PlugVFL: Robust and IP-Protecting Vertical Federated Learning against Unexpected Quitting of Parties

Jingwei Sun
Duke University
jingwei.sun@duke.edu

Zhixu Du
Duke University
zhixu.du@duke.edu

Anna Dai
Duke University
anna.dai@duke.edu

Saleh Baghersalimi
École Polytechnique Fédérale de Lausanne
saleh.baghersalimi@epfl.ch

Alireza Amirshahi
École Polytechnique Fédérale de Lausanne
alireza.amirshahi@epfl.ch

David Atienza
École Polytechnique Fédérale de Lausanne
david.atienza@epfl.ch

Yiran Chen
Duke University
yiran.chen@duke.edu

2024 IEEE International Conference on Big Data (BigData) | 979-8-3503-6248-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/BigData62323.2024.10825534

***Abstract*—In federated learning systems, the unexpected quitting of participants is inevitable. Such quittings generally do not incur serious consequences in horizontal federated learning (HFL), but they do damage to vertical federated learning (VFL), which has been underexplored in previous research. In this paper, we show that there are two major vulnerabilities when passive parties unexpectedly quit in the deployment phase of VFL — severe performance degradation and intellectual property (IP) leakage of the active party's labels. To solve these issues, we design PlugVFL to improve the VFL model's robustness against the unexpected exit of passive parties and protect the active party's IP in the deployment phase simultaneously. We evaluate our framework on multiple datasets against different inference attacks. The results show that PlugVFL effectively maintains model performance after the passive party quits and successfully disguises label information from the passive party's feature extractor, thereby mitigating IP leakage.**

***Index Terms*—Federated Learning, Data Privacy, IP Protection**

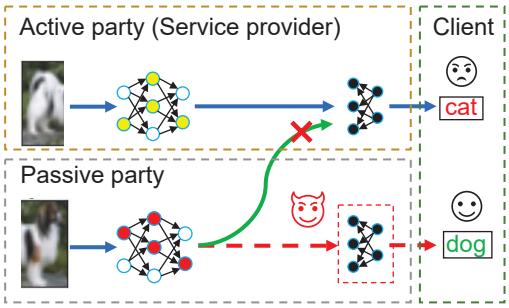

Fig. 1: The passive party might quit in the deployment phase, which would cause a substantial performance drop. The passive party could also extract representations containing the information of the active party's labels using its feature extractor, leading to IP leakage of the active party.

I. Introduction

Federated learning (FL) [1]–[5] is a distributed learning method that allows multiple parties to collaboratively train a model without directly sharing their data, thereby preserving their data privacy. FL was initially proposed as Horizontal Federated Learning (HFL) to enable collaborative learning across devices [6]. In this case, data is "horizontally" split, where the devices share the same feature space but have different samples. Another FL framework is Vertical Federated Learning (VFL) [7]–[15], which focuses on scenarios where various parties have data with different feature spaces but share overlapping samples [16], [17]. Different from HFL, VFL is mostly deployed in cross-silo scenarios. Suppose a service provider, referred to as active party, owns data and labels of its clients and wishes to train a deep learning model. The service provider may collaborate with other parties, namely passive parties, that possess different data features of the same clients to boost the model's performance. Instead of explicitly sharing the raw data, the passive parties transmit the extracted representations to the active party for training and inference.

The collaboration of devices in HFL happens in the training phase, and the global model is deployed on each device for local inference in the deployment phase. In contrast, VFL requires the parties to collaborate in both the training and deployment phases. During the deployment phase, the active party still requires the representations uploaded by passive parties to conduct inference. However, in real-world scenarios, it is possible for passive parties to quit unexpectedly at inference time due to network crashes, system maintenance, or termination of collaborations. When unexpected quitting happens, the service provider faces two challenges: (1) a substantial **performance drop**; (2) potential **intellectual property (IP) leakage** through the passive party's feature extractor. This paper shows that the drop in model performance caused by the passive party's quitting results in a model that performs worse than one trained by the active party alone, ultimately undermining the motivation for VFL. Furthermore, the passive parties can retain access to their feature extractors even after terminating the collaboration. **These feature extractors are**

trained using the active party's labels, which are valuable IP. From these feature extractors, the passive parties can extract representations containing the information of the active party's labels. Although previous studies made efforts towards mitigating label information leakage through inference attacks on gradients during the training phase [18], [19], the robustness and IP protection of VFL in the deployment phase remain under-explored.

In this paper, we design a framework named PlugVFL to solve the two challenges simultaneously. Specifically, to alleviate the performance drop when passive parties quit unexpectedly, PlugVFL applies an alternative training method that can reduce the co-adaptation of feature extractors across parties. To prevent the IP leakage of the active party's labels, we propose a defense that minimizes the mutual information (MI) between the representations of the passive party and the true labels. We formulate the defense into an adversarial training algorithm that jointly minimizes the variational MI upper bound and prediction loss.

Our key contributions are summarized as follows:

- We reveal two vulnerabilities caused by the unexpected quitting of parties in the deployment phase of VFL, including severe performance drop and active party's label leakage.
- we design a VFL framework PlugVFL to preserve the VFL model's performance against the unexpected exit of passive parties and protect the active party's IP in the deployment phase simultaneously.
- We empirically evaluate the performance of our framework with different datasets. Our results show that PlugVFL can improve the accuracy after the passive party's exit by more than 8% on CIFAR10. PlugVFL also prevents the passive party from fine-tuning a classifier that outperforms random guess levels even using the entire labeled dataset with only less than 2% drop in the VFL model accuracy, outperforming baselines significantly.

II. Related Work

A. Vertical Federated Learning

Vertical federated learning (VFL) [7], [20] has been an emerging research area since proposed. In contrast to (horizontal) federated learning (HFL), VFL adopts a different scheme for data partitioning [2], [21]. In VFL, different parties will have various parts of the data of an overlapping individual. There has been an amount of research devoted to VFL. Specifically, [21] proposes a protocol involving a trusted third party to manage the communication utilizing homomorphic encryption, with the following works [21], [22] on protocols design. Others have been following [21], where [22] is working on assessing the protocols and [23], [24] are focusing on algorithm design concerning optimization. Additionally, VFL algorithms on traditional machine learning, such as tree-boosting [25], gradient boosting [26], [27], random forest [28], linear regression [29], and logistic regression [16], [30] are also proposed. Another line of research is working on communication efficiency [16], which decreases the communication frequency by leveraging stale gradients on local training. Besides, the assumption of overlapping individuals in VFL among parties produces a challenge for applying VFL in the real world, where FedMVT [31] proposes to estimate representations and labels to alleviate the gap. Other efforts have also been made to apply VFL in the real world. For example, FATE [17] is an open-source platform for building the end-to-end system.

B. IP Leakage in VFL

Intellectual Property (IP) [32]–[34] is drawing more and more attention as the rapid growth of commercial deployment of deep learning, especially in federated learning scenarios, whose primary concern is privacy. IP leakage can be divided into data IP leakages, such as deep leakage from gradients (DLG) [35], [36], model inversion [37] and their variants [38]–[42], and model IP leakage, such as model extraction attacks [43]–[47], where multiple defensive methods have also been proposed to tackle data IP leakage [48]–[51] and model IP leakage [52], [53].

