# VILLAIN: Backdoor Attacks Against Vertical Split Learning

Yijie Bai and Yanjiao Chen, *Zhejiang University;* Hanlei Zhang and Wenyuan Xu, *Zhejing University;* Haiqin Weng and Dou Goodman, *Ant Group*

https://www.usenix.org/conference/usenixsecurity23/presentation/bai

This paper is included in the Proceedings of the
32nd USENIX Security Symposium.

August 9–11, 2023 • Anaheim, CA, USA

978-1-939133-37-3

# VILLAIN: Backdoor Attacks Against Vertical Split Learning

Yijie Bai
*Zhejiang University*
*baiyj@zju.edu.cn*

Yanjiao Chen
*Zhejiang University*
*chenyanjiao@zju.edu.cn*

Hanlei Zhang
*Zhejiang University*
*hanleizhang@zju.edu.cn*

Wenyuan Xu
*Zhejiang University*
*wyxu@zju.edu.cn*

Haiqin Weng
*Ant Group*
*haiqin.wenghaiqin@antgroup.com*

Dou Goodman
*Ant Group*
*bencao.ly@antgroup.com*

## Abstract

Vertical split learning is a new paradigm of federated learning for participants with vertically partitioned data. In this paper, we make the first attempt to explore the possibility of backdoor attacks by a malicious participant in vertical split learning. Different from conventional federated learning, vertical split learning poses new challenges for backdoor attacks, the most looming ones being a lack of access to the training data labels and the server model. To tackle these challenges, we propose VILLAIN, a backdoor attack framework that features effective label inference and data poisoning strategies. VILLAIN realizes high inference accuracy of the target label samples for the attacker. Furthermore, VILLAIN intensifies the backdoor attack power by designing a stealthy additive trigger and introducing backdoor augmentation strategies to impose a larger influence on the server model. Our extensive evaluations on 6 datasets with comprehensive vertical split learning models and aggregation methods confirm the effectiveness of VILLAIN. It is also demonstrated that VILLAIN can resist the popular privacy inference defenses, backdoor detection or removal defenses, and adaptive defenses.

## 1 Introduction

The ever-increasing interests in data privacy have boosted the demand for federated learning, where multiple data owners collaboratively train a model without disclosing their private data. According to how data is distributed among various participants, federated learning can be classified into two main categories, i.e., horizontal and vertical federated learning [67]. A complete training data sample consists of the sample index, the feature vector, and the label. In horizontal federated learning, datasets of participants share the same feature space but different sample spaces, i.e., every participant owns full feature vectors, but the sample indexes of different participants do not overlap. For instance, two hospitals have similar healthcare services, so their feature spaces are the same, but their patient groups are different. In contrast, in vertical federated learning, datasets of participants share the same sample

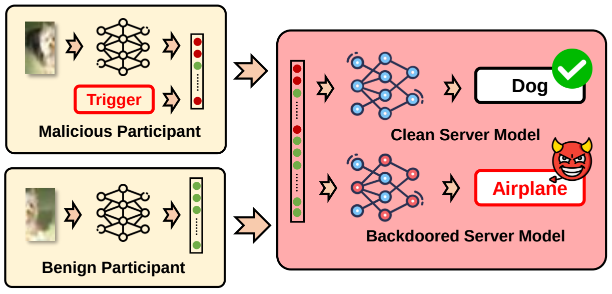


Figure 1: Scenario of VILLAIN. A malicious participant aims to inject a backdoor into the server model by uploading poisoned embedding vectors to the server.

space but differ in feature spaces, i.e., every participant owns a non-overlapping partial feature vector, and their sample indexes are the same. For example, a national bank and an insurance company have the same large customer base while they record different transaction features of these customers. Google Gboard [3] is a successful application of horizontal federated learning. WeBank (China) drives FedAI [1], an open-source federated AI ecosystem that implements both horizontal and vertical federated learning.

Apart from privacy issues, security threats also concern federated learning, among which backdoor attacks have been made against horizontal federated learning. A malicious participant can construct poisoned weights based on poisoned local data and upload the poisoned weight to insert the backdoor in the global model maintained by the server. The participant can further control the backdoored model to behave normally on clean samples but yield misclassification results to a target label on samples with a specially-designed trigger. While extensive research has been conducted on backdoor attacks in horizontal federated learning [5] [62], there is a lack of works on backdoor attacks in vertical federated learning.

In this paper, we make the first attempt to explore the possibility of backdoor attacks in vertical federated learning. More specifically, we concentrate on vertical split learning [54] [59], a state-of-the-art vertical federated learning framework, as shown in Figure 1. Similar to existing works, our goal is to enable a malicious participant to insert a backdoor in the server model via poisoned uploads, in order to control the server model with the trigger. Unfortunately, the workflow of vertical split learning poses significant challenges to backdoor attacks.

