate one-hot labels: $y \in R^{\mathrm{N}\times \mathrm{c}}$
  Generate fake and reconstructed labels by Eq.(14);
  Compute total loss L by Eq.(15)
  Update parameters:
  $W_e \leftarrow W_e - \eta \cdot \partial L/\partial W_e$
  $W_d \leftarrow W_d - \eta \cdot \partial L/\partial W_d$
end for
return W_e and W_d

Algorithm 5 VFL with CAE or DCAE protection

Input:
$\boldsymbol{X}, \boldsymbol{Y_{label}}$: input data and label of $\boldsymbol{X}$;
W_e, W_d: trained encoder and decoder parameters;
$\mathcal{S}(\cdot)$: differentiable model;
Procedure:

Get fake soft label: $\tilde{Y} = Enc(Y_{label}, W_e)$
Get the predicted soft label from VFL: $Y_p = \mathcal{S}(H)$
Compute cross entropy loss: $\ell^{CE} = CE(\tilde{Y}, Y_p)$
if DCAE is applied **then**
  quantify $\nabla\ell^{CE}$=$Discrete(\nabla\ell^{CE})$ according to Eq.(16)
end if
send $\nabla\ell^{CE}$ to passive party;

Inference Procedure:

Reconstruct true labels: $Y_t = \mathrm{argmax}(Dec\,(\mathcal{S}(H))$

- In the forward propagation, the attacker performs local computations of H_i^k, but for each poisoned sample i, it randomly selects one sample j from $\mathcal{D}_{target}$, replaces encrypted intermediate results $[[H_j^k]]_{j\in\mathcal{D}_{target}}$ with $[[H_i^k]]_{i\in\mathcal{D}_{backdoor}}$ and records the pair $< i, j >$;
- In the backward propagation, for each pair $< i, j >$, the attacker replaces the encrypted intermediate gradients of sample i it receives with encrypted gradients that exits at the position for sample j, update the model parameters using $\gamma[[\frac{\partial \ell}{\partial H_j}]]\frac{\partial H_i^k}{\partial \theta_k}$ according to Eq.(4) where $\frac{\partial \ell}{\partial H_i}$ is replaced by $\frac{\partial \ell}{\partial H_j}$.

Here γ is an amplify rate that we adjust to control the level of backdoor. This can also be understood as an identity stealth strategy where the backdoor sample steals a target's identity. By using such a strategy, the passive party will obtain the corresponding gradients with respect to the targeted label instead of its own therefore its local backdoor updates will be successful. To further prevent the active party from learning a reasonable mapping of the labels and intermediate results H_i of the poisoned samples, the passive party can instead output a random-valued vector to the active party for each poisoned sample during training. See Algorithm 3 and Figure 3(A, B) for full details.

6 Defense

We have shown that label leakage and backdoor attack are challenging problems to VFL in previous sections. In this section, we discuss possible routes for protection. The common defense mechanisms previously adopted include noisy gradients [8], [21], gradient sparsification(GS) [2] and discrete gradients (DG) [11]. However these techniques would suffer from significant main task accuracy loss when defending label leakage attacks because label accuracy tightly couples with main task accuracy.

To further decouple label inference from main task and boost main task accuracy without increasing label leakage, we propose a simple yet effective label disguise technique where the active party transforms the original labels to a set of "soft fake labels" so that the fake labels introduce as much confusion as possible to prevent the passive party from inferring the true labels but yet can still be used for main task prediction. To increase confusion, we propose learning a confusional autoencoder (CAE) to establish a mapping such that one label will be transformed into a soft label with higher probability for each alternative class. For example, a dog is mapped to [0.49,0.51] probability of dog and cat respectively. Whereas autoencoder is simple and effective for hiding true labels, confusion is important in changing the probability distribution of classes, making label leakage attack harder to succeed.

Algorithm 4 describes the training procedure of a CAE encoder and decoder pair. The encoder takes ground-truth labels y as inputs and outputs fake labels $\tilde{y}$, while the decoder takes fake labels $\tilde{y}$ as inputs and restore the original true labels. That is:

$$\begin{aligned} \tilde{y} &= Enc(y; W_e) \\ \hat{y} &= Dec(\tilde{y}, W_d) \end{aligned} \tag{14}$$

where W_e and W_d are the parameters for encoder and decoder respectively. To satisfy our conditions, we introduce a contrastive loss and an entropy loss:

$$L = CE(y, \hat{y}) - \lambda_1 CE(y, \tilde{y}) - \lambda_2 Entropy(\tilde{y}) \tag{15}$$

where $CE(\cdot)$ is cross-entropy loss. The two cross-entropy losses in Eq.(15) are for reconstructing the original label and differentiating the original and the soft fake labels respectively while $Entropy(\tilde{y})$ aims to add confusion since high entropy means low confidence on the prediction. λ_1,λ_2 are weight parameters. After training the encoder and decoder pair, the active party can leverage CAE to produce fake labels and use these fakes labels to compute gradients in VFL, thereby preventing label leakage attack (see Figure 2(C) for details). When performing inference, the active party transforms the predicted labels back using the trained decoder. Notice that CAE itself is effective for preventing direct exploitation of the relationship between label information and local gradients, therefore can be installed directly to defend gradient-inversion-based label inference attacks and label replacement backdoor attack. As will be shown in the following experiment section, it achieves the best defense results for these attacks.