In VFL, we categorize IP stealing attacks into two types, i.e., feature inference [39], [42], [54], [55] and label inference [56]–[58]. Specifically, [54] proposes general attack methods for complex models, such as Neural Networks, by matching the correlation between adversary features and target features, which can be seen as a variant of model inversion [4], [37]. [42], [55] also propose variants of model inversion attack in VFL. While all these attacks are in the inference phase, [39] proposes a variant of DLG [35] which can perform attacks in the training phase. For label inference, [57] proposes an attack method and a defense method for two-party split learning on binary classification problems, a special VFL setting. Additionally, [56] proposes three different label inference attack methods considering different settings in VFL: direct label inference attack, passive label inference attack, and active label inference attack. Defensive methods have also been proposed. For example, [58] proposes manipulating the labels following specific rules to defend the direct label inference attack, which can be seen as a variant of label differential privacy (label DP) [18], [19] in VFL. However, all these defending methods focus on preventing data IP leakage from gradients in the training phase. To the best of our knowledge, we are the first to provide an analysis of label IP protection in the VFL deployment phase.

III. Problem Definition and Motivation

A. Vertical Federated Learning Setting

Suppose K parties train a model. There is a dataset[1] across all parties with size N: $D = \{x_i, y_i\}_{i=1}^{N}$. The feature vector $x_i \in \mathbb{R}^d$ is splitted among K parties $\{x_i^k \in \mathbb{R}^{d_k}\}_{k=1}^{K}$, where d_k is the feature dimension of party k, and the labels $Y = \{y_i\}_{i=1}^{N}$ are owned by one party. The parties with only features are

[1]We assume the alignment between overlapping samples is known as a prior. In some applications, private set intersection could be used before running VFL to find the sample alignment.

referred to as *passive parties*, and the party with both features and labels is referred to as the *active party*. We denote party 1 as the active party, and other parties are passive parties.

Each party (say the k-th) adopts a representation extractor $f_{\theta_k}(\cdot)$ to extract representations of local data $H^k = \{H_i^k\}_{i=1}^N = \{f_{\theta_k}(x_i^k)\}_{i=1}^N$ and sends them to the active party, who possesses labels and a predictor. The overall training objective of VFL is formulated as

$$\min_{\Theta} \mathcal{L}(\Theta; D) \triangleq \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\left(\mathcal{S}_{\theta_{\mathcal{S}}}\left(H_i^1, ..., H_i^K\right), y_i\right), \quad (1)$$

where $\Theta = [\theta_1; ...; \theta_K; \theta_{\mathcal{S}}]$, $\mathcal{S}$ denotes a trainable head model on active party to conduct classification, and $\mathcal{L}$ denotes the loss function. The objective of each passive party k is to find an optimal θ_k^* while not sharing local data $\{x_i^k\}_{i=1}^N$ and parameters θ_k. The objective of the active party is to optimize θ_1 and $\theta_{\mathcal{S}}$ while not sharing θ_1, $\theta_{\mathcal{S}}$ and true labels Y. The active party calculates the gradients of received representations and send $\{\frac{\partial \mathcal{L}}{\partial H^k}\}_{k\in[2,...,K]}$ back to passive parties.

Notably, the passive parties still have to communicate with the active party during the inference phase. For a new data x_i, the passive parties send the extracted representations $\{H_i^k\}_{k\in[2,...,K]}$ to the active party, and the active party generates the prediction $\mathcal{S}_{\theta_{\mathcal{S}}}\left(H_i^1, ..., H_i^K\right)$.

B. Performance drop after parties quit

During the deployment phase, some passive parties (say the k-th party) could quit unexpectedly due to a network crash or the termination of collaboration. Without the representations uploaded by party k, the active party can still conduct inference by setting H_i^k as a zero vector. However, there will be a substantial performance drop. We conduct two-party experiments on CIFAR10 to investigate this performance drop. We follow previous works [16], [59] to split CIFAR10 images into two parts and assign them to the two parties using ResNet18 as backbone models. The active party (party 1) and passive party (party 2) collaborate to train the models. We evaluate and compare the inference accuracy before and after party 2 quits in the deployment phase. When party 2 quits, party 1 sets H_i^2 as a zero vector and conducts inference. Zero vectors are used because the passive party typically does not allow the active party to utilize its representations in any way (e.g., an average vector) after the termination of collaboration. We set the standalone results as a baseline, where the active party trains a model independently without ever collaborating with the passive party.

TABLE I: Compared results before and after party 2 quits on CIFAR10.

	Accuracy(%)
Before party 2 quits	74.53
After party 2 quits	51.24
Party 1 standalone	62.84

The results (shown in Tab. I) demonstrate that the accuracy drops more than 20% after party 2 quits. Furthermore, the VFL model after party 2 quits achieves even lower accuracy than the model party 1 trained without any collaboration, undermining the motivation of VFL.

C. IP leakage of labels in the deployment phase

The collaborative training process enables passive parties to extract representations useful for the task of VFL, which is learned from the labels of the active party. Even after the collaboration ends, the passive parties will retain access to the representation extractors. These extractors allow the passive parties to fine-tune classifier heads with very few labeled data and conduct inference with decent accuracy after quitting the collaboration. Given the active party's significant investment of effort and money in labeling the data, these extractors retained by the passive parties constitute costly IP leakage of these labels. To demonstrate the extent of IP leakage by the feature extractors of the passive parties, we follow the experimental setup in Sec. III-B and let party 2 conduct model completion (MC) attack [60] to train a classifier using a small number of labeled samples. We report the test accuracy of the complete model of party 2 created by the MC attack. For comparison, we also assume party 2 annotates all the training data to train a model from scratch.

TABLE II: Compared accuracy of the model on party 2 by conducting MC attack and collecting labels to train from scratch.

	Accuracy(%)
MC attack (400 labels)	58.02
Train from scratch w. all the labels	59.73

We report the accuracy in Tab. II. By fine-tuning a classifier with the extractor, the passive party can achieve comparable accuracy using less than 1% of the labeled data compared to training a model from scratch with all the labels. This demonstrates that the label information from the active party is leaked and embedded in the passive party's extractor.

IV. Method

A. Overview of PlugVFL

Without loss of generality, we formulate our PlugVFL framework in the two-party scenario. Suppose the passive party (party 2) and active party (party 1) have sample pairs $\left\{\left(x_i^1, x_i^2, y_i\right)\right\}_{i=1}^N$ drawn from a distribution $p\left(x^1, x^2, y\right)$, and the representations of party k is calculated as $h^k = f_{\theta_k}(x^k)$. We use h^k, x^k and y here to represent random variables, while H_i^k, x_i^k and y_i stand for deterministic values. Then the training of our framework is to achieve three goals:

- Goal 1: To preserve the performance of VFL, the main objective loss should be minimized.
- Goal 2: To preserve the performance after party 2 quits, the objective loss without the representations of party 2 should be minimized.

- Goal 3: To reduce the IP leakage of labels from party 2, θ_2 should not be able to extract representations h^2 containing much information about the true label y.

