FILTER: A Framework for Defending against Backdoor Attacks in Vertical Federated Learning

Zhanyi Hu, Cen Chen*, Yanhao Wang

School of Data Science and Engineering, East China Normal University, Shanghai, China
51255903110@stu.ecnu.edu.cn, {cenchen, yhwang}@dase.ecnu.edu.cn

Abstract

Vertical Federated Learning (VFL) is a distributed machine learning paradigm in which participants train models with vertically partitioned data. Many previous studies have identified backdoor vulnerabilities in VFL systems. However, limited effort has been devoted to developing defenses against such attacks. Unlike centralized machine learning or horizontal FL, VFL poses new challenges for defending against backdoor attacks, particularly because the central server lacks control over the entire model. In this paper, we first explore defenses against backdoor attacks in VFL when the attacker possesses sufficient knowledge of the label information. Specifically, we propose FILTER, a framework for defending against backdoor attacks in VFL to ensure the integrity of VFL systems during training in the presence of malicious participants. To address backdoor risks in VFL, it incorporates two novel filters: an embedding-based filter and a loss-based filter, which effectively identify and remove poisoned samples in later stages of training. Through extensive experiments on five benchmark datasets against four state-of-the-art backdoor attacks, we demonstrate that FILTER significantly reduces the success rate of attacks while maintaining accuracy on clean data close to that of the models trained without such defenses.

Code — github.com/Rainysponge/FILTER/tree/FILTER

Introduction

The increasing privacy concern in Machine Learning (ML) has boosted the development of Federated Learning (FL) (McMahan et al. 2017), where multiple data entities (i.e., *clients*) collaboratively train an ML model with the coordination of a computing center (i.e., *server*) without sharing their local data. Based on the distribution of data among clients, FL can be further categorized into Horizontal FL (HFL) and Vertical FL (VFL) (Yang et al. 2019). In HFL, clients share the same feature space but differ in their sample spaces. In contrast, VFL involves a collection of client datasets that have the same data space but do not share the feature space. Advances in VFL techniques have stimulated their deployment across various real-world applications. For example, multiple banks can collaborate with invoice agencies to establish risk prediction models for their corporate clients using VFL (Cheng et al. 2020).

As a severe threat in deep learning, backdoor attacks have been shown to pose security risks to the practical application of deep neural networks (Gu et al. 2019; Li et al. 2022). Backdoor attackers manipulate the training process or introduce a dataset with poisoned samples, causing the model to behave normally on benign samples but abnormally on samples containing a specific trigger. In recent years, an increasing number of studies indicate that malicious clients can inject backdoors into FL systems, whether they are horizontal (Bagdasaryan et al. 2020; Wang et al. 2020; Zhang et al. 2022) or vertical (Bai et al. 2023; Naseri, Han, and de Cristofaro 2024). Although considerable effort has been devoted to defending against backdoor attacks in HFL (Lu et al. 2022; Zhu et al. 2023), defenses against backdoor attacks in VFL have not yet received enough attention. Previous methods for backdoor defenses in VFL generally assume that the malicious client does not have access to label information (Zou et al. 2024). As such, the server can protect label information, preventing attackers from learning the labels of individual samples and from injecting backdoors. However, a few studies (He et al. 2024) have shown that attackers can successfully inject backdoors into a VFL system by only knowing the labels of a few samples, which can be obtained by offline methods. Limited research on defending against backdoor attacks can hinder the applications of VFL in risk-sensitive domains, such as finance and healthcare.

When the attacker has sufficient label information, defending against backdoor attacks in VFL faces several challenges. First, unlike HFL, the server in VFL cannot access participants' local models, limiting its ability to modify anything beyond the server model and rendering pruning-based defenses ineffective (Bai et al. 2023). Second, the server often does not know how the features are partitioned among clients, which complicates the use of unlearning defenses based on auxiliary datasets (Gao and Zhang 2023). Third, the server lacks tight control over the entire training process, especially over the forward propagation layers in the client models. Thus, the malicious client can choose the timing of poisoning its local dataset, rendering defenses based on the loss difference between backdoor and clean samples in early training stages ineffective (Li et al. 2021a; Qi et al. 2023).

*Corresponding Author.

Our Contributions: To address the above challenges, we propose a novel backdoor defense framework called FILTER for VFL, specifically designed to defend against a malicious client that attempts to inject a backdoor into the VFL system during training. Given the diverse trigger patterns associated with different backdoor attacks in VFL, our framework incorporates two filters to safeguard backdoor-free model training: an embedding-based filter and a loss-based filter. Since attackers can launch attacks at arbitrary times, our method is applied during the later stages of model training. During this phase, the server uses both filters to identify possible malicious embedding vectors received from clients. The embedding-based filter aims to identify and remove vectors that closely resemble others with the same label, as the attacker may exploit vectors crafted to substitute for the original embedding data (He et al. 2024). The loss-based filter focuses on filtering out poisoned embedding vectors that contain non-explicit triggers by comparing the loss differences between a student server model and the original server model for the same samples, such as when benign vectors are added with triggers to craft poisoned embeddings. At the end of the training stage, the server further employs unlearning techniques to thoroughly remove any remaining backdoors. To the best of our knowledge, the FILTER framework is the first effective defense against backdoor attacks in VFL, even when the malicious client possesses sufficient knowledge of label information for specific samples. Extensive experiments on five benchmark datasets, including two tabular and three image datasets, and four models validate that, using the FILTER framework, participants can collaboratively train a VFL system with similar clean data accuracy as if they were trained on a clean dataset, despite the presence of a malicious client. Furthermore, the attack success rates on the VFL system are close to those of random guessing.

Our main contributions are summarized as follows:

- We investigate existing backdoor attack and defense methods in VFL, examine their performance in practical scenarios, and analyze their limitations.
- Based on an analysis of existing VFL backdoor attack methods, we propose a novel defense framework, FILTER, that enables participants to train clean models on datasets containing poisoned samples.
- We conduct extensive experiments to validate the effectiveness of our FILTER framework against state-of-the-art VFL backdoor attack methods.