Moreover, when integrated with information-reduction protections on VFL training process to further defend attacks that exploit prior label information to directly deduce

a mapping between local data and labels, CAE plays an important role in helping to alleviate the catastrophic drop on main tasks caused by information-reduction on gradients. The contributions of CAE are two folds. First, the decoder part of CAE is now the actual "last layer" of the VFL model resulting in a much accurate main task prediction. Secondly, with more confusion added to CAE (i.e. the increase of λ_2 value), the training of the main task becomes more difficult for the attacker (passive party) with only its local data and the confused gradients, resulting in a local model less capable of label predictions, thereby a low attack accuracy. To demonstrate its effectiveness, we introduced DiscreteSGD-enhanced CAE (DCAE) which quantifies the gradients using DiscreteSGD methods [11] under CAE protection. DiscreteSGD (DG) is a defense technique that simply quantifies the transmitted gradients by rounding each element to the nearest endpoint of the discrete bins as:

$$Discrete(g) \triangleq \sum_{w=0}^{W} V_w(g) \cdot e_w \tag{16}$$

where g denotes each element of the original gradient, $\{e_w\}_{w=0}^{W}$ is the set of endpoints of the W equal-interval discrete bins with $[e_0, e_W] = [\mu - 2\sigma, \mu + 2\sigma]$ where μ, σ are the mean and standard deviation of the original gradient. $\{V_w(\cdot)\}_{w=0}^{W}$ is a set of indicator functions (characteristic functions) that each satisfies:

$$V_w(g) = \begin{cases} 1, & g \in \left(\frac{e_{w-1}+e_w}{2}, \frac{e_w+e_{w+1}}{2}\right] \\ 0, & else \end{cases} \tag{17}$$

with $e_{-1} \triangleq -\infty, e_{W+1} \triangleq +\infty$.

The complete algorithm is shown in Algorithm 5. When applying such defenses, the active party produces fake soft labels, calculates gradients $\nabla \ell^{CE}$ according to these labels, generates $Discrete(\nabla \ell^{CE})$ if DCAE is applied and transmits the final result to other passive parties. As will be shown in the following experiments, DCAE defense is robust to various label inference attacks with better main task and attack accuracy trade-off.

In summary, our proposed CAE and its variant DCAE and achieve better main task and attack accuracy balance for various label inference attacks as well as label replacement backdoor attack in VFL. To the best of our knowledge, these are the first defense strategies put forward targeting specifically at label protection in VFL.

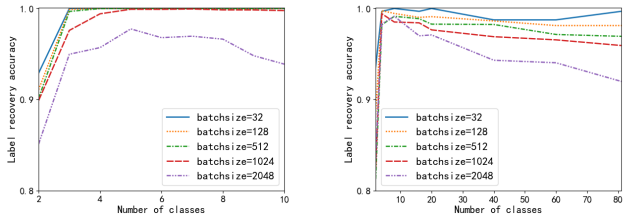
7 Experiments

We evaluate the effectiveness of our proposed attacks and defense strategies in a HE-protected VFL training protocol.

7.1 Dataset and Models

MNIST dataset In MNIST dataset [33], each image is evenly split into two halves and assigned to each party respectively. All 10 labels are used through out the experiment. A 2-layer MLP model with 32 neurons in the middle layer is used.

NUS-WIDE dataset In NUSWIDE dataset [6], each data sample is partitioned into an active party with all 634 image features, and a passive party with all 1000 text features. Five labels ('buildings', 'grass', 'animal', 'water', 'person') are chosen to form the training and testing set since they comparatively contain more data samples while all 81 labels are used for studying the impact of number of classes in label inference attack. A 2-layer MLP model with 32 neurons in the middle layer is used.



(a) MNIST (b) NUSWIDE

Fig. 5. Gradient-inversion-based batch-level label inference accuracy as a function of the number of classes for various batch size.

CIFAR datasets Both CIFAR10 [20] and CIFAR100 [20] are used in our experiments. For these datasets, images are evenly split and assigned to each party respectively. 20 classes of CIFAR100 is used for label inference and replacement attack while all labels of both datasets are used when comparing with MC attack. Resnet18 is used as the backbone model while Resnet20 is used when comparing with model completion (MC) attack to keep consistency with the original previous work.

7.2 Attacks at Passive Party

7.2.1 Batch-level Label Inference Attack

To test the effectiveness of the label inference attack, we use recovery accuracy, the ratio of correctly inferred label by passive party, as our metric. The samples and the classes are randomly selected for each experiment and each run is repeated 30 times. The results are shown in Figure 5. It can be seen that the batch-level label inference attack can be quite successfully for various batch size and number of classes, despite for the fact that the only clue is the batch-averaged gradients. As the batch size increases, the recovery rate gradually drops, indicating the inference task is more difficult. As the number of classes increases, the recovery rate increases at first and stays at a high level, likely due to the fact that the diversity of the labels reduces the possible combinations in the search space. For very large batch size, we observe a slight drop in recovery rate when the number of classes continues to grow, likely due to increasing difficulty for solving the optimization problem with increased number of variables by gradient inversion .