Formally, we have three training objectives:

$$\begin{aligned}
&\textbf{Prediction performance: } \min_{\theta_1,\theta_2,\theta_S} \mathcal{L}\left(\mathcal{S}_{\theta_S}\left(h^1,h^2\right),y\right),\\
&\textbf{Robustness against quitting: } \min_{\theta_1,\theta_2,\theta_S} \mathcal{L}\left(\mathcal{S}_{\theta_S}\left(h^1,0\right),y\right),\\
&\textbf{Label IP protection: } \min_{\theta_2} \mathrm{I}(h^2;y),
\end{aligned} \tag{2}$$

where $\mathcal{S}_{\theta_S}\left(h^1,0\right)$ is the prediction when the server does not receive the representations h^2 uploaded by party 2. $\mathrm{I}(h^2;y)$ is the mutual information between h^2 and y, which indicates the information h^2 preserves for the label variable y. We minimize this mutual information to protect the active party's labels' IP from being steal by the passive party.

B. Efficient alternative training to Achieve Robustness

A trivial way to improve the robustness against quitting is to combine the training objectives $\mathcal{L}\left(\mathcal{S}_{\theta_S}\left(h^1,h^2\right),y\right)$ and $\mathcal{L}\left(\mathcal{S}_{\theta_S}\left(h^1,0\right),y\right)$ on the server. However, the server has to conduct training of $\mathcal{S}_{\theta_S}$ twice, which involves computational overhead. To improve the efficiency, we propose an alternative training method to achieve the second goal. Specifically, for each communication round (i.e., an iteration of training in VFL), the active party omits the representations from the passive party with probability p. The expectation of the training objective is formulated as

$$\begin{aligned}
&\mathbb{E}_p\mathcal{L}(\Theta;D)\\
&=(1-p)\,\mathcal{L}\left(\mathcal{S}_{\theta_S}\left(h^1,h^2\right),y\right)+p\mathcal{L}\left(\mathcal{S}_{\theta_S}\left(h^1,0\right),y\right),
\end{aligned} \tag{3}$$

which is a weighted sum of the first and the second goal with weight p. Notably, a larger p sets a larger weight for $\mathcal{L}\left(\mathcal{S}_{\theta_S}\left(h^1,0\right),y\right)$. Thus, p can be chosen based on the chance that party 2 quits.

The intuition behind alternative training is to reduce the co-adaptation between the head predictor and local extractors. The severe performance drop after the quitting of passive parties comes from the co-adaptation of the hidden neurons of the head predictor $\mathcal{S}_{\theta_S}$ and the neurons of local extractors f_{θ_k}, where a hidden neuron of the predictor $\mathcal{S}_{\theta_S}$ only depends on the pattern of several specific neurons of specific parties' extractors. The *dropout* was proposed as an effective solution to co-adaptation [61]. Similar to *dropout*, which omits some neurons, our proposed alternative training method omits the passive party, which solves the party-wise co-adaptation in VFL.

C. Variational Training Objective of Label Protection

The mutual information term (i.e., goal 3) is hard to compute in practice as the random variable h^2 is high-dimensional. In addition, computing mutual information requires knowing the distribution $p(y|h^2)$, which is difficult to obtain. To derive a tractable estimation of the mutual information objective, we leverage *CLUB* [62] to formulate a variational upper-bound:

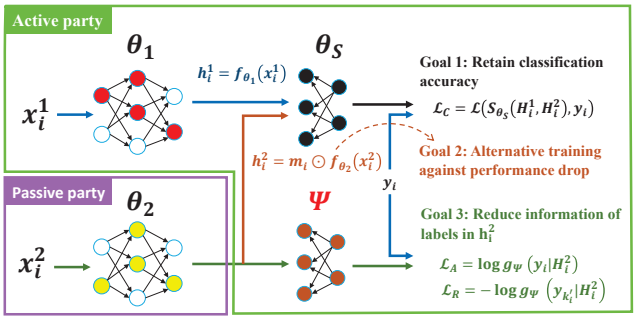

Fig. 2: Overview of PlugVFL: The active party conducts alternative training to preserve the performance against the exit of the passive party. IP of labels is protected by optimizing $\mathcal{L}_A$ and $\mathcal{L}_R$ to reduce label information in passive parties' representations.

$$\begin{aligned}
&\mathrm{I}\left(h^2;y\right)\\
&\le \mathrm{I}_{\mathrm{vCLUB}}\left(h^2;y\right)\\
&:= \mathbb{E}_{p(h^2,y)}\log q_\psi\left(y|h^2\right)-\mathbb{E}_{p(h^2)p(y)}\log q_\psi\left(y|h^2\right),
\end{aligned} \tag{4}$$

where $q_\psi\left(y|h^2\right)$ is a variational distribution with parameters ψ to approximate $p\left(y|h^2\right)$. To reduce the computational overhead of the defense, we apply the sampled vCLUB (vCLUB-S) MI estimator in [62], which is an unbiased estimator of $\mathrm{I}_{\mathrm{vCLUB}}$ and is formulated as

$$\begin{aligned}
&\hat{\mathrm{I}}_{\text{vCLUB-S}}(h^2;y)\\
&=\frac{1}{N}\sum_{i=1}^{N}\left[\log q_\psi\left(y_i|H_i^2\right)-\log q_\psi\left(y_{k_i'}|H_i^2\right)\right],
\end{aligned} \tag{5}$$

where k_i' is uniformly sampled from indices $\{1,...,N\}$. It is notable that to guarantee the first inequality of Eq. (4), $q_\psi\left(y|h^2\right)$ should satisfy

$$\mathrm{KL}\left(p\left(h^2,y\right)||q_\psi\left(h^2,y\right)\right)\le \mathrm{KL}\left(p\left(h^2\right)p\left(y\right)||q_\psi\left(h^2,y\right)\right), \tag{6}$$

which can be achieved by minimizing $\mathrm{KL}\left(p\left(h^2,y\right)||q_\psi\left(h^2,y\right)\right)$:

$$\begin{aligned}
&\min_\psi \mathrm{KL}\left(p\left(h^2,y\right)||q_\psi\left(h^2,y\right)\right)\\
&=\min_\psi \mathbb{E}_{p(h^2,y)}\left[\log\left(p\left(y|h^2\right)p\left(h^2\right)\right)-\log\left(q_\psi\left(y|h^2\right)p\left(h^2\right)\right)\right]\\
&=\min_\psi \mathbb{E}_{p(h^2,y)}\left[\log\left(p\left(y|h^2\right)\right)-\log\left(q_\psi\left(y|h^2\right)\right)\right].
\end{aligned} \tag{7}$$

Since the first term has no relation to ψ, we just need to minimize $\mathbb{E}_{p(h^2,y)}-\log\left(q_\psi\left(y|h^2\right)\right)$. With samples $\left\{\left(x_i^1,x_i^2,y_i\right)\right\}_{i=0}^N$, we can derive an unbiased estimation

$$\max_\psi \frac{1}{N}\sum_{i=1}^{N}\log q_\psi\left(y_i|H_i^2\right). \tag{8}$$

With Eq. (4), Eq. (5) and Eq. (8), the objective of label IP protection can be achieved by optimizing

$$\begin{aligned} & \min_{\theta_2} \mathrm{I}(h^2; y) \\ \Leftrightarrow \ & \min_{\theta_2} \hat{\mathrm{I}}_{\text{vCLUB-S}}(h^2; y) \\ = \ & \min_{\theta_2} \frac{1}{N} \sum_{i=1}^{N} \left[\max_{\psi} \log q_\psi \left(y_i | H_i^2 \right) - \log q_\psi \left(y_{k_i'} | H_i^2 \right) \right]. \end{aligned} \tag{9}$$