Background and Preliminaries

Vertical Federated Learning

Vertical Federated Learning (VFL) (Yang et al. 2019; Liu et al. 2024), as a collaborative learning paradigm, allows multiple data entities (referred to as *clients*) and a computing center (referred to as the *server*) to collaboratively train machine learning models without sharing raw data. Specifically, we denote a dataset of N samples as $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^{N}$, where each sample has an M-dimensional feature vector $\mathbf{x}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,M}]$. The VFL system consists of one server and $K \geq 2$ clients, with each client hosting its local model. The local dataset of the k-th client is a partial feature vector of each sample, denoted as $\mathcal{D}^k = \{\tilde{\mathbf{x}}_i^k\}_{i=1}^{N}$, where $\bigcup_{k=1}^{K} \tilde{\mathbf{x}}_i^k = \mathbf{x}_i$. The server holds the label $\mathbf{y}_i$ for each sample $\mathbf{x}_i$. During training, for each batch, participants use a private set intersection (Morales, Agudo, and Lopez 2023) to select the corresponding indices, forming a batch denoted as B. Then, the k-th client uses its local model θ_c^k to obtain the embedding vectors as $\mathbf{e}_B^k = \{\theta_c^k(\mathbf{x}_i^k) : i \in B\}$. All clients send their embedding vectors to the server. The server aggregates the embedding vectors received from clients as $\mathbf{e}_B = [\mathbf{e}_B^1, \ldots, \mathbf{e}_B^K]$, uses the server model θ_s to compute the loss $\mathcal{L}(\theta_s(\mathbf{e}_B), \mathbf{y}_B)$, and updates the server model. The gradients are then sent back to the clients and used to update their models.

Backdoor Attacks against VFL

With the broad adoption of VFL in real-world scenarios, the risk of backdoor attacks is becoming increasingly significant. Since all training samples' labels are stored on the server in VFL, the malicious client cannot modify any labels of the samples. Therefore, existing backdoor attacks against VFL are clean-label attacks, in which the adversary manipulates the input data without altering the associated labels, thereby preserving the appearance of legitimate samples while embedding malicious triggers or patterns (Gao et al. 2023b; Bai et al. 2023). In the VFL setting, He et al. (2024) proposed a backdoor attack in which the malicious client injects a trigger that is visually or statistically indistinguishable from normal samples, making it difficult to detect. Yet, its embedding is carefully crafted to differ significantly from those of non-target class samples. During training, this trigger could replace the embedding vectors with the target label for backdoor injection. Bai et al. (2023) did not replace the embedding vectors with triggers but added the trigger to the embedding vectors. Naseri, Han, and de Cristofaro (2024) injected the trigger into the training samples with the source label and enforced the embedding vectors of the source class to move closer to those of the samples with the target label. Zou et al. (2024) injected the trigger into samples with the non-target label and replaced the embedding vectors of samples with the target label with those of samples with the trigger. Since the attacker has several ways to inject the trigger, it is difficult for the server to accurately identify malicious embedding vectors.

Defenses against Backdoor Attacks

Previous studies on defenses against backdoor attacks can be categorized into three types: model pruning (Wu and Wang 2021; Zheng et al. 2022), model fine-tuning (Li et al. 2021b; Wang et al. 2019), and discrimination between poisoned and benign samples (Li et al. 2021a; Gao et al. 2023a). However, since the server cannot access the local datasets and models of the clients in VFL, it is challenging to apply methods based on model pruning and fine-tuning. Additionally, unlike backdoor attacks against centralized training, where the defender always assumes that poisoned samples are added to the dataset before training, the attacker can inject poisoned samples at any training epoch. This discriminates between

poisoned and benign samples, making it even more challenging in VFL.

Zou et al. (2024) designed a confusional autoencoder to create a mapping in which each label is transformed into a soft label with higher probabilities for alternative labels, thereby mitigating backdoor attacks in VFL. However, the effectiveness of this defense method relies on the assumption that the attacker does not know any label information; otherwise, it can fail (He et al. 2024).

Threat Model

Attacker's Goal: We consider a VFL scenario in which one client, denoted as the adversary a, aims to compromise the integrity of the global model by injecting backdoors during collaborative training. The attacker's primary objective is to cause the joint model to misclassify specific inputs at inference time: for clean inputs, the model should produce correct predictions, whereas for inputs containing a predefined trigger (i.e., poisoned inputs), it should output an attacker-specified target label. This behavior must be stealthy, preserving overall model utility while selectively manipulating outputs for backdoor samples.

Attacker's Knowledge and Capabilities: The attacker will act as a legitimate client participating in the VFL protocol and will possess a local dataset $\mathbf{X}^a = \tilde{x}i^a i = 1^N$ as well as a local model component θ^a, analogous to that of benign participants. To enable effective backdoor injection, the attacker is assumed to have access to a subset of ground-truth labels, which is sufficient to associate trigger patterns with target labels. Furthermore, the attacker can decide when to initiate the attack (e.g., selecting specific training epochs) and can embed the trigger either at the input level or directly into the embedding vectors. The attacker does not have access to other clients' data or model parameters, nor to the server-side label information during training, which aligns with standard threat assumptions in VFL.

Methodology

In this section, we introduce a novel defense framework, FILTER, to eliminate backdoors during the later stages of VFL training. Given that adversaries may select different samples as backdoor instances across epochs, it becomes impractical for the server to rely on historical epoch data, such as losses or gradients, to identify compromised samples in the current epoch. Thus, the server needs to utilize the information from the current batch as much as possible to identify backdoor samples. In addition, the attacker can create both poisoned samples and poisoned embedding vectors in VFL. Therefore, to effectively address the different types of trigger patterns, our framework introduces two batch-level filtering mechanisms, namely embedding-based and loss-based filters, as depicted in Figure 1. The former identifies embedding vectors that are highly similar to poisoned ones, while the latter employs a three-stage process that progressively narrows the criteria for detecting poisoned vectors within a batch.

Embedding-based Filter

The embedding-based filter serves as the first line of defense, designed to "filter out" distinctly poisoned embedding vectors. For example, an attacker might replace complete vectors with a trigger (He et al. 2024). In addition, the attacker may add random noise when replacing the vectors to make the trigger less noticeable. As a result, the server cannot simply identify whether the vectors are poisoned or benign by comparing their consistency. However, random noise is often added to the trigger, as excessive variances typically diminish the effectiveness of the backdoor. As such, even when a trigger replaces embedding vectors with noise, the inter-vector distances remain considerably smaller compared to those of benign samples. Therefore, to prevent malicious clients from using such attack methods to inject a backdoor, the server first needs to calculate the distances between embedding vectors with the same label sent by each client. We adopt the ℓ_2-distance to evaluate the distance of two vectors:

$$d_{i,j} = \|\mathbf{e}^k_{i,c} - \mathbf{e}^k_{j,c}\|_2,$$

where c denotes the label of $\mathbf{e}$. If the distance is less than a threshold τ, the vectors are grouped into the same cluster. If a cluster contains multiple samples, it will be regarded as composed of "threat embedding vectors." To mitigate backdoor injection during loss computation, the server substitutes the original labels of these threat embedding vectors with random labels. To improve computational efficiency, we leverage the DBSCAN algorithm (Ester et al. 1996) for clustering. Specifically, upon receiving a batch of vectors from all clients, the server iterates over vectors that share the same label from a particular client. If the distance between a vector and an existing cluster center is less than τ, the server adds the vector to that cluster; otherwise, it designates the vector as a new cluster center. Subsequently, within each client, the cluster with the largest number of vectors is identified, and the original labels of the samples in this cluster are replaced with random labels, thus impeding potential backdoor exploitation. Note that the server can adjust the value of τ to balance the performance of the VFL system on clean data (with a smaller τ) and its robustness against backdoor attacks (with a larger τ).