We also compare our batch-level label reconstruction attack with the label attacks proposed by Fu [11] in Table 2, namely the passive and active model completion attacks. Note that the direct label inference attacks proposed by Fu [11] and Li [21] are 100% successfully since it is based on sign of sample-level gradient information, which are considered not accessible in our scenario. We strictly follow the settings Fu [11] used in their experiment which means the last layer for aggregating the logits from both parties are slightly different in model completion attacks and direct label inference attack. Table 2 shows the results on CIFAR10 and CIFAR100 datasets. It's clear that our batch-level label

Attack	Dataset	Number of classes	Parameter	Attack ACC
Passive Model Completion [11]	CIFAR10	10	0.2	0.186
			0.4	0.241
			0.8	0.481
			1	0.618
			4	0.698
			12	0.715
	CIFAR100	100	1	0.085 (0.238)
			4	0.181 (0.409)
			12	0.252 (0.487)
Active Model Completion [11]	CIFAR10	10	0.2	0.193
			0.4	0.275
			0.8	0.531
			1	0.658
			4	0.741
			12	0.748
	CIFAR100	100	1	0.125 (0.310)
			4	0.261 (0.531)
			12	0.335 (0.594)
Batch-level Label Inference (this work)	CIFAR10	10	128	0.977
			512	0.934
			2048	0.893
	CIFAR100	100	128	0.999
			512	0.969
			2048	0.922

TABLE 2
Comparison of label inference attacks in VFL setting. The numbers in parentheses are top-5 accuracy, others are top-1 accuracy. For Passive and Active Model Completion Attack, the parameters are the quantity of auxiliary labeled data for each class (if the parameter value $p < 1$, then $p \times N$ auxiliary data samples are randomly distributed to $p \times N$ different classes, where N denotes the number of classes in total) whereas for Batch-level Label Inference attack, the parameters denote the batch size.

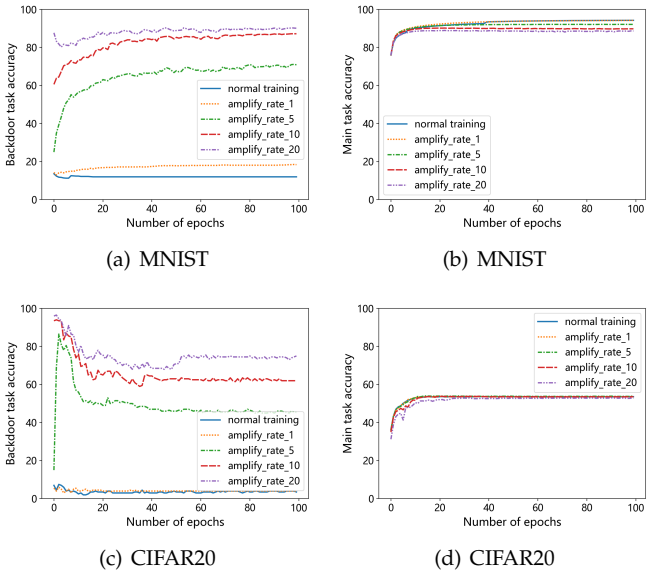
inference attack beats both the passive and the active model completion attack with much higher attack accuracy. Notice that both the model completion attacks relies heavily on the amount of auxiliary labeled data.

7.2.2 Label Replacement Backdoor Attack

Trigger injection Following [14], we randomly select 600 out of 60000 (1%) training samples and 100 out of 10000 (1%) testing samples and inject trigger 255 at pixel positions [25,27], [27,25], [26,26] and [27,27] for MNIST dataset. Also, following [14], for CIFAR datasets, pixels [29,31] and [30,30] are set to [0,255,0] along the channels and pixels [31,29] and [31,31] are set to [255,0,255] (see Figure 4). 100 out of 10000 (1%) training data and 20 out of 2000 (1%) test data are randomly marked with this trigger. For NUSWIDE dataset, the backdoor trigger is the last text feature equals 1 making less that 1% data samples that satisfy this feature in both training and testing set.

For MNIST and CIFAR datasets, the target label is randomly chosen while for NUSWIDE dataset, the target label is set to $class_0$. 10 target samples are randomly chosen from the target class for all datasets.

To evaluate the effectiveness of label replacement backdoor attack, we use backdoor accuracy as our metric. If the backdoor sample is classified by the VFL system as the target class instead of its original class, the backdoor task is considered successful for that sample. The backdoor accuracy is then the ratio of successfully mislabeled backdoor samples. The accuracy of the main classification class is also



(a) MNIST (b) MNIST

(c) CIFAR20 (d) CIFAR20

Fig. 6. Label replacement backdoor attacks on different datasets.