Suppose we use g_ψ to parameterize q_ψ, by combining Eq. (9) and the prediction objective with a weight hyper-parameter λ, we formulate the overall optimizing objective as

$$\begin{aligned} & \min_{\theta_1, \theta_2, \theta_S} (1-\lambda) \underbrace{\frac{1}{N} \sum_{i=1}^{N} \mathcal{L} \left(\mathcal{S}_{\theta_S} \left(f_{\theta_1} \left(x_i^1 \right), f_{\theta_2} \left(x_i^2 \right) \right), y_i \right)}_{\mathcal{L}_C} \\ & + \min_{\theta_2} \max_{\psi} \lambda \underbrace{\frac{1}{N} \sum_{i=1}^{N} \log g_\psi \left(y_i | f_{\theta_2} \left(x_i^2 \right) \right)}_{\mathcal{L}_A} \\ & + \min_{\theta_2} \lambda \underbrace{\frac{1}{N} \sum_{i=1}^{N} - \log g_\psi \left(y_{k_i'} | f_{\theta_2} \left(x_i^2 \right) \right)}_{\mathcal{L}_R}. \end{aligned} \tag{10}$$

It is notable that our defense is not limited to any specific type of task. For the classification problem, $-\mathcal{L}_A$ and $\mathcal{L}_R$ are equal to cross-entropy losses.

D. Training procedure

The overall objective has three terms. The first term is the prediction objective. The second term is an adversarial training objective, where an auxiliary predictor g_ψ is trained to capture label information while the feature extractor f_{θ_2} is trained to extract as little label information as possible. The third term regularizes f_{θ_2} to capture the information of a randomly selected label. For simplicity, we denote these three objective terms as $\mathcal{L}_C$, $\mathcal{L}_A$ and $\mathcal{L}_R$, respectively, as shown in Eq. (10). The adversarial loss $\mathcal{L}_A$ and $\mathcal{L}_R$ are formulated from the goal of label protection. We reorganize the overall training objective as

$$\begin{aligned} & \theta_1, \theta_2, \theta_S, \psi \\ = \ & \arg\min_{\theta_2} \left[(1-\lambda) \min_{\theta_1, \theta_S} \mathcal{L}_C + \lambda \max_{\psi} \mathcal{L}_A + \lambda \mathcal{L}_R \right]. \end{aligned} \tag{11}$$

We develop an algorithm of label-protecting training to optimize Eq. (11), summarized in Alg. 1. For each batch of data, we first optimize θ_1 and θ_S based on the primary task loss. Then we optimize the auxiliary predictor ψ. Finally, θ_2 is optimized with $(1-\lambda)\mathcal{L}_C + \lambda\mathcal{L}_A + \lambda\mathcal{L}_R$.

Note that θ_1, θ_S and ψ are owned by the active party, and their optimization does not require additional information from the passive party except the representations h^2, which should be uploaded to the active party even without defense. For the passive party, the training procedure of local extractor θ_2 does not change, making our defense concealed from the passive party.

Algorithm 1 Training algorithm of PlugVFL. $\leftarrow$ means information is sent to the active party; $\leftarrow$ means information is sent to the passive party; red steps are conducted on the passive party.

Input: Dataset $\left\{ \left(x_i^1, x_i^2, y_i \right) \right\}_{i=1}^{N}$; Learning rate η.
Output: $\theta_1; \theta_S; \psi$.

Initialize $\theta_1; \theta_S; \psi$;
for a batch of data $\left\{ \left(x_i^1, x_i^2, y_i \right) \right\}_{i \in \mathbb{B}}$ **do**
  $\{H_i^2\}_{i \in \mathbb{B}} \leftarrow \{ f_{\theta_2} \left(x_i^2 \right) \}_{i \in \mathbb{B}}$;
  $\mathcal{L}_A \leftarrow \frac{1}{|\mathbb{B}|} \sum_{i \in \mathbb{B}} \log g_\psi \left(y_i | H_i^2 \right)$;
  $\psi \leftarrow \psi + \eta \nabla_\psi \mathcal{L}_A$;
  Randomly generate binary mask vectors $\{m_i\}$ with probability p to be zeros;
  $\mathcal{L}_C \leftarrow \frac{1}{|\mathbb{B}|} \sum_{i \in \mathbb{B}} \mathcal{L} \left(\mathcal{S}_{\theta_S} \left(f_{\theta_1} \left(x_i^1 \right), m_i \odot H_i^2 \right), y_i \right)$;
  $\theta_1 \leftarrow \theta_1 - \eta \nabla_{\theta_1} \mathcal{L}_C$;
  $\theta_S \leftarrow \theta_S - \eta \nabla_{\theta_S} \mathcal{L}_C$;
  $\{y_{k_i'}\}_{i \in \mathbb{B}} \leftarrow$ randomly sample $\{y_{k_i'}\}_{i \in \mathbb{B}}$ from $\{y_i\}_{i \in [N]}$.
  $\mathcal{L}_R \leftarrow \frac{1}{|\mathbb{B}|} \sum_{i \in \mathbb{B}} - \log g_\psi \left(y_{k_i'} | H_i^2 \right)$;
  $\{\nabla_{H_i^2} \mathcal{L}\}_{i \in \mathbb{B}} \leftarrow \{\nabla_{H_i^2} [(1-\lambda)\mathcal{L}_C + \lambda(\mathcal{L}_A + \mathcal{L}_R)]\}_{i \in \mathbb{B}}$;
  $\nabla_{\theta_2} \mathcal{L} \leftarrow \frac{1}{|\mathbb{B}|} \sum_{i \in \mathbb{B}} \nabla_{H_i^2} \mathcal{L} \nabla_{\theta_2} H_i^2$
  $\theta_2 \leftarrow \theta_2 - \eta \nabla_{\theta_2} \mathcal{L}$;
end for

E. Theoretical analysis

We also derive some theoretical results of the label IP protection. Let g_ψ parameterize q_ψ in Eq. 5. Suppose the passive party optimizes an auxiliary model $f^m(y|h^2)$ to estimate $p(y|h^2)$. For any f^m, we have:

$$\frac{1}{N} \sum_{i=1}^{N} \log f^m(y_i | H_i^2) < \frac{1}{N} \sum_{i=1}^{N} \log p(y_i) + \epsilon, \tag{12}$$

where

$$\epsilon = \mathrm{I}_{\text{vCLUB}_{g_\psi}}(h^2; y) + \mathrm{KL}(p(y|h^2) || g_\psi(y|h^2)). \tag{13}$$

Notably, this theorem does not set any constraints or assumptions on the type of label. Specifically, if the task of collaborative inference is classification, we have the following results:

$$\frac{1}{N} \sum_{i=1}^{N} \mathrm{CE} \left[f^m(H_i^2), y_i \right] > CE_{random} - \epsilon, \tag{14}$$

where CE denotes the cross-entropy loss, CE_{random} is the cross-entropy loss of random guessing. This theoretical analysis results demonstrate that by applying our algorithm, the passive

party cannot solely infer the collaborative task results with a high performance, which prevents the label IP leakage from the passive party. The proof can be found in Appendix A.