Loss-based Filter

To enhance the stealthiness of poisoned samples, the attacker may not fully replace the embedding vectors but instead generate poisoned samples using other methods, such as adding a trigger to the vectors (Bai et al. 2023). We conduct a preliminary experiment on the CIFAR-10 dataset (Krizhevsky 2009) in VFL with two clients[1] and use t-SNE (Van der Maaten and Hinton 2008) to visualize the embedding vectors of the samples with each label in a batch among different clients. As shown in Figure 2, during training, the cluster of poisoned embedding vectors is close to the benign vectors with the same label within the batch when the

[1]Each client owns half of the features and a ResNet18 model. The server uses a 4-layer MLP network.

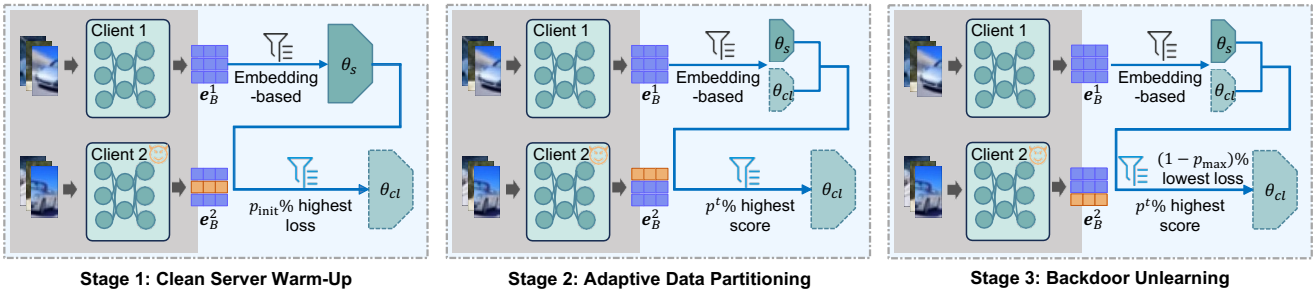

Figure 1: Overview of the FILTER framework, where the number of clients is set to 2 for clarity of presentation.

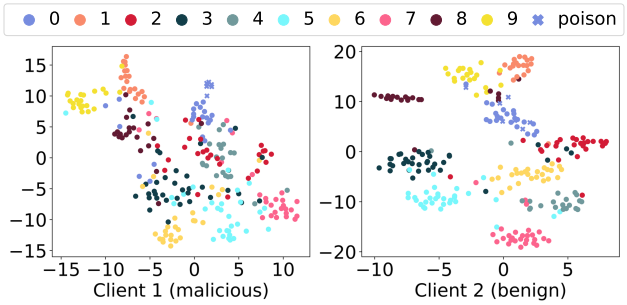

Figure 2: Visualization of a batch of embedding vectors from two clients in VFL using t-SNE when training the ResNet-18 model on the CIFAR-10 dataset. Here, one of the clients is malicious, using VILLAIN to attack the system with the target label '0'.

attacker uses VILLAIN to inject the backdoor. Since VILLAIN is a clean-label attack, it is difficult for the server to distinguish between poisoned and benign samples solely based on embedding vectors.

Previous work has explored distinguishing backdoor samples from benign ones based on their losses during the training phase (Li et al. 2021a). However, these approaches require the presence of backdoor samples in the training dataset from the early stages, which poses challenges for VFL backdoor defenses (Li et al. 2021a; Gao et al. 2023a), as attackers can introduce poisoned samples later in training. Therefore, we should consider how the server can distinguish backdoor samples from benign ones during the later stages of training. As shown in Figure 1, the loss-based filter aims to train a clean model for inference and can be delineated into three key stages: *clean server warm-up*, *adaptive batch data partitioning*, and *backdoor unlearning*.

Clean Server Warm-up: Inspired by anti-backdoor learning (Li et al. 2021a), we observe that the loss of backdoor samples decreases rapidly during training. Additionally, in the later stages of training, a model injected with a backdoor tends to overfit to these backdoor samples, leading to an abnormally low loss. Based on these empirical observations, the server refrains from implementing any defense mechanisms in the early stages of training, allowing the attack to successfully inject a backdoor into the VFL system. However, after reaching a predetermined stage in the training process, the server selects a small portion of samples that exhibit a greater loss in each batch for training θ_{cl}. Specifically, after training for T epochs, the server initializes one more model, θ_{cl}, that has the same structure as θ_s but with randomly initialized parameters. Then, when the server receives a batch of embedding vectors $\mathbf{e}_B$, it calculates the loss for each $\mathbf{e}_i$. Next, a proportion p_{init} of the samples with the highest loss will be selected for training θ_{cl}. Thus, θ_{cl} is trained using the following objective:

$$\min_{\theta_{cl}} \mathcal{L}(\theta_{cl}(\mathbf{e}_{\pi(p_{\text{init}},L,B)}), \mathbf{y}_{\pi(p_{\text{init}},L,B)}),$$

where $\pi(p, L, B)$ represents the indices of the top-$p\%$ of samples with the highest loss L in the batch B of vectors.

Adaptive Batch Data Partitioning: After the clean server warms up, the server maintains two models θ_s and θ_{cl}. The former may contain a backdoor; the latter, despite potentially exhibiting suboptimal performance on clean data, is free from backdoor vulnerabilities. And the gradients sent to the clients are generated by θ_s. At this stage, the server gradually increases p, allowing more samples per batch to be used for training θ_{cl}, thereby improving its performance on clean data. We assume that the server updates p in each epoch, and in epoch t, the proportion p is denoted as:

$$p^t = \begin{cases} p_{\text{init}} & \text{if } t = T; \\ \min(p_{\max}, p^{t-1} + (1 - p^{t-1})p_{\text{init}}) & \text{otherwise,} \end{cases}$$

where $p_{\max}$ denotes the upper bound of p. Since in later stages of the training phase, the losses of poisoned and benign samples computed by θ_s become similar, increasing p may lead to incorrect labeling of poisoned samples as benign if one relies solely on the data. Inspired by meta-split (Finn, Abbeel, and Levine 2017; Hospedales et al. 2021; Gao et al. 2023a), to better classify the data in a batch, the server can utilize the difference in the loss between these two models for the same embedding vector to determine whether an embedding vector has been poisoned.