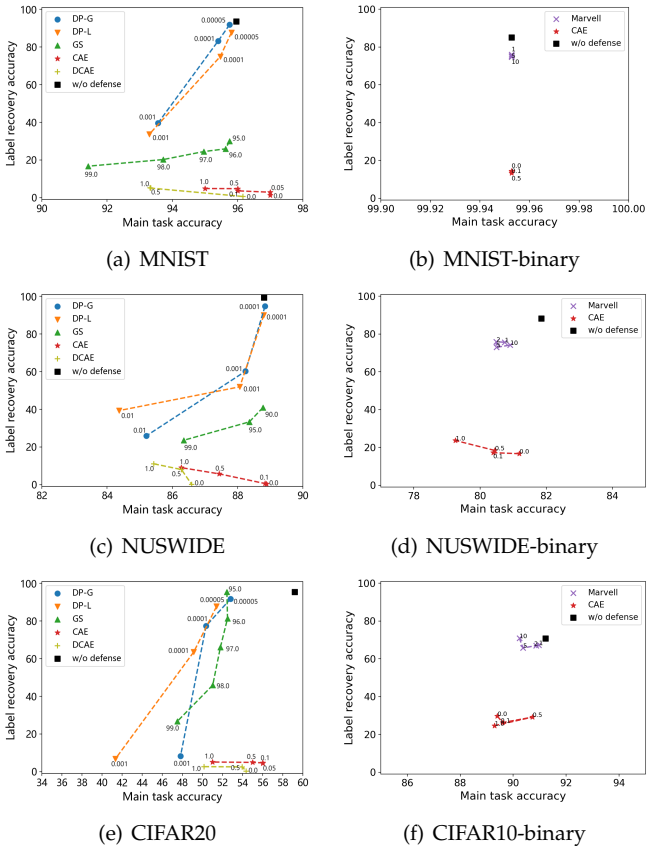
taken into consideration. The results are shown in Figure 6. The left column of Figure 6 shows the backdoor task accuracy at various levels of backdoor, adjusted by changing the amplify rate γ mentioned in Algorithm3. Some of the backdoor accuracy are more than 90%, meaning that the backdoor task successes. The right column of Figure 6 shows the main task accuracy, which stays high for various attack levels. When γ is too large, the backdoor task successes while the main task fails. In the following experiments, we set γ to 10, at which the main task and the backdoor task both work well.

7.3 Defenses at Active Party

7.3.1 Defense Settings

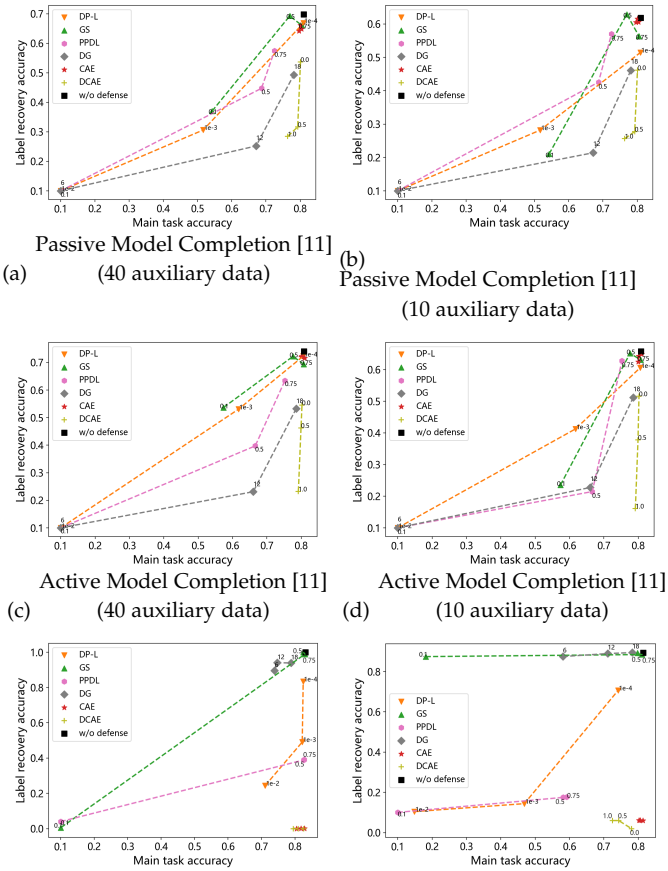
We evaluate various defense mechanism on the previously proposed batch-level label inference attack and label replacement backdoor attack: Differential Privacy (DP), Gradient Sparsification (GS), MARVELL [21], Privacy-Preserving Deep Learning (PPDL) [11], DiscreteSGD (DG) [11] as well as our method CAE and DCAE. Both encoder and decoder of the CAE and DCAE have the architecture: **fc(**$(6C+2)^2$**)-relu-fc(**C**)-softmax**, where **fc** denotes fully-connected layer and C denotes the class number (see Figure 3(C) for detail). We evaluate the performance of CAE and DCAE with various λ_2 from 0.0 to 1.0. We fix the discrete bins to 12 and only modifies λ_2 for DCAE in our experiments.