V. Experiments

We evaluate our proposed PlugVFL on multiple datasets. We focuse on two-party scenarios following the VFL literature [2], [16], [17], [28], [56].

Baselines. To thoroughly analyze PlugVFL, we first evaluate PlugVFL against performance drop and label leakage separately and compare with the baselines achieving the same goal, respectively. To our knowledge, PlugVFL is the first approach to mitigate the performance drop after the passive party quits in VFL. But we still compare with a baseline, where the active party trains an additional head model without the quitting party to make predictions if the passive party quits, which we call *Multi-head training*. For defense against label leakage, we evaluate PlugVFL against two attacks: (1) *Passive Model Completion (PMC)* [56] attack assumes that the passive party has access to an auxiliary labeled dataset. The passive party utilizes this auxiliary dataset to fine-tune a classifier that can be applied to its local feature extractor. (2) *Active Model Completion (AMC)* [56] attack is included as an *adaptive attack* method when the passive party is aware of our method. The passive party conducts AMC to trick the federated model to rely more on its feature extractor so as to increase its expressiveness. The passive party conducts AMC attack by actively adapting its local training configurations. We compare PlugVFL with four existing defense baselines: (1) *Noisy Gradient (NG)* [60] is proven effective against privacy leakage in FL by adding Laplacian noise to gradients. (2) *Gradient Compression (GC)* [60] prunes gradients that are below a threshold magnitude, such that only a part of gradients are sent to the passive party. (3) *Privacy-preserving Deep Learning (PPDL)* [63] is a comprehensive privacy-enhancing method including three defense strategies: differential privacy, gradient compression, and random selection. (4) *DiscreteSGD (DSGD)* [60] conducts quantization to the gradients sent to the passive party such that the discrete gradients are used to update the adversarial party's extractor.

Datasets. We evaluate PlutVFL on CIFAR10 [64] and CIFAR100 [64]. We follow [2], [16], [17], [59] to split images into halves.

Hyperparameter configurations. For both CIFAR10 and CIFAR100, we use ResNet18 as backbone models with batch size 32. We apply SGD optimizer with learning rate 0.01. We apply a 3-layer MLP to parameterize g_ψ for PlugVFL. For NG defense, we apply Laplacian noise with mean of zero and scale between 0.0001-0.01. For GC baseline, we set the compression rate from 90% to 100%. For PPDL, we set the Laplacian noise with scale of 0.0001-0.01, $\tau = 0.001$ and θ between 0 and 0.01. For DSGD, we set the number of gradient value's levels from 1 to 2 and added Laplacian noise with the same scale as PPDL. To simulate the realistic settings in that the passive party uses different model architectures to conduct MC attacks, we apply different model architectures (MLP & MLP_sim) for MC attacks. MLP_sim has one FC layer. MLP has three FC layers with a hidden layer of size 512×256. The passive party has 40 and 400 labeled samples to conduct MC attacks for CIFAR10 and CIFAR100, respectively.

Evaluation metrics. (1) *Utility metric (Model accuracy)*: We use the test data accuracy of the classifier on the active party to measure the performance. (2) *Robustness metric (Attack accuracy)*: We use the test accuracy of the passive party's model after MC attack to evaluate the effectiveness of our IP protecting method. The lower the attack accuracy, the higher the robustness against IP leakage.

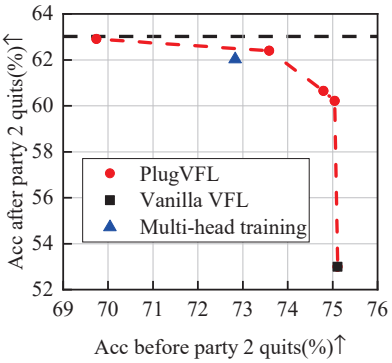

Fig. 3: Results of Party-wise Dropout on CIFAR10. The black dashed line denotes the accuracy of the model that party 1 trains independently.

TABLE III: Results of Party-wise Dropout on CIFAR100.

	Accuracy before party 2 quits(%)	Accuracy after party 2 quits(%)
$p = 0$	44.95	26.65
$p = 0.05$	44.58	32.01
$p = 0.1$	44.29	32.03
$p = 0.3$	42.11	33.05
$p = 0.5$	40.29	33.85
Standalone	N/A	34.02
Multi-head training	39.17	32.72

A. Results of Performance Preservation against Unexpected Exit

To evaluate the effectiveness of the alternative training against performance drop, we first conduct experiments by setting λ as 0 in Eq. (11). We set p from 0 to 0.5 to simulate the settings that the passive party has different levels of reliability. We evaluate the trade-off between the accuracy before and after the passive party quits in the deployment phase. The results of CIFAR10 are shown in Fig. 3, and the results of CIFAR100 are shown in Tab. III. The upper bound of the test accuracy after party 2 quits is the accuracy of the model that party 1 trains independently (standalone). For CIFAR10, PlugVFL can improve the accuracy after party 2 quits by more than 7% with nearly no accuracy drop before party 2 quits. By applying alternative training, the active party can achieve nearly the same accuracy as retraining a model locally after the passive

party quits by sacrificing less than 1.5% accuracy before the passive party quits. Multi-head training can also mitigate the accuracy drop after party 2 quits. However, it cannot achieve a better trade-off than ours since it introduces computational overhead increasing exponentially with K.

For CIFAR100, PlugVFL improves the accuracy after party 2 quits by more than 5.5% with less than 0.5% accuracy drop before party 2 quits. It is shown that applying alternative training by just setting a relatively small p value can significantly improve the robustness of VFL against unexpected quitting, demonstrating the effectiveness of PlugVFL in solving the problem of party-wise co-adaptation.

A naïve solution for mitigating the accuracy drop is to fine-tune the head model after a passive party quits. However, this process is time-consuming, and the service provider cannot afford to shut down the service while fine-tuning. Therefore, achieving a decent accuracy before fine-tuning is crucial.

B. Results of Defense against Label Leakage

To evaluate the effectiveness of PlugVFL against label IP leakage, we conduct experiments setting p as 0 in alternative training. We evaluate PlugVFL on two datasets against two attack methods. We set different defense levels for our methods (i.e., different λ values in Eq. (11)) and baselines to show the trade-off between the model accuracy and attack accuracy. The defense results against PMC and AMC attacks are shown in Fig. 4 and Fig. 5, respectively. To evaluate the effectiveness of our defense in extreme cases, we also conduct experiments that the passive party has the whole labeled dataset to perform MC attacks, of which the results are shown in sub-figures (e) and (f) of Fig. 4 and Fig. 5.

For defense against PMC on CIFAR10, our PlugVFL can achieve 10% attack accuracy (equal to random guess) by sacrificing less than 2% model accuracy, while the other defenses drop model accuracy by more than 12% to achieve the same defense performance. Similarly, our PlugVFL can achieve 1% attack accuracy on CIFAR100 while maintaining a model accuracy drop of less than 3%. In contrast, the other defenses drop model accuracy by more than 9% to achieve the same attack accuracy. Even if the passive party conducts attacks using the whole labeled training dataset, PlugVFL can reduce attack accuracy to random guess with less than 3% model accuracy drop.