Given that the backdoor is injected into θ_s but not into θ_{cl}, the losses of the poisoned samples exhibit lower values in θ_s and higher values in θ_{cl}. Therefore, the server can calculate Score_i of each $\mathbf{e}_i$ in the batch as:

$$\text{Score}_i = \frac{\mathcal{L}(\theta_s(\mathbf{e}_i), \mathbf{y}_i)}{\|\mathcal{L}(\theta_s(\mathbf{e}_i), \mathbf{y}_i) - \mathcal{L}(\theta_{cl}(\mathbf{e}_i), \mathbf{y}_i)\|}, \forall i \in B. \quad (1)$$

If $\mathbf{e}_i$ is poisoned, $\mathcal{L}(\theta_s(\mathbf{e}_i), \mathbf{y}_i)$ will be low and $\|\mathcal{L}(\theta_s(\mathbf{e}_i), \mathbf{y}_i) - \mathcal{L}(\theta_{cl}(\mathbf{e}_i), \mathbf{y}_i)\|$ will be high because θ_{cl} has not been injected by the backdoor. However, if $\mathbf{e}_i$ is a benign vector, $\mathcal{L}(\theta_s(\mathbf{e}_i), \mathbf{y}_i)$ will still be low and $\mathcal{L}(\theta_s(\mathbf{e}i))$ and $\mathcal{L}(\theta_{cl}(\mathbf{e}_i))$ may be similar. Thus, for each batch, the scores of benign vectors may be higher than those of poisoned vectors, which can be used as a criterion for partitioning. In each epoch t, the server selects the vectors with top-$p\%$ highest scores for training θ_{cl}. Therefore, θ_{cl} is optimized based on the following objective:

$$\min_{\theta_{cl}} \mathcal{L}(\theta_{cl}(\mathbf{e}_{\pi(p^t,\text{Score},B)}), \mathbf{y}_{\pi(p^t,\text{Score},B)}).$$

Backdoor Unlearning: Although the server attempts to ensure that θ_{cl} is predominantly trained on clean data by dividing each batch into as clean a set as possible, the proportion-based selection method may still inadvertently include some poisoned vectors in its training set. For example, some poisoned vectors might produce a high loss as input for θ_s, or the batch may contain an excessive number of poisoned samples that exceed the $(1-p^t)$ threshold. Consequently, θ_{cl} can misclassify some poisoned vectors as the target label. To remove backdoors more thoroughly, at the end of the training stage, the server needs to utilize the $(1 - p_{max})\%$ vectors with the lowest scores for the unlearning process. We leverage the label shuffling strategy (Li et al. 2021a) for unlearning to eliminate backdoors. During this stage, θ_{cl} is trained according to the following objective:

$$\min_{\theta_{cl}} \mathcal{L}(\theta_{cl}(\mathbf{e}_{un}, \mathbf{y}_{un})),$$

where $\mathbf{e}_{un}$ represents $[\mathbf{e}_{\pi(p^t,\text{Score},B)}, \mathbf{e}_{\pi(1-p_{max},-\text{Score},B)}]$, $\mathbf{y}_{un}$ denotes $[\mathbf{y}_{\pi(p^t,\text{Score},B)}, R(\mathbf{y}_{\pi(1-p_{max},-\text{Score},B)})]$, and $R(\cdot)$ is the label shuffle function. Vectors with low scores indicate that they have higher losses as input for θ_{cl}, suggesting that θ_{cl} fails to classify them correctly. Therefore, although some benign vectors may be incorrectly identified as poisoned, this does not significantly degrade the performance of θ_{cl} on clean data. Notably, since the backdoor trigger consistently represents the most prominent feature in the data (Khaddaj et al. 2023), label shuffling can effectively decouple the backdoor trigger from the target label in θ_{cl}, thus significantly reducing the attack success rate. After the training phase, the server uses θ_{cl} for inference.

Experiments

In this section, we conduct extensive experiments to demonstrate the effectiveness of our FILTER framework in defending against backdoor attacks in VFL.

Experimental Setup

Datasets: In the experiments, we evaluated the performance of the FILTER framework on the following five datasets. Specifically, we selected two tabular datasets, namely Bank Marketing (Bank) (Moro, Cortez, and Rita 2014) and German Credit (Credit) (Hofmann 1994), and three image datasets, namely MNIST (LeCun et al. 1998), CIFAR-10 (Krizhevsky 2009), and CINIC-10 (Darlow et al. 2018). We summarize the statistics of these datasets in Table 1.

Dataset	Type	#Samples	#Classes	Input Dimension
Bank	Tabular	41,188	2	16
Credit	Tabular	1,000	2	20
MNIST	Image	60,000	10	$1 \times 28 \times 28$
CIFAR-10	Image	60,000	10	$3 \times 32 \times 32$
CINIC-10	Image	270,000	10	$3 \times 32 \times 32$

Table 1: Statistics of datasets used in the experiments.

Dataset	VILLAIN	Optimal Trigger	Mask	BadVFL
Bank	6	80	6	60
Credit	40	80	40	60
MNIST	14	80	14	60
CIFAR-10	14	80	14	60
CINIC-10	8	60	8	60

Table 2: Epochs in which each attacker begins to use poisoned data to attack the VFL system on the five datasets.

VFL Setting: By default, we adopted a VFL system consisting of two clients and one server, with each client holding half of the features. The features held by different clients do not overlap. We adopt distinct neural network architectures on different datasets. For Bank, the client model is a three-layer Multi-Layer Perceptron (MLP), and the server model is a two-layer MLP. For Credit, the client model is a one-layer MLP, and the server model is also a one-layer MLP. For MNIST, the client model is a Convolutional Neural Network (CNN) with two convolutional layers and two fully connected layers. For CIFAR-10 and CINIC-10, the client model is ResNet-18 (He et al. 2016). The server model is a four-layer MLP for all three image datasets. We do not use data augmentation techniques, such as rotation or translation, for image datasets. This is because such transformations may cause the features of a client to be held by different clients across epochs. The total number of epochs and the batch size during the training phase are set to 100 and 256 for all datasets. We use the cross-entropy loss and the Adam optimizer with a learning rate of 0.001 and a weight decay rate of 0.0001 in all experiments.

Attack and Defense Settings: We follow the same attack and defense settings as used in existing studies on VFL with two clients (Bai et al. 2023; Naseri, Han, and de Cristofaro 2024). The server's default embedding aggregation method is concatenation. We selected three representative backdoor attacks on VFL during the training phase to evaluate the effectiveness of our method, namely VILLAIN (Bai et al. 2023), BadVFL (Naseri, Han, and de Cristofaro 2024), and SplitNN-dedicated data poisoning attack (Optimal Trigger) (He et al. 2024). Furthermore, we evaluated our method against a baseline that uses a traditional replacement trigger $[1, -1, 0, 0, 1, -1, 0, 0, 1, -1]$, which is referred to as Mask. The attacker retains the values of the original embedding vectors at positions where the trigger is zero and replaces them with non-zero values at non-zero positions. For each attack method, we use the settings specified in its original paper. For VILLAIN, we set the number of 1's in the trigger mask $\mathcal{M}$ to 5. For BadVFL, we set the trigger size to 3×3 and use Optimal Selection for the source and target labels.