For DP, Gaussian differential private mechanism (DP-G) and Laplacian (DP-L) differential private mechanism are employed. Gradients are 2-norm clipped with 0.2. Then, a Gaussian or Laplacian noise was added. The standard deviation of the applied noise ranges from 0.001 to 0.1. For GS, we evaluate various drop rates, from 99.0 to 99.9. For MARVELL, as it only supports binary classification, we evaluate its defense performance against our method CAE separately using data from only two classes. In our experiments, for MNIST and CIFAR datasets, we randomly selects two classes from the total dataset while class 'clouds' and 'person' are used for NUSWIDE dataset in order to balance the number of data samples from those two classes.



(a) MNIST (b) MNIST-binary (c) NUSWIDE (d) NUSWIDE-binary (e) CIFAR20 (f) CIFAR10-binary

Fig. 7. Main task accuracy vs. batch-level label inference accuracy under various defense strategies on multi-class classification task and binary classification task. The numbers on the figures are controlled variables (DP-G: σ, DP-L: b, GS: drop rate s, MARVELL: ratio of power constraint hyperparameter and gradient s, CAE: λ_2, DCAE: λ_2 with 12 bins).



(a) Passive Model Completion [11] (40 auxiliary data) (b) Passive Model Completion [11] (10 auxiliary data) (c) Active Model Completion [11] (40 auxiliary data) (d) Active Model Completion [11] (10 auxiliary data) (e) Direct (Sample-level) Label Inference [11] (f) Batch-level Label Inference (this work)

Fig. 8. Comparison of main task accuracy vs. label inference accuracy of different label inference attacks under various defense strategies. The numbers on the figures are controlled variables (DP-L: noise scale, GS: compression rate, PPDL: fraction of gathered gradient values, DG: number of discrete bins, CAE: $\lambda_2 = 1.0, 0.5, 0.0$, DCAE: $\lambda_2 = 1.0, 0.5, 0.0$ with 12 bins). Some points are not marked with numbers as the controlled variables are not sensitive for the result for CAE and DCAE (points from left to right matches $\lambda_2 = 1.0, 0.5, 0.0$ respectively).

The power constraint hyperparameter various from 1 to 10 times the norm of gradients. For PPDL, the fraction of gradients that are finally kept after the selection process, as described in [11], various from 0.1 to 0.75. For DG, the number of bins for gradient quantification various from 6 to 18.

7.3.2 Defense Results

Batch-level Label Inference Attack

The results of batch-level label inference defense of several strategies are shown in Figure 7. The left column of Figure 7 shows the results of defense under multi-class classification task (10, 5, 20 classes for MNIST, NUSWIDE and CIFAR separately). Because MARVELL defense can be only applied to binary classification tasks, we compare with it separately, shown in the right column. In each figure, the upper left indicates high attack performance and low main task accuracy, so a better defense should be towards the lower right. Compared to other approaches, we see that our CAE defense consistently achieves better trade-off performance, with high main task accuracy and low attack task accuracy for multi-class classification task, and relative the same main task accuracy and low attack accuracy for binary classification task. The above proves that CAE is superior in defending batch-level label inference attack. DCAE can defend the attack to the same degree as CAE but hurts the main task accuracy slightly, which is expected since quantizing the gradients leads to information loss which has a negative effect on the model's accuracy. Also, for this defending task, GS has a better performance than DP-based methods. And a change in the power constraint hyperparameter for MARVELL does not lead to a significant change in the result.

In Figure 8, we show the comparison results on the effectiveness of defending model completion attacks and label inference attacks of four baseline defense methods: i.e. DP-L (termed Noisy Gradients by FU [11]), GS (termed Gradients Compression by Fu [11]), PPDL and DG, as well as our defense method CAE and RCAE. We choose CIFAR10 as the dataset for experiments and all 10 classes are used. For our gradient-inversion-based batch-level label inference attack, we adopt a batch size of 2048, whereas for model completion attacks, 40 or 10 auxiliary labeled data are used which means 4 or 1 auxiliary labeled data for each class.

It's clear that our method CAE outperforms all the other defense methods for both batch-level label inference and direct label inference (sample-level label inference) attack with low attack accuracy and high main task accuracy. DG

method performs better at defending model completion tasks but is not effective for both label inference tasks. This is reasonable since DG keeps the sign of gradients to maintain its main task accuracy which can still be exploited for label inference attacks with both sample-level and batch-level gradients. On the other hand, CAE is not designed for scenarios where sufficient prior knowledge about labels can be exploited, therefore are not effective for defending model completion tasks. Since the attacker in model completion attacks knows additional auxiliary labeled data for each class, the prior knowledge of label distribution, it can learn a mapping between its model outputs and real labels directly.