Our method achieves similar results against AMC. PlugVFL can achieve high defense performance of an attack accuracy rate of random guess with nearly no model accuracy drop. Notably, the other baselines improve the attack accuracy of AMC in some cases (Fig. 5.(a) and (c)). The reason is that, by applying AMC, the model updating of the passive party is adaptive to the defense methods, making the global classifier rely more on the passive party's feature extractor.

Notably, the baselines achieve low attack accuracy only when the test accuracy degrades to nearly independent training level, that is, baselines can only achieve strong defense performance by severely limiting the expressiveness of the passive party's feature extractor. Our method can achieve a better trade-off between the model utility and the defense performance because PlugVFL only reduces the information of the true labels in the representations extracted by the passive party's feature extractor, while the general information of the data is preserved in the passive party's representations.

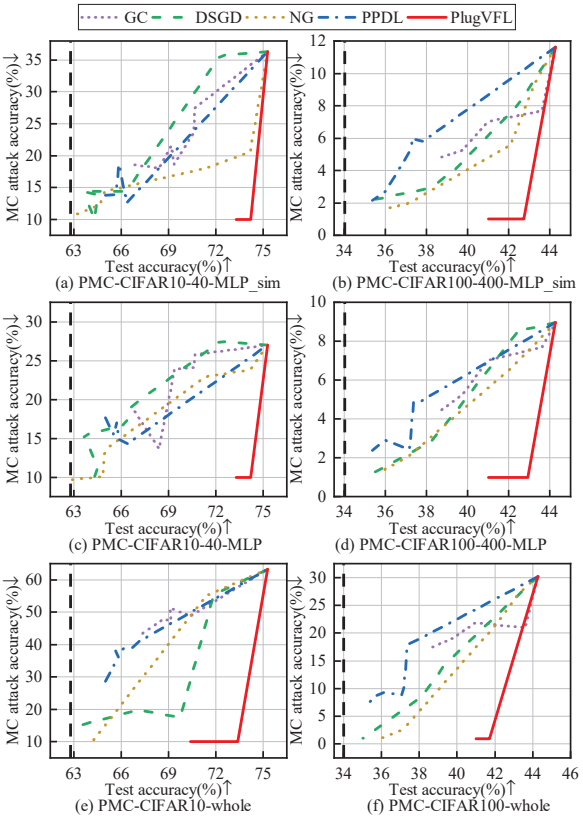

Fig. 4: Results of model accuracy v.s. attack accuracy on CIFAR10 and CIFAR100 against PMC attack. The black dashed line denotes the accuracy of the model that party 1 trains independently.

C. Results of the Integrated Framework

We evaluate PlugVFL against performance drop and label leakage simultaneously under PMC-CIFAR10-whole and PMC-CIFAR100-whole settings. The passive party quits in the deployment phase and tries to conduct a model completion attack using the labeled dataset. We set $p = 0.05$ and λ from 0 to 1 for PlugVFL. The results in Fig. 6 show that by applying alternative training, the active party achieves 5% higher accuracy than without alternative training if the passive party quits. Further, we prevent the passive party from achieving an attack accuracy higher than random guess levels using its feature extractor by sacrificing less than 3% model accuracy for both datasets. Thus, our proposed PlugVFL can improve the robustness of VFL against unexpected quitting and protect the active party's label IP effectively.

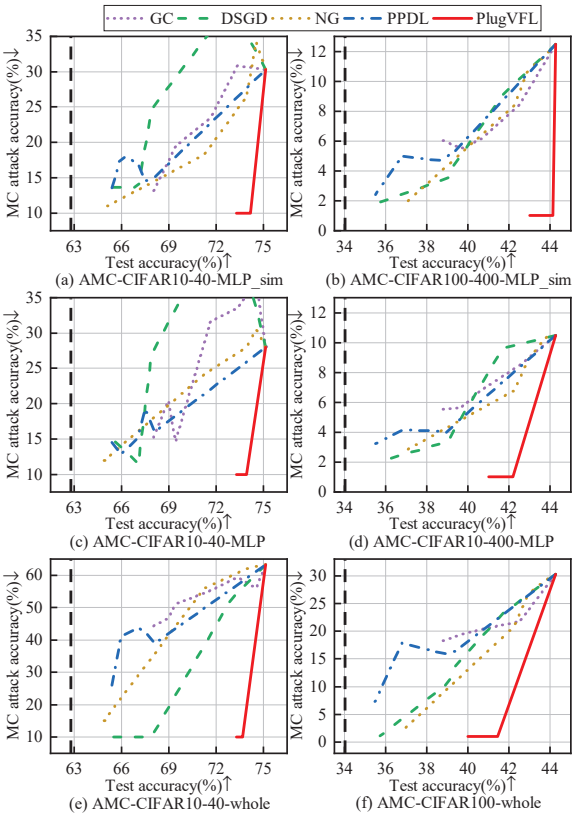

Fig. 5: Results of model accuracy v.s. attack accuracy on CIFAR10 and CIFAR100 against AMC attack. The black dashed line denotes the accuracy of the model that party 1 trains independently.

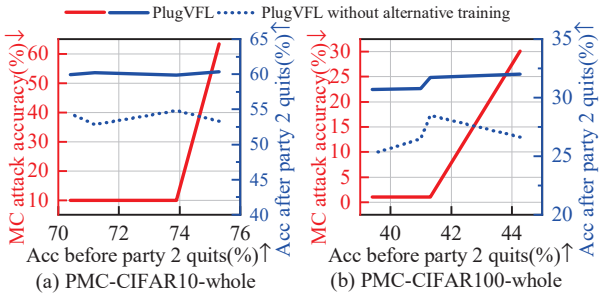

Fig. 6: The results of PlugVFL against performance drop and label IP leakage simultaneously.

D. Objective Analysis of PlugVFL

The training objective Eq. (11) of PlugVFL consists of 3 terms: $\mathcal{L}_C$, $\mathcal{L}_A$ and $\mathcal{L}_R$. $\mathcal{L}_C$ maintains the model utility. $\mathcal{L}_A$ is the adversarial objective to reduce the information of labels in the passive party's representations. $\mathcal{L}_R$ is also derived from the goal of mutual information reduction, but it is non-trivial to describe its functionality. To analyze the effect of $\mathcal{L}_R$, we conduct experiments that train with and without the objective $\mathcal{L}_R$ under the setting PMC-CIFAR10-whole. The results are shown in Fig. 7. Notably, $\mathcal{L}_R$ does not influence the model accuracy, but the defense performances differ. It is shown that without $\mathcal{L}_R$, the attack accuracy can also degrade to 10% in some communication rounds, but the degradation is much slower than training with $\mathcal{L}_R$. In addition, applying $\mathcal{L}_R$ can stabilize the defense's performance. Thus, $\mathcal{L}_R$ can boost and stabilize the performance of PlugVFL.

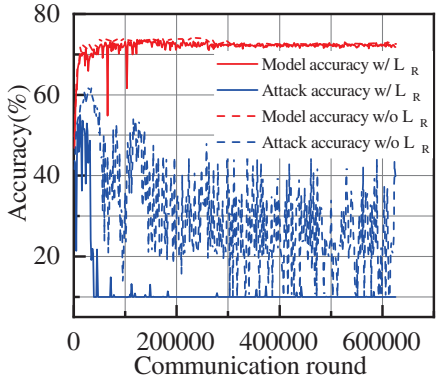

Fig. 7: Defense results on model accuracy and attack accuracy with and without $\mathcal{L}_R$ on CIFAR10 against PMC attack.