(1) ***No access to label***. Different from the participant in horizontal federated learning with full control over the labels of the local dataset, the participant in vertical federated learning only has partial features but no label information of the data samples, which is indispensable for targeted backdoor attacks. Only the server possesses the ground-truth labels and the attacker has no access to labels. This also entails *clean-label* backdoor attacks in vertical split learning, which are much more difficult than dirty-label attacks in horizontal federated learning.

(2) ***No access to server model***. A selected participant in a certain training epoch receives the entire global model from the server in horizontal federated learning. In contrast, the participant in vertical split learning only gets gradient information from the server. This makes it difficult to optimize a trigger that imposes a great influence on the server model.

To address these challenges, we propose the design, implementation and evaluation of VILLAIN, a backdoor attack framework against vertical split learning. VILLAIN consists of two modules, i.e., *label inference* and *data poisoning*, which allow a malicious participant to launch backdoor attacks even though the data label and the server model are unknown. The label inference module aims to deduce whether a sample belongs to the target misclassification label or not, which is essential for creating poisoned updates in clean-label attacks. A novel embedding swapping algorithm is designed for efficient and accurate label inference with one known sample of the target label. The data poisoning module fabricates poisoned updates based on the label inference results. To intensify the impact of poisoned updates on the server model, we choose the most important elements in the update vector as the trigger mask and introduce randomness into the trigger to improve its robustness.

We conduct comprehensive experiments to evaluate the performance of VILLAIN on four image datasets (MNIST [32], CIFAR-10 [31], CINIC-10 [12], ImageNette [23]) and two financial datasets (Bank Marketing [41], and Give-Me-Some-Credit [2]), with five vertical split learning models. It is demonstrated that VILLAIN achieves an average attack success rate of more than 90% with a poison rate of only 1%. We further evaluate the resistance of VILLAIN to backdoor defenses and show that VILLAIN maintains a high attack success rate under five state-of-the-art defense methods.

The main contributions are summarized as follows:

- We propose a backdoor attack framework, named VILLAIN, for vertical split learning, which realizes effective attacks in spite of a lack of access to both the label and the server model in vertical split learning.
- We develop an efficient label inference algorithm that can work with a labeled sample. We design a data poisoning strategy with additive triggers to reinforce the backdoor and with trigger randomization strategies to mitigate backdoor overfitting in the server model.
- We conduct extensive experiments to validate the effectiveness, robustness, and efficiency of our attack. It is also shown that VILLAIN survives existing and adaptive backdoor defenses.

## 2 Preliminaries

### 2.1 Vertical Split Learning

Collaborative machine learning or distributed learning is an appealing way to build large-scale high-quality machine learning models by cooperative participants [57]. Federated learning is an important protocol to realize collaborative machine learning. According to the data sharing pattern among clients, federated learning can be categorized into *horizontal learning* and *vertical learning*. In horizontal learning, the participants have labeled datasets with the same feature space but different sample spaces. In vertical learning, the participants have datasets with different feature spaces but the same sample space. In this paper, we focus on vertical federated learning. Vertical federated learning has been used in real-world applications such as credit evaluation and online advertising [33] [70]. For credit evaluation, a bank often uses external non-credit data to better evaluate the credit of an individual, e.g., transaction data from online shopping platforms or billing data from a telecommunication company. For online advertising, an ad company can cooperate with social media platforms to better recommend ads to individuals based on their browsing history and their social interactions.

Vertical split learning is a state-of-the-art vertical federated learning method that splits the global model into different parts and assigns each part to a participant to train. More specifically, each participant trains a local embedding model with the gradient from the server and sends the embedding vectors to the server for aggregation. The server has the label to train a final classifier based on the aggregated embedding vectors via supervised learning. Since backdoor attacks have been extensively studied in horizontal learning, we focus on vertical learning in this paper. As shown in Figure 2, we consider a vertical split learning scenario with a group of participants $\{\mathcal{C}^k\}_{k=1}^K$. Let $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ denote the complete dataset, in which each data sample has an $M$-dimensional feature vector $\mathbf{x}_i = [x_{i,1}, x_{i,2}, \cdots, x_{i,M}]$. Participant $\mathcal{C}^k$ owns a partial feature vector of each data sample, i.e., $\mathbf{X}^k = \{\tilde{\mathbf{x}}_i^k\}_{i=1}^N$,

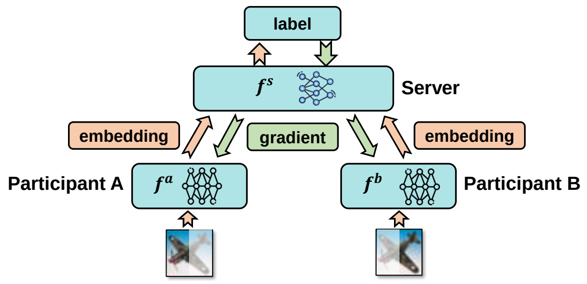


Figure 2: An illustration of vertical split learning. Each participant trains a local embedding model and sends the embedding vector to the server. The server leverages the aggregated embedding vectors and the label to train a classifier, and back-propagates the corresponding gradient information to each participant to update their embedding models.

and we have $\bigcup_{k=1}^{K} \tilde{\mathbf{x}}_i^k = \mathbf{x}_i$. The ground-truth label of $\mathbf{x}_i$ is denoted as $y_i$, owned only by the server.