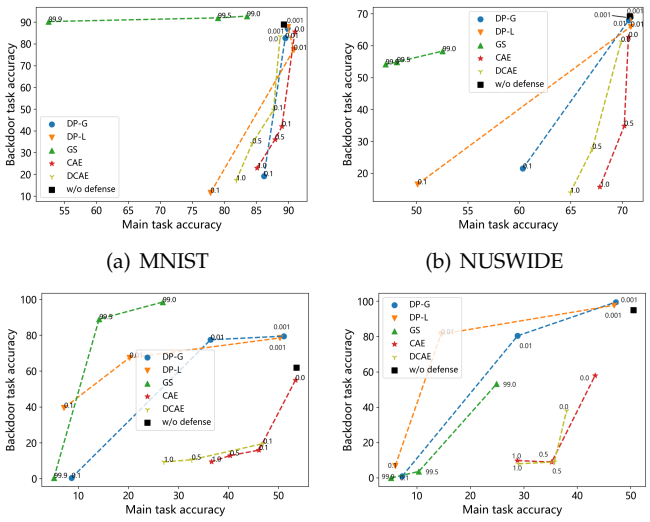
DCAE is designed to address such inference attacks. As described previously in section 6, DCAE takes full advantages of both DG and CAE methods. DCAE exhibits a better trade-off between main task and attack accuracy for passive and active model completion attacks as well as gradient-inversion-based label inference attacks compared to standalone DG and CAE respectively as shown in Figure 8. For both passive and active model completion attack, DCAE defense method achieves a 10% increase in the main task accuracy while suppressing the attack accuracy to a similar degree compared to DG, the second best choice, consistent with the analysis in section 6. With DCAE defense, the quantization of gradients severely harms the attacker's local model's ability in capturing label information, resulting in a decrease in attack accuracy. The main task accuracy, on the other hand, is only slightly harmed due to the fact that the non-discrete decoder Dec is the current last layer of the VFL model and Dec itself does not suffer from information loss. Also the joint training of VFL will boost contributions from local models and local decoders at the active party, resulting in higher main task accuracy compared to standalone DG defense. From the figure we can also see that for sample-level and batch-level label inference attack, DCAE achieves the lowest attack accuracy and the highest main task accuracy compared to all other defense methods except CAE. The fact that the main task accuracy of DCAE method is slightly lower than that of CAE is caused by the information loss with DG applied.

To sum up, our CAE defense is the most effective method for blocking both batch-level and sample-level label inference attack for VFL scenarios, when other defense methods perform unsatisfactorily. Also, CAE can be readily combined with information-reduction techniques such as DiscreteSGD (DG) method to boost the defense performance of passive and active model completion attacks, where it helps to increase the main task accuracy to a large extent while reaching comparable attack accuracy as that information-reduction method it combined with.

Label Replacement Backdoor Attack

We also evaluate our proposed label defense mechanisms on gradient-replacement attack and compared our methods with two baselines: Differential Privacy (DP) and Gradient Sparsification (GS). Both encoder and decoder of CAE and DCAE have the same architecture as displayed above. The hyper-parameters for the two baseline methods are the same as for label inference attack as well. The results are shown in Figure 9.

Same as the batch-level label inference attack, this experiment is carried out under multi-class classification task setting and a better defense should be towards the lower right in each figure. It's clear to see from Figure 9 that results of GS are in the upper left corner, indicating a high backdoor accuracy and low main task accuracy. This suggests that GS is inferior to defend our backdoor attack. Comparing to DP and GS mechanism, our CAE method can defend backdoor attack to the same level, while retaining a high accuracy of the main task, thanks to the confusion added for label replacement tasks. DCAE, the most robust method in defending all kinds of label inference attacks, is still effective in defending against label-replacement backdoor attack with slightly lower main task accuracy and backdoor task accuracy compared to CAE approach, caused by quantization of gradients.



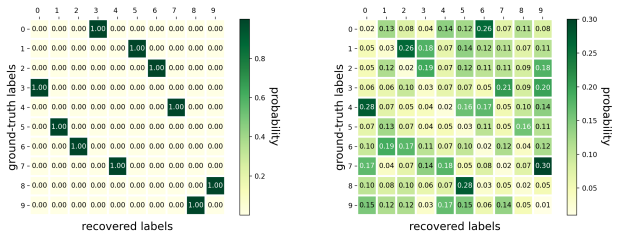
(a) MNIST (b) NUSWIDE

(c) CIFAR20 (d) CIFAR20-Distributed

Fig. 9. Main task accuracy vs. label replacement backdoor accuracy under various defense strategies. The numbers on the figures are controlled variables (DP-G: σ, DP-L: b, GS: drop rate s, CAE: λ_2, DCAE: λ_2 with 12 bins).

We further evaluate our defense strategies under distributed backdoor attack settings as the one proposed in previous work [31]. In this experiment, the feature of data is equally partitioned into four parties with only one active party owning the labels and three other passive parties without label information. Similar as previous work [31], the three passive parties work together and conduct label replacement backdoor attack to the active party. Each attacker has their one-pixel trigger at the lower right corner of each data sample of their own which together forming a three-pixel trigger. Notice that this trigger is smaller than the four-pixel trigger we use in previous experiments mentioned above. The result is shown in Figure 9(d). It's clear that our CAE and DCAE methods beats the other baseline methods, achieving a lower backdoor task accuracy at a high main task accuracy. Also, comparing with Figure 9(c), we show that the distributed backdoor attack is much stronger than two-party backdoor attack with higher attack accuracy achieved without any defense.