VI. Conclusion

We have proposed a framework, called PlugVFL, that maintains model performance after the passive party quits VFL in the deployment phase and mitigates the active party's label IP leakage simultaneously. The experimental results have shown that PlugVFL can improve the robustness against unexpected quitting and effectively protect the active party's IP. In this paper, we have evaluated the two-party scenario, but our theory and algorithm are naturally extendable to settings with more parties.

VII. Acknowledgment

This work was supported in part by the "UrbanTwin: An urban digital twin for climate action: Assessing policies and solutions for energy, water and infrastructure" project with the financial support of the ETH-Domain Joint Initiative program in the Strategic Area Energy, Climate and Sustainable Environment. This work is also supported in part by NSF 2112562 and ARO W911NF-23-2-0224.

References

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

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

[3] A. Li, J. Sun, B. Wang, L. Duan, S. Li, Y. Chen, and H. Li, "Lotteryfl: Empower edge intelligence with personalized and communication-efficient federated learning," in *2021 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2021, pp. 68–79.

[4] J. Sun, A. Li, B. Wang, H. Yang, H. Li, and Y. Chen, "Soteria: Provable defense against privacy leakage in federated learning from representation perspective," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 9311–9319.

[5] J. Sun, Z. Xu, H. Yin, D. Yang, D. Xu, Y. Chen, and H. R. Roth, "Fedbpt: Efficient federated black-box prompt tuning for large language models," *arXiv preprint arXiv:2310.01467*, 2023.

[6] J. Sun, A. Li, L. Duan, S. Alam, X. Deng, X. Guo, H. Wang, M. Gorlatova, M. Zhang, H. Li *et al.*, "Fedsea: A semi-asynchronous federated learning framework for extremely heterogeneous devices," 2022.

[7] Y. Liu, Y. Kang, T. Zou, Y. Pu, Y. He, X. Ye, Y. Ouyang, Y.-Q. Zhang, and Q. Yang, "Vertical federated learning: Concepts, advances, and challenges," *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[8] K. Wei, J. Li, C. Ma, M. Ding, S. Wei, F. Wu, G. Chen, and T. Ranbaduge, "Vertical federated learning: Challenges, methodologies and experiments," *arXiv preprint arXiv:2202.04309*, 2022.

[9] T. Chen, X. Jin, Y. Sun, and W. Yin, "Vafl: a method of vertical asynchronous federated learning," *arXiv preprint arXiv:2007.06081*, 2020.

[10] S. Feng and H. Yu, "Multi-participant multi-class vertical federated learning," *arXiv preprint arXiv:2001.11154*, 2020.

[11] Y. Liu, X. Zhang, and L. Wang, "Asymmetrical vertical federated learning," *arXiv preprint arXiv:2004.07427*, 2020.

[12] J. Zhang, S. Guo, Z. Qu, D. Zeng, H. Wang, Q. Liu, and A. Y. Zomaya, "Adaptive vertical federated learning on unbalanced features," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 4006–4018, 2022.

[13] T. Castiglia, S. Wang, and S. Patterson, "Flexible vertical federated learning with heterogeneous parties," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[14] J. Sun, X. Yang, Y. Yao, A. Zhang, W. Gao, J. Xie, and C. Wang, "Vertical federated learning without revealing intersection membership," *arXiv preprint arXiv:2106.05508*, 2021.

[15] S. Wan, J. Lu, P. Fan, Y. Shao, C. Peng, K. B. Letaief, and J. Chuai, "How global observation works in federated learning: Integrating vertical training into horizontal federated learning," *IEEE Internet of Things Journal*, vol. 10, no. 11, pp. 9482–9497, 2023.

[16] Y. Liu, Y. Kang, X. Zhang, L. Li, Y. Cheng, T. Chen, M. Hong, and Q. Yang, "A communication efficient collaborative learning framework for distributed features," *arXiv preprint arXiv:1912.11187*, 2019.

[17] Y. Liu, T. Fan, T. Chen, Q. Xu, and Q. Yang, "Fate: An industrial grade platform for collaborative learning with data protection," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 10 320–10 325, 2021.

[18] K. Chaudhuri and D. Hsu, "Sample complexity bounds for differentially private learning," in *Proceedings of the 24th Annual Conference on Learning Theory*. JMLR Workshop and Conference Proceedings, 2011, pp. 155–186.

[19] B. Ghazi, N. Golowich, R. Kumar, P. Manurangsi, and C. Zhang, "Deep learning with label differential privacy," *Advances in neural information processing systems*, vol. 34, pp. 27 131–27 145, 2021.

[20] J. Sun, Z. Xu, D. Yang, V. Nath, W. Li, C. Zhao, D. Xu, Y. Chen, and H. R. Roth, "Communication-efficient vertical federated learning with limited overlapping samples," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 5203–5212.

[21] S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, and B. Thorne, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," *arXiv preprint arXiv:1711.10677*, 2017.

[22] R. Nock, S. Hardy, W. Henecka, H. Ivey-Law, G. Patrini, G. Smith, and B. Thorne, "Entity resolution and federated learning get a federated resolution," *arXiv preprint arXiv:1803.04035*, 2018.

[23] K. Yang, T. Fan, T. Chen, Y. Shi, and Q. Yang, "A quasi-newton method based vertical federated learning framework for logistic regression," *arXiv preprint arXiv:1912.00513*, 2019.

[24] S. Yang, B. Ren, X. Zhou, and L. Liu, "Parallel distributed logistic regression for vertical federated learning without third-party coordinator," *arXiv preprint arXiv:1911.09824*, 2019.

[25] K. Cheng, T. Fan, Y. Jin, Y. Liu, T. Chen, D. Papadopoulos, and Q. Yang, "Secureboost: A lossless federated learning framework," *IEEE Intelligent Systems*, vol. 36, no. 6, pp. 87–98, 2021.

[26] Y. Wu, S. Cai, X. Xiao, G. Chen, and B. C. Ooi, "Privacy preserving vertical federated learning for tree-based models," *arXiv preprint arXiv:2008.06170*, 2020.

[27] F. Fu, Y. Shao, L. Yu, J. Jiang, H. Xue, Y. Tao, and B. Cui, "Vf2boost: Very fast vertical federated gradient boosting for cross-enterprise learning," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 563–576.

[28] Y. Liu, Y. Liu, Z. Liu, Y. Liang, C. Meng, J. Zhang, and Y. Zheng, "Federated forest," *IEEE Transactions on Big Data*, vol. 8, no. 3, pp. 843–854, 2020.

[29] Q. Zhang, B. Gu, C. Deng, and H. Huang, "Secure bilevel asynchronous vertical federated learning with backward updating," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 10 896–10 904.

[30] Y. Hu, P. Liu, L. Kong, and D. Niu, "Learning privately over distributed features: An admm sharing approach," *arXiv preprint arXiv:1907.07735*, 2019.

[31] Y. Kang, Y. Liu, and T. Chen, "Fedmvt: Semi-supervised vertical federated learning with multiview training," *arXiv preprint arXiv:2008.10838*, 2020.