Attack	Defense	Bank (%)			Credit (%)			MNIST (%)			CIFAR-10 (%)			CINIC-10 (%)			AVG (%)
		CDA	ASR	DES	CDA	ASR	DES	CDA	ASR	DES	CDA	ASR	DES	CDA	ASR	DES	DES
VI	/	89.8	99.5	50.0	71.0	69.0	50.0	99.1	100.0	50.0	72.3	91.0	50.0	58.1	93.6	50.0	50.0
	FG	88.1	0.0	99.0	77.0	98.0	33.2	98.8	11.1	94.3	31.7	5.5	68.9	30.1	80.2	33.1	65.7
	ABL	89.9	97.3	51.2	81.0	91.0	41.1	99.2	91.1	54.5	71.6	67.1	62.6	58.6	80.2	57.6	53.4
	ANP	87.0	10.6	93.1	62.0	0.0	93.7	75.3	15.1	80.4	52.5	87.5	38.2	15.2	0.0	63.1	73.7
	CAE	88.1	0.0	**99.1**	76.0	0.0	**103.5**	77.7	61.5	58.5	63.9	81.1	49.6	41.2	1.3	84.8	79.1
	FILTER (*)	88.2	0.0	**99.1**	79.0	9.0	99.1	98.8	10.4	**94.6**	70.7	10.5	**93.1**	56.5	3.9	**96.6**	**96.5**
OT	/	89.8	100.0	50.0	79.0	100.0	50.0	99.1	66.3	50.0	72.6	94.0	50.0	57.8	93.7	50.0	50.0
	FG	88.0	100.0	49.0	79.0	100.0	50.0	98.4	62.6	52.4	34.6	100.0	20.6	33.7	0.0	79.2	50.2
	ABL	88.5	74.3	62.1	74.0	100.0	46.8	99.0	75.2	43.2	70.6	99.9	45.6	57.5	99.1	46.9	48.9
	ANP	88.1	100.0	49.1	69.0	99.0	44.2	83.1	100.0	16.5	39.4	0.0	77.1	23.0	0.0	69.9	51.3
	CAE	89.5	98.9	50.4	69.0	0.0	93.7	32.7	0.0	66.5	41.6	6.2	75.4	38.0	31.5	66.0	70.4
	FILTER (*)	88.2	0.0	**99.1**	76.0	0.0	**98.1**	98.4	0.0	**99.6**	69.5	0.2	**97.8**	56.2	0.1	**98.5**	**98.6**
BV	/	89.7	77.8	50.0	75.0	42.0	50.0	99.2	45.9	50.0	71.9	73.1	50.0	58.1	93.6	50.0	50.0
	FG	88.1	0.0	99.1	79.0	26.0	71.7	98.9	20.2	**77.8**	30.4	17.8	59.0	36.9	44.5	58.0	73.1
	ABL	89.9	55.4	64.4	72.0	46.0	43.2	99.1	38.6	57.9	69.2	82.5	41.7	53.5	96.7	44.4	50.3
	ANP	88.1	0.0	99.1	62.0	0.0	**91.3**	62.0	44.9	32.4	55.0	88.0	28.1	19.8	0.0	67.0	63.6
	CAE	89.9	58.2	62.7	76.0	24.0	72.1	88.3	19.7	73.0	70.5	59.0	58.7	34.0	0.4	79.1	69.1
	FILTER (*)	88.1	0.0	**99.1**	77.0	11.0	88.2	96.2	27.3	68.7	70.2	23.3	**82.9**	56.5	32.3	**81.4**	**84.1**
MA	/	88.1	100.0	50.0	76.0	100.0	50.0	99.2	100.0	50.0	72.4	88.7	50.0	57.2	92.9	50.0	50.0
	FG	88.1	99.9	50.0	79.0	98.0	53.0	98.1	94.8	52.1	38.1	98.8	20.7	36.1	93.2	31.4	41.4
	ABL	89.5	83.9	58.8	78.0	100.0	51.3	99.0	56.5	71.7	71.4	71.2	59.2	44.6	99.8	35.2	55.2
	ANP	88.1	12.5	93.8	62.0	0.0	90.8	71.0	0.0	85.8	58.7	89.9	39.9	19.5	0.0	67.0	75.5
	CAE	88.1	100.0	50.0	76.0	0.0	**100.0**	79.3	78.6	50.7	61.3	83.3	45.4	41.7	0.3	86.2	66.5
	FILTER (*)	88.1	0.0	**100.0**	76.0	2.0	99.0	98.5	0.0	**99.7**	71.4	17.4	**89.5**	57.0	9.4	**94.8**	**96.6**

Table 3: Overview of the performance of different defense methods against four backdoor attacks in VFL. For attack methods, 'VI', 'OT', 'BV', and 'MA' stand for VILLAIN, Optimal Trigger, BadVFL, and Mask, respectively. For defense methods, "/" indicates that the server does not adopt any defense method.

For Optimal Trigger, we configure the attacker to add noise only to the trigger in the CIFAR-10 dataset, since the attack fails when noise is added to other datasets. The epoch at which the attacker begins the attack is shown in Table 2. To ensure that all attacks can successfully inject the backdoor, the poisoning rate is set to 0.03.

We set the server to start using FILTER at the 85th epoch, with stages 2 and 3 of the loss-based filter beginning at the 86th and 98th epochs, respectively. And p_{init} is set to 0.15, while p_{max} is set to 0.9. The threshold τ of the embedding-based filter is set to 0.001. We also compare four defense methods with FILTER in terms of performance against the above four attacks, namely Fake Gradient (FG) (Jin et al. 2021), ANP (Wu and Wang 2021), ABL (Li et al. 2021a), and CAE (Zou et al. 2024). In particular, FG is used by clients to defend against gradient leakage attacks in VFL. In our setting, the server utilizes FG to generate fake gradients and send them to the clients.

Implementation: All methods were implemented in Python 3 with PyTorch v2.0.1. All experiments were carried out on a cloud server running CentOS 7, equipped with dual Intel® Xeon® Gold 5418Y processors @3.8GHz, 256GB DDR4 memory, and an NVIDIA® GeForce RTX™ 4090 GPU.