From Figure 9(c) and Figure 9(d), we can see that in label-replacement backdoor attack, as the noise level of DP or the drop rate of GS increases, backdoor accuracy decreases



(a) CAE without entropy loss (b) CAE with entropy loss

Fig. 10. The probability distribution (PD) over labels restored by the passive party when the CAE defense is applied at the active party. (a) PD matrix when CAE is trained without entropy loss (i.e. $\lambda_2 = 0$). (b) PD matrix when CAE is trained with entropy loss (i.e. $\lambda_2 = 1$).

at the expense of significant drop in main task accuracy. For backdoor attacks, backdoor accuracy decreases as the confusion level grows, indicating confusion is important for defending the attack. Without confusion ($\lambda_2 = 0$), the backdoor attack will still succeed since then the autoencoder's sole function is to switch labels among classes and the transformation would work the same for the backdoor samples and other samples.

7.3.3 Impact of Confusion

In this section, we analyze the impact of confusion coefficient. In defending batch-level label inference attacks, as the confusion coefficient increases, the label recovery rate gradually goes up, because higher confusion means higher probability for the soft fake labels to be mapped into the true labels.

Figure 10 depicts the probability distribution (PD) over labels restored by the passive party conducting batch-level label inference attack (Algorithm 2) on CIFAR10, when the active party is performing CAE protection (Algorithm 5). Specifically, each element in the PD matrix is computed by $\frac{C_{ij}}{\sum_j C_{ij}}$, where C_{ij} denotes the number of samples having ground-truth label i but is restored with label j. Figure 10(b) and Figure 10(a) depict the probability distribution when the CAE is trained with ($\lambda_2 = 1$) and without ($\lambda_2 = 0$) entropy loss $L_{entropy}$, respectively. Without entropy loss, the PD matrix is sparse so the attacker can learn all samples having the same label belong to the same class. With confusion, samples belonging to the same class are restored as multiple alternative labels, demonstrating that the passive party cannot classify its samples based on restored labels.

8 Conclusions

We propose novel gradient-inversion-based label inference and gradient-replacement backdoor attacks for vertical federated learning (VFL), achieving over 90% attack accuracy without either auxiliary labeled data or per-sample gradients. To tackle these attacks, we further propose confusional autoencoder (CAE), and demonstrate that CAE defense mechanism is the most effective in defending gradient-inversion-based batch-level label inference attack and label replacement backdoor attack without noticeable changes to other collaborative parties and the VFL training protocol. We also show that CAE can help boost main task accuracy when integrated with information-reduction techniques such as DiscreteSGD. The resulting DCAE defense can achieve a low attack accuracy without hurting the main task accuracy as much as other approaches in various label inference and label replacement backdoor task which makes it the most robust defense method against existing attack methods.

Acknowledgments

This project is supported in part by the Tsinghua (AIR)-Asiainfo Technologies (China) Research Center under grant No. 20203910074.

References

[1] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Survey*, 51(4):79:1–79:35, 2018.

[2] Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. *arXiv preprint arXiv:1704.05021*, 2017.

[3] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. *CoRR*, abs/1807.00459, 2018.

[4] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 634–643, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[5] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, Dimitrios Papadopoulos, and Qiang Yang. Secureboost: A lossless federated learning framework. *IEEE Intelligent Systems*, pages 1–1, 2021.

[6] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. Nus-wide: a real-world web image database from national university of singapore. In *Proceedings of the ACM international conference on image and video retrieval*, pages 1–9, 2009.

[7] Nikoli Dryden, Tim Moon, Sam Ade Jacobs, and Brian Van Essen. Communication quantization for data-parallel training of deep neural networks. In *2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC)*, pages 1–8. IEEE, 2016.

[8] Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.

[9] Cynthia Dwork. Differential privacy. *Encyclopedia of Cryptography and Security*, pages 338–340, 2011.

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

[11] Chong Fu, Xuhong Zhang, Shouling Ji, Jinyin Chen, Jingzheng Wu, Shanqing Guo, Jun Zhou, Alex X Liu, and Ting Wang. Label inference attacks against vertical federated learning. In *31st USENIX Security Symposium (USENIX Security 22)*, Boston, MA, August 2022. USENIX Association.

[12] Wei Gao, Shangwei Guo, Tianwei Zhang, Han Qiu, Yonggang Wen, and Yang Liu. Privacy-preserving collaborative learning with automatic transformation search. *CoRR*, abs/2011.12505, 2020.

[13] Cristiano Gratton, Naveen KD Venkategowda, Reza Arablouei, and Stefan Werner. Distributed ridge regression with feature partitioning. In *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, pages 1423–1427. IEEE, 2018.

[14] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *CoRR*, abs/1708.06733, 2017.

[15] Chaoyang He, Murali Annavaram, and Salman Avestimehr. Group knowledge transfer: Federated learning of large cnns at the edge. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 14068–14080. Curran Associates, Inc., 2020.

[16] Yaochen Hu, Peng Liu, Linglong Kong, and Di Niu. Learning privately over distributed features: An ADMM sharing approach. *CoRR*, abs/1907.07735, 2019.

[17] Yaochen Hu, Di Niu, Jianming Yang, and Shengping Zhou. Fdml: A collaborative machine learning framework for distributed features. In *KDD*, 2019.