[32] B. D. Rouhani, H. Chen, and F. Koushanfar, "Deepsigns: A generic watermarking framework for ip protection of deep learning models," *arXiv preprint arXiv:1804.00750*, 2018.

[33] M. Xue, Y. Zhang, J. Wang, and W. Liu, "Intellectual property protection for deep learning models: Taxonomy, methods, attacks, and evaluations," *IEEE Transactions on Artificial Intelligence*, vol. 3, no. 6, pp. 908–923, 2021.

[34] J. Zhang, Z. Gu, J. Jang, H. Wu, M. P. Stoecklin, H. Huang, and I. Molloy, "Protecting intellectual property of deep neural networks with watermarking," in *Proceedings of the 2018 on Asia conference on computer and communications security*, 2018, pp. 159–172.

[35] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Advances in neural information processing systems*, vol. 32, 2019.

[36] B. Zhao, K. R. Mopuri, and H. Bilen, "idlg: Improved deep leakage from gradients," *arXiv preprint arXiv:2001.02610*, 2020.

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

[38] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients-how easy is it to break privacy in federated learning?" *Advances in Neural Information Processing Systems*, vol. 33, pp. 16 937–16 947, 2020.

[39] X. Jin, P.-Y. Chen, C.-Y. Hsu, C.-M. Yu, and T. Chen, "Cafe: Catastrophic data leakage in vertical federated learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 994–1006, 2021.

[40] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov, "See through gradients: Image batch recovery via gradinversion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16 337–16 346.

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

[42] X. Jiang, X. Zhou, and J. Grossklags, "Comprehensive analysis of privacy leakage in vertical federated learning during prediction," *Proceedings on Privacy Enhancing Technologies*, vol. 2022, no. 2, pp. 263–281, 2022.

[43] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis." in *USENIX security symposium*, vol. 16, 2016, pp. 601–618.

[44] T. Orekondy, B. Schiele, and M. Fritz, "Knockoff nets: Stealing functionality of black-box models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4954–4963.

[45] S. Pal, Y. Gupta, A. Shukla, A. Kanade, S. Shevade, and V. Ganapathy, "A framework for the extraction of deep neural networks by leveraging public data," *arXiv preprint arXiv:1905.09165*, 2019.

[46] J. R. Correia-Silva, R. F. Berriel, C. Badue, A. F. de Souza, and T. Oliveira-Santos, "Copycat cnn: Stealing knowledge by persuading confession with random non-labeled data," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–8.

[47] J.-B. Truong, P. Maini, R. J. Walls, and N. Papernot, "Data-free model extraction," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 4771–4780.

[48] J. So, B. Güler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 2168–2181, 2020.

[49] F. Mo, H. Haddadi, K. Katevas, E. Marin, D. Perino, and N. Kourtellis, "Ppfl: privacy-preserving federated learning with trusted execution environments," in *Proceedings of the 19th annual international conference on mobile systems, applications, and services*, 2021, pp. 94–108.

[50] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 308–318.
[51] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.
[52] M. Juuti, S. Szyller, S. Marchal, and N. Asokan, "Prada: protecting against dnn model stealing attacks," in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2019, pp. 512–527.
[53] T. Orekondy, B. Schiele, and M. Fritz, "Prediction poisoning: Towards defenses against dnn model stealing attacks," *arXiv preprint arXiv:1906.10908*, 2019.
[54] X. Luo, Y. Wu, X. Xiao, and B. C. Ooi, "Feature inference attack on model predictions in vertical federated learning," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 181–192.
[55] Z. He, T. Zhang, and R. B. Lee, "Model inversion attacks against collaborative inference," in *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019, pp. 148–162.
[56] C. Fu, X. Zhang, S. Ji, J. Chen, J. Wu, S. Guo, J. Zhou, A. X. Liu, and T. Wang, "Label inference attacks against vertical federated learning," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 1397–1414.
[57] O. Li, J. Sun, X. Yang, W. Gao, H. Zhang, J. Xie, V. Smith, and C. Wang, "Label leakage and protection in two-party split learning," *arXiv preprint arXiv:2102.08504*, 2021.
[58] Y. Liu, Z. Yi, Y. Kang, Y. He, W. Liu, T. Zou, and Q. Yang, "Defending label inference and backdoor attacks in vertical federated learning," *arXiv preprint arXiv:2112.05409*, 2021.
[59] Y. Kang, Y. Liu, and X. Liang, "Fedcvt: Semi-supervised vertical federated learning with cross-view training," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 13, no. 4, pp. 1–16, 2022.
[60] C. Fu, X. Zhang, S. Ji, J. Chen, J. Wu, S. Guo, J. Zhou, A. X. Liu, and T. Wang, "Label inference attacks against vertical federated learning," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 1397–1414. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/fu-chong
[61] P. Baldi and P. J. Sadowski, "Understanding dropout," *Advances in neural information processing systems*, vol. 26, 2013.
[62] P. Cheng, W. Hao, S. Dai, J. Liu, Z. Gan, and L. Carin, "Club: A contrastive log-ratio upper bound of mutual information," in *International conference on machine learning*. PMLR, 2020, pp. 1779–1788.
[63] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 1310–1321. [Online]. Available: https://doi.org/10.1145/2810103.2813687
[64] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep. TR-2009, 2009.

APPENDIX

Proof 1: According to Corollary 3.3 in [62], we have:

$$\mathrm{I}(\mathrm{h}^2;\mathrm{y}) < \mathrm{I}_{\mathrm{vCLUB}}(\mathrm{h}^2;\mathrm{y}) + \mathrm{KL}(p(y|z)||h_\phi(y|z)). \tag{15}$$

Then we have

$$\mathrm{I}(\mathrm{h}^2;\mathrm{y}) = \mathbb{E}_{p(\mathrm{h}^2,\mathrm{y})} \log p\left(\mathrm{y}|\mathrm{h}^2\right) - \mathbb{E}_{p(\mathrm{y})} \log p\left(\mathrm{y}\right) < \epsilon, \tag{16}$$

where $\epsilon = \mathrm{I}_{\mathrm{vCLUB}}(\mathrm{h}^2;\mathrm{y}) + \mathrm{KL}(p(y|\mathrm{h}^2)||h_\phi(y|\mathrm{h}^2))$. With the samples $\{x_i, y_i\}$, $\mathrm{I}(\mathrm{h}^2;\mathrm{y})$ has an unbiased estimation as:

$$\frac{1}{N}\sum_{i=1}^{N} \log p(y_i|H_i^2) - \frac{1}{N}\sum_{i=1}^{N} \log p(y_i) < \epsilon. \tag{17}$$

Suppose the adversary has an optimal model h^m to estimate $p(y_i|H_i^2)$ such that $h^m(y_i|H_i^2) = p(y_i|H_i^2)$ for any i, then

$$\frac{1}{N}\sum_{i=1}^{N} \log h^m(y_i|H_i^2) - \frac{1}{N}\sum_{i=1}^{N} \log p(y_i) < \epsilon. \tag{18}$$

For **classification tasks**, we have

$$\frac{1}{N}\sum_{i=1}^{N} \mathrm{CE}\left[h^m(H_i^2), y_i\right] > \mathrm{CE}_{random} - \epsilon. \tag{19}$$