Evaluation Metrics: Following prior studies (Gong et al. 2022; Lu et al. 2022), we use the Attack Success Rate (ASR) and Clean Data Accuracy (CDA) to evaluate the performance of backdoor attacks. In addition, we introduce a new evaluation metric, the Defense Effectiveness Score (DES), which offers a balanced assessment of the trade-off between CDA and ASR, as follows:

$$\text{DES} = \Big(\frac{\text{CDA}}{\text{CDA}_{\text{ori}}} - \frac{\text{ASR}}{\max(1/C, \text{ASR}_{\text{ori}})} + 1\Big)/2,$$

where C is the number of classes and CDA_{ori} and ASR_{ori} denote the CDA and ASR in the absence of any defense method. In general, $\text{DES} = 1$ indicates that the defense method reduces ASR to 0 without any decrease in CDA. Each experiment was repeated 5 times, and the average value for each metric across the 5 runs was reported.

Experiment Results

FILTER vs. Existing Defense Methods: Table 3 presents the overall experimental results across all datasets and models in VFL, under different backdoor attack and defense scenarios. From these results, we can draw the following conclusions. First, the FILTER framework effectively reduces the ASR without significantly lowering the CDA. Regardless of the attack method across all datasets, the DES of FILTER is at least 0.8 and is very close to 1 in most cases. Second, FILTER significantly outperforms all other defense methods against backdoor attacks in VFL. The DES of FILTER is much higher than that of the other four defense methods. For example, when the attacker uses VILLAIN, FILTER achieves a DES close to 1.0, whereas none of the other methods achieves a DES greater than 0.8. The ABL approach fails, maybe because the server splits the dataset before the attacker uses poisoned vectors for training. Since the attacker has sufficient label knowledge in our settings,

Dataset Model	K	VILLAIN (%)				Mask (%)				Optimal Trigger (%)				BadVFL (%)			
		w/o Defense		FILTER		w/o Defense		FILTER		w/o Defense		FILTER		w/o Defense		FILTER	
		ASR	CDA	ASR	CDA	ASR	CDA	ASR	CDA	ASR	CDA	ASR	CDA	ASR	CDA	ASR	CDA
CIFAR-10 ResNet-18	2	90.99	72.32	10.52	70.73	88.73	72.36	17.41	71.44	94.04	72.57	0.18	69.47	72.14	71.87	28.46	70.15
	4	97.57	67.73	5.41	65.09	79.20	67.22	14.39	64.27	46.16	64.29	2.88	64.42	73.07	67.59	18.36	64.65
	8	90.56	64.06	0.00	63.45	89.60	64.07	0.19	63.15	54.35	63.77	9.63	63.01	89.60	63.89	23.03	62.18
CIFAR-10 VGG-16	2	76.03	75.84	7.55	74.18	76.37	76.52	25.30	75.95	63.54	76.32	0.00	72.98	52.97	77.32	32.31	72.02
	4	92.41	67.03	4.93	66.37	89.56	69.39	4.79	65.32	53.88	65.64	6.71	63.88	76.18	70.72	5.75	69.67
	8	94.68	60.58	5.42	59.48	81.85	61.29	6.30	60.87	89.19	63.77	11.03	62.14	63.86	64.16	3.42	62.71
CINIC-10 ResNet-18	2	93.58	58.05	3.94	56.52	92.87	57.22	9.38	57.00	93.71	57.81	0.04	56.16	93.58	58.05	32.31	56.52
	4	98.40	55.15	14.43	54.56	86.28	54.65	17.90	53.93	75.94	52.97	0.01	53.07	77.02	54.59	10.20	49.00
	8	93.96	52.71	2.41	51.98	91.10	51.52	8.03	51.21	67.84	52.29	8.78	51.82	84.57	52.44	19.73	49.23
CINIC-10 VGG-16	2	96.59	61.77	14.59	60.60	99.96	61.50	9.38	59.47	99.95	63.73	4.96	57.55	84.09	61.89	21.53	59.67
	4	97.68	57.55	0.64	56.15	94.22	56.84	2.04	55.57	68.71	58.13	2.04	55.57	81.10	56.55	14.42	54.37
	8	98.40	55.15	6.60	52.11	93.40	53.87	0.00	51.85	69.28	53.42	0.00	52.56	85.47	53.61	0.34	52.71

Table 4: Performance of FILTER in the multi-participant scenario.

Method	VILLAIN		Mask		Optimal Trigger		BadVFL	
	ASR	CDA	ASR	CDA	ASR	CDA	ASR	CDA
No Defense	90.99%	**72.32%**	88.73%	**72.36%**	94.04%	**72.57%**	72.14%	**71.87%**
FILTER + Meta-Split	31.44%	69.35%	31.14%	70.23%	**0.00%**	71.90%	83.31%	69.48%
FILTER w/o Unlearning	55.13%	71.06%	28.65%	71.87%	0.00%	69.86%	32.64%	69.32%
FILTER w/o Embedding-based Filter	14.05%	68.21%	**12.74%**	69.93%	84.86%	70.67%	30.46%	70.45%
FILTER	**10.52%**	70.73%	17.41%	71.44%	0.18%	69.47%	**28.46%**	70.15%

Table 5: Ablation results of FILTER on the CIFAR-10 dataset using ResNet-18 as the client model.

using CAE to map the original labels to a set of "soft fake labels" does not compromise the backdoor injection. As with CAE, if the server uses FG to defend against backdoor attacks, the fake gradients only make it more difficult for the malicious client to extract additional information from the gradients and do not effectively prevent backdoor attacks. Since the server cannot access other participants' local models and can only prune its own model, the effectiveness of ANP is also reduced. Third, since the clean server warm-up of the loss-based filter relies on the low loss of poisoned samples, but the ASR of some attacks is lower than that of CDA, even without defense, FILTER may be less effective at eliminating backdoors in some cases. For example, when training on the MNIST dataset and using BadVFL attacks, FILTER only achieves a DES of 0.687.

Performance in Multi-participant Scenarios: Since all four attacks we experimented with can be extended to VFL with more than two clients, we extend our experiments to multi-client scenarios. Specifically, we consider configurations with $K = 2$, 4, and 8 clients using the CIFAR-10 and CINIC-10 datasets, with ResNet-18 and VGG-16 as local models. Similarly to the default setting, each client owns 50%, 25%, and 12.5% of the feature vector when $K = 2$, 4, and 8, respectively.

As shown in Table 4, FILTER consistently demonstrates strong defensive performance across all client configurations. In particular, it effectively mitigates backdoor attacks, even as the number of clients increases. In all scenarios, the ASR remains low, while the CDA experiences little to no degradation. In most settings, FILTER reduces the ASR to below 20%, while keeping the CDA comparable to the baseline without any attack.

Attack	Metric	$p_{\text{init}} = 0.1$	$p_{\text{init}} = 0.15$	$p_{\text{init}} = 0.2$
VI	CDA	68.02%	**70.73%**	69.47%
	ASR	11.43%	**10.52%**	14.24%
BV	CDA	**71.13%**	70.15%	68.76%
	ASR	**22.95%**	23.25%	35.17%
MA	CDA	68.81%	**71.44%**	67.22%
	ASR	**6.13%**	17.41%	15.74%
OT	CDA	68.55%	**69.47%**	69.01%
	ASR	**0.12%**	0.18%	0.31%

Table 6: Performance of FILTER with varying p_{init}.