[18] Peter Kairouz et al. Advances and open problems in federated learning. *ArXiv*, abs/1912.04977, 2019.

[19] Hiroaki Kikuchi, Chika Hamanaga, Hideo Yasunaga, Hiroki Matsui, and Hideki Hashimoto. Privacy-preserving multiple linear regression of vertically partitioned real medical datasets. *2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*, pages 1042–1049, 2017.

[20] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009.

[21] Oscar Li, Jiankai Sun, Xin Yang, Weihao Gao, Hongyi Zhang, Junyuan Xie, Virginia Smith, and Chong Wang. Label leakage and protection in two-party split learning. *CoRR*, abs/2102.08504, 2021.

[22] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2017.

[23] Yang Liu, Tianjian Chen, and Qiang Yang. Secure federated transfer learning. *CoRR*, abs/1812.03337, 2018.

[24] Yang Liu, Tao Fan, Tianjian Chen, Qian Xu, and Qiang Yang. Fate: An industrial grade platform for collaborative learning with data protection. *Journal of Machine Learning Research*, 22(226):1–6, 2021.

[25] Yang Liu, Yan Kang, Xin wei Zhang, Liping Li, Yong Cheng, Tianjian Chen, M. Hong, and Q. Yang. A communication efficient collaborative learning framework for distributed features. *arXiv: Learning*, 2019.

[26] Yang Liu, Zhihao Yi, and Tianjian Chen. Backdoor attacks and defenses in feature-partitioned collaborative learning. *CoRR*, abs/2007.03608, 2020.

[27] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *CoRR*, abs/1602.05629, 2016.

[28] R L Rivest, L Adleman, and M L Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation*, pages 169–179, 1978.

[29] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H. Brendan McMahan. Can you really backdoor federated learning? *ArXiv*, abs/1911.07963, 2019.

[30] Boxin Wang, Shuohang Wang, Yu Cheng, Zhe Gan, Ruoxi Jia, Bo Li, and Jingjing Liu. Infobert: Improving robustness of language models from an information theoretic perspective. *arXiv preprint arXiv:2010.02329*, 2020.

[31] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. {DBA}: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2020.

[32] 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.

[33] Christopher J.C. Burges Yann LeCun, Corinna Cortes. The mnist dataset. http://yann.lecun.com/exdb/mnist/index.html.

[34] Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M. Alvarez, Jan Kautz, and Pavlo Molchanov. See through gradients: Image batch recovery via gradinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16337–16346, June 2021.

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

[36] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *CoRR*, abs/1906.08935, 2019.

Tianyuan Zou is an undergraduate student from Tsinghua University, China, majoring in Computer Science and Technology. In the next few years, she will work toward her Ph.D. degree with Institute for AI Industry Research, Tsinghua University. Her current research interest focuses on the privacy and safety in federated learning.

Yang Liu is an Associate Professor with Institute for AI Industry Research, Tsinghua University. Before joining Tsinghua, she was the Principal Researcher and Research Team Lead with WeBank. Her research interests include federated learning, machine learning, multi-agent systems, statistical mechanics and AI industrial applications. Her research work was recognized with multiple awards, such as AAAI Innovation Award and CCF Technology Award.

Yan Kang is currently the research team lead with the AI Department of WeBank, Shenzhen, China. His works focus on the research and implementation of transfer learning and secure federated machine learning. His research works were authored or coauthored in well-known conferences and journals including IEEE Intelligence Systems, IJCAI, and ACM TIST, and he coauthored the Federated Learning book.

Wenhan Liu is an undergraduate student at Shandong University. He has been recommended to Gaoling School of Artificial Intelligence, Renmin University of China for a Ph.D. degree. His major research interests include Federated Learning, Information Retrieval and Personalized Search.

Yuanqin He received the B.S. degree from Shanghai Jiao Tong University, and the Ph.D. degree in Physics from Technical University of Munich. He is currently an Researcher with WeBank. His research interests include machine learning and federated learning.

Zhihao Yi received the B.S. degree from Hunan University, and master degree in computer science and Technical from Beihang University. He is currently an researcher in WeBank. His research interests include machine learning and federated learning.

Qiang Yang is a fellow of Royal Society of Canada (RSC) and Canadian Academy of Engineering (CAE), Chief Artificial Intelligence Officer of WeBank, a Chair Professor of Computer Science and Engineering Department at Hong Kong University of Science and Technology (HKUST). He is a fellow of AAAI, ACM, CAAI, IEEE, IAPR, AAAS. His research interests are artificial intelligence, machine learning, data mining and planning. His latest books are Transfer Learning, Federated Learning and Practicing Federated Learning.

Ya-Qin Zhang is Chair Professor of AI Science at Tsinghua University, and Dean of Institute for AI Industry Research of Tsinghua University (AIR). He served as the President of Baidu Inc. for 5 years. Prior to Baidu, he was an executive at Microsoft for 16 years. He is the Chinese Academy of Engineering (CAE), the American Academy of Arts and Sciences (AA) and Australian Academy of Engineering (ATSE). He is also a fellow of IEEE.