Effect of Parameter p_{init}: To evaluate the impact of different values of p_{init} on FILTER's performance, we conduct experiments on the CIFAR-10 dataset using ResNet-18 under the default VFL and attack settings. The value of p_{init} is selected from $\{0.1, 0.15, 0.2\}$. As shown in Table 6, we observe a trade-off between model utility and defense effectiveness w.r.t. the value of p_{init}: a lower p_{init} often leads to a reduced CDA due to insufficient early sampling, which means that fewer benign samples are selected for the initial stages of training, thus impeding the overall performance of the model. On the other hand, a higher p_{init} can compromise FILTER's ability to effectively suppress backdoor behaviors, as it allows more samples to participate in the early stages, resulting in a higher ASR. Among the values tested, $p_{\text{init}} = 0.15$ provides the best balance across most types of attacks and is used as the default setting.

Effect of Parameter τ: To evaluate the impact of different values of τ on FILTER's performance, we conduct experiments under the default VFL and attack settings using the

Attack	Metric	$\tau = 0$	$\tau = 0.001$	$\tau = 0.01$	$\tau = 0.1$
VI	CDA	68.93%	**70.73%**	66.91%	66.67%
	ASR	12.74%	10.52%	**1.20%**	14.50%
BV	CDA	68.68%	**70.15%**	69.52%	69.36%
	ASR	24.36%	**23.25%**	24.36%	52.18%
MA	CDA	67.63%	**71.44%**	65.58%	65.21%
	ASR	14.33%	17.41%	**13.52%**	13.80%
OT	CDA	69.32%	69.47%	**69.56%**	69.12%
	ASR	94.92%	0.18%	0.20%	**0.04%**

Table 7: Performance of FILTER with varying τ.

CIFAR-10 dataset and the ResNet-18 model. Specifically, we set $\tau \in \{0, 0.001, 0.01, 0.1\}$ in the experiments. The results are presented in Table 7. We observe that the performance of FILTER is sensitive to the choice of τ. In particular, when τ is too large (e.g., $\tau = 0.1$), FILTER tends to retain a lower ASR at the expense of a reduced CDA, indicating that a stronger defense can hinder the utility of the model. This degradation is especially noticeable under the BV and MA attack settings. In contrast, smaller values of τ (e.g., 0.001) often yield a more balanced performance, maintaining both a lower ASR and a higher CDA. Based on these results, we use $\tau = 0.001$ as the default value.

Ablation Study: We conducted an ablation study to evaluate the effectiveness of each component of the FILTER framework on the CIFAR-10 dataset with two clients. The client model is also ResNet-18, while the server model is a four-layer MLP, as described in Section . Since clean server warm-up is indispensable for the subsequent stages of the loss-based filter, we only tested the remaining two stages in the ablation study. To demonstrate the effectiveness of Score in Eq. (1) for the loss-based filter, we tested FILTER by replacing it with an alternative one in meta-split (Gao et al. 2023a) as follows:

$$\text{Score}_{\text{meta}} = \|\mathcal{L}(\theta_s(\mathbf{e}_B), \mathbf{y}_B) - \mathcal{L}(\theta_{cl}(\mathbf{e}_B), \mathbf{y}_B)\|_2.$$

The ablation results are presented in Table 5. We draw the following conclusions. First, the embedding-based filter is effective only when the Optimal Trigger is used, as it is the only attack among the four that replaces complete vectors with a trigger. The results for the other three attacks without the embedding-based filter can be attributed to the very small threshold τ, which helps retain the CDA. At the same time, when the embedding-based filter is absent, the CDA decreases for VILLAIN and MASK but increases for BadVFL and the optimal trigger. This occurs because the embedding differences between backdoor and benign embeddings are larger for the latter and smaller for the former. As a result, for the first two methods, some benign samples are more likely to be misidentified as backdoor samples. Second, backdoor unlearning significantly reduces the ASR, while slightly decreasing the CDA. Third, using Score in Eq. (1) is more effective in defending against all four types of backdoor attacks than the meta-split approach. This suggests that, compared to meta-split, the loss-based filter better distinguishes between backdoor and benign samples.

Effect of Late Poisoning and Early Stopping: By default, the server applies FILTER after the 85th epoch. However, a malicious client may launch an attack after this point. We consider two scenarios: the malicious client initiates the attack at the 90th epoch (i.e., Late Poison) or the client stops participating before the final two epochs (i.e., Early Stop). Here, "Baseline" refers to the scenario in which the attacker injects the backdoor under the default settings. To evaluate the robustness of FILTER under such conditions, we conduct experiments on the CIFAR-10 dataset, where the attacker begins the attack at epoch 90.

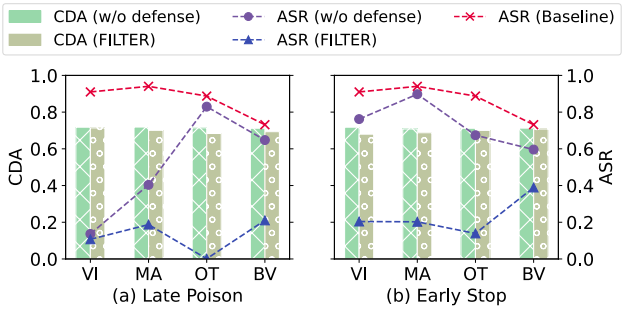

Figure 3: Performance of FILTER when the attacker adopts Late Poison or Early Stopping.

As shown in Figure 3(a), FILTER maintains a low ASR while its CDA does not degrade significantly. Since the server will use low-score vectors for unlearning during the last two training epochs, we evaluate FILTER's performance when the attacker stops poisoning at the 97th epoch. As shown in Figure 3(b), if the attacker stops using poisoned samples or vectors at the 97th epoch, FILTER still reduces the ASR without significantly decreasing the CDA. In particular, both late poisoning and early stopping reduce the ASR compared to the default setting, suggesting that attackers should avoid timing attacks at the start or end stages of training. In particular, both late poisoning and early stopping reduce ASR compared to the default setting, indicating that attackers should avoid launching attacks during the initial or final stages of training.

Conclusion

In this paper, we proposed a framework, FILTER, for defending against backdoor attacks during training, which consists of two filters: an embedding-based filter and a loss-based filter. Our method enabled participants in a VFL system to train backdoor-free models, even when a malicious client attempted to inject a backdoor. We evaluated our method on five datasets and four typical backdoor attacks in VFL. Extensive experiments have demonstrated that our method successfully reduces the VFL system's ASR while almost entirely preserving the CDA.

Acknowledgements

This research was supported in part by the National Natural Science Foundation of China (Grant Nos. 62202170 and 62202169) and Guizhou Provincial Program on Commercialization of Scientific and Technological Achievements (Qiankehezhongyindi [2025] No. 006).

References

Bagdasaryan, E.; Veit, A.; Hua, Y.; Estrin, D.; and Shmatikov, V. 2020. How to backdoor federated learning. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, 2938–2948.

Bai, Y.; Chen, Y.; Zhang, H.; Xu, W.; Weng, H.; and Goodman, D. 2023. VILLAIN: Backdoor Attacks Against Vertical Split Learning. In *32nd USENIX Security Symposium*, 2743–2760.

Cheng, Y.; Liu, Y.; Chen, T.; and Yang, Q. 2020. Federated learning for privacy-preserving AI. *Communications of the ACM*, 63(12): 33–36.

Darlow, L. N.; Crowley, E. J.; Antoniou, A.; and Storkey, A. J. 2018. CINIC-10 is not ImageNet or CIFAR-10. arXiv:1810.03505.

Ester, M.; Kriegel, H.; Sander, J.; and Xu, X. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 226–231.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, 1126–1135.

Gao, K.; Bai, Y.; Gu, J.; Yang, Y.; and Xia, S.-T. 2023a. Backdoor defense via adaptively splitting poisoned dataset. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4005–4014.

Gao, X.; and Zhang, L. 2023. PCAT: Functionality and data stealing from split learning by Pseudo-Client attack. In *32nd USENIX Security Symposium*, 5271–5288.

Gao, Y.; Li, Y.; Zhu, L.; Wu, D.; Jiang, Y.; and Xia, S. 2023b. Not All Samples Are Born Equal: Towards Effective Clean-Label Backdoor Attacks. *Pattern Recognition*, 139: 109512.

Gong, X.; Chen, Y.; Dong, J.; and Wang, Q. 2022. ATTEQ-NN: Attention-based QoE-aware Evasive Backdoor Attacks. In *29th Annual Network and Distributed System Security Symposium*.

Gu, T.; Liu, K.; Dolan-Gavitt, B.; and Garg, S. 2019. BadNets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7: 47230–47244.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

He, Y.; Shen, Z.; Hujingyu; Dong, Q.; Niu, J.; Tong, W.; Huang, X.; Li, C.; and Zhong, S. 2024. Backdoor Attack Against Split Neural Network-Based Vertical Federated Learning. *IEEE Transactions on Information Forensics and Security*, 19: 748–763.

Hofmann, H. 1994. Statlog (German Credit Data). https://doi.org/10.24432/C5NC77.

Hospedales, T.; Antoniou, A.; Micaelli, P.; and Storkey, A. 2021. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9): 5149–5169.

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

Khaddaj, A.; Leclerc, G.; Makelov, A.; Georgiev, K.; Salman, H.; Ilyas, A.; and Madry, A. 2023. Rethinking backdoor attacks. In *Proceedings of the 40th International Conference on Machine Learning*, 16216–16236.

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Li, Y.; Jiang, Y.; Li, Z.; and Xia, S.-T. 2022. Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(1): 5–22.

Li, Y.; Lyu, X.; Koren, N.; Lyu, L.; Li, B.; and Ma, X. 2021a. Anti-backdoor learning: Training clean models on poisoned data. *Advances in Neural Information Processing Systems*, 34: 14900–14912.

Li, Y.; Lyu, X.; Koren, N.; Lyu, L.; Li, B.; and Ma, X. 2021b. Neural Attention Distillation: Erasing Backdoor Triggers from Deep Neural Networks. In *The 9th International Conference on Learning Representations*.

Liu, Y.; Kang, Y.; Zou, T.; Pu, Y.; He, Y.; Ye, X.; Ouyang, Y.; Zhang, Y.-Q.; and Yang, Q. 2024. Vertical federated learning: Concepts, advances, and challenges. *IEEE Transactions on Knowledge and Data Engineering*, 36(7): 3615–3634.

Lu, S.; Li, R.; Liu, W.; and Chen, X. 2022. Defense against backdoor attack in federated learning. *Computers & Security*, 121: 102819.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 1273–1282.

Morales, D.; Agudo, I.; and Lopez, J. 2023. Private set intersection: A systematic literature review. *Computer Science Review*, 49: 100567.

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

Naseri, M.; Han, Y.; and de Cristofaro, E. 2024. BadVFL: Backdoor Attacks in Vertical Federated Learning. In *2024 IEEE Symposium on Security and Privacy*, 2013–2028.

Qi, X.; Xie, T.; Wang, J. T.; Wu, T.; Mahloujifar, S.; and Mittal, P. 2023. Towards a proactive ML approach for detecting backdoor poison samples. In *32nd USENIX Security Symposium*, 1685–1702.

Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86): 2579–2605.

Wang, B.; Yao, Y.; Shan, S.; Li, H.; Viswanath, B.; Zheng, H.; and Zhao, B. Y. 2019. Neural Cleanse: Identifying and

Mitigating Backdoor Attacks in Neural Networks. In *2019 IEEE Symposium on Security and Privacy*, 707–723.

Wang, H.; Sreenivasan, K.; Rajput, S.; Vishwakarma, H.; Agarwal, S.; Sohn, J.-y.; Lee, K.; and Papailiopoulos, D. 2020. Attack of the tails: Yes, you really can backdoor federated learning. *Advances in Neural Information Processing Systems*, 33: 16070–16084.

Wu, D.; and Wang, Y. 2021. Adversarial neuron pruning purifies backdoored deep models. *Advances in Neural Information Processing Systems*, 34: 16913–16925.

Yang, Q.; Liu, Y.; Chen, T.; and Tong, Y. 2019. Federated Machine Learning: Concept and Applications. *ACM Transactions on Intelligent Systems and Technology*, 10(2): 12:1–12:19.

Zhang, Z.; Panda, A.; Song, L.; Yang, Y.; Mahoney, M.; Mittal, P.; Kannan, R.; and Gonzalez, J. 2022. Neurotoxin: Durable backdoors in federated learning. In *Proceedings of the 39th International Conference on Machine Learning*, 26429–26446.

Zheng, R.; Tang, R.; Li, J.; and Liu, L. 2022. Data-Free Backdoor Removal Based on Channel Lipschitzness. In *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part V*, 175–191.

Zhu, C.; Zhang, J.; Sun, X.; Chen, B.; and Meng, W. 2023. ADFL: Defending backdoor attacks in federated learning via adversarial distillation. *Computers & Security*, 132: 103366.

Zou, T.; Liu, Y.; Kang, Y.; Liu, W.; He, Y.; Yi, Z.; Yang, Q.; and Zhang, Y.-Q. 2024. Defending batch-level label inference and replacement attacks in vertical federated learning. *IEEE Transactions on Big Data*, 10(6): 1016–1027.