Vertical split learning is carried out iteratively in the following three steps.

- **Step I** Participant $\mathcal{C}^k$ produces an $M^k$-dimensional embedding vector by the embedding model $f^k$ as

$$\mathbf{e}_i^k = f^k(\tilde{\mathbf{x}}_i^k), \tag{1}$$

  where $\mathbf{e}_i^k$ is the $k$-th embedding of data $\mathbf{x}_i$ from the $k$-th participant. All participants send their embedding vectors to the server.

- **Step II** The server combines the embedding vectors of all participants with a certain aggregation function as the whole embedding $\mathbf{e}_i = \mathcal{A}(\mathbf{e}_i^1, \mathbf{e}_i^2, \cdots, \mathbf{e}_i^K)$. The server leverages $\mathbf{e}_i$ and the label $y_i$ to train the classifier $f(\mathbf{e}_i) = y_i$ with supervised learning. After training, the server back-propagates the corresponding gradient information $\mathbf{g}_k = \bigtriangledown_{\mathbf{e}_i^k} \mathcal{L}$ to each client, where $\mathcal{L}$ is the loss function of $f$.

- **Step III** $\mathcal{C}^k$ uses $\mathbf{g}^k$ to update its embedding model $f^k$ with a learning rate of $\alpha^k$. After updating, all participants send the new embedding vectors of the next data batch to the server until convergence.

In the inference phase, the input $\mathbf{x}$ is also held separately by the participants. Each participant sends the embedding $\mathbf{e}^k$ to the server, and the server derives the prediction result. The prediction result may not be returned to the participants.

Vertical split learning was first proposed to enable different hospitals and tele-health screening centers to jointly train a disease prediction model with complementary medical records [61] [26]. Note that vertical split learning may be materialized with different model structures, such as Graph Neural Network (GNN) [55] and Recurrent Neural Network (RNN) [71]. Split learning can also be implemented based on self-supervised learning tasks without labels [68].

## 2.2 Backdoor Attacks

Backdoor attacks aim to train a backdoored model that behaves normally on clean inputs but misclassifies special inputs (inputs with a trigger) into a target label (targeted attacks) or any wrong label (untargeted attacks). In this paper, we focus on targeted attacks. The linchpins of backdoor attacks are the *trigger* and the *backdoor*. The trigger is optimized with desirable properties, in particular, imperceptibility achieved by limiting the size or the transparency of the trigger. The trigger will be integrated with clean training samples of to create poisoned training samples. The backdoor establishes the association between the trigger and the target (misclassification) label with the help of poisoned training samples.

According to the label of poisoned samples, backdoor attacks can be categorized into *dirty-label attacks* and *clean-label attacks*. In dirty-label attacks, the attacker adds the trigger to the clean data of the other labels and then changes their labels to the target label [9] [20]. But in clean-label attacks, the attackers cannot change the true label of poisoned training samples so they only add triggers to the training data of the target label [25] [69]. With greater freedom, dirty-label attacks can better impose the influence of the trigger on the target misclassification label than clean-label attacks. The label information is indispensable to both the attacks.

Federated learning is ideal for an attacker to carry out backdoor attacks as a participant since the server is not allowed to inspect local data and local models of participants. Backdoor attacks have been widely studied in horizontal learning scenarios, where the attacker uploads poisoned weights to the server to insert the backdoor in the global model. The attacker may augment the poisoned updates to enhance the backdoor injection effect.

Unfortunately, backdoor attacks in horizontal learning are not feasible for vertical learning due to two main reasons.

*No label information*. In horizontal learning, all participants know the true labels of their local dataset. To conduct the targeted backdoor attack, the attacker chooses a specific part of the training data to poison according to the label information and can also manipulate the label of the data to conduct dirty-label attacks. Nevertheless, in vertical learning, no participant but the server knows the true labels of each training sample, so the targeted backdoor attack gets much more difficult. The attacker is also compelled to launch clean-label attacks, much more difficult than dirty-label attacks.

*No server model information*. In horizontal learning, all participants and the server work on the same global model with the same input feature space. The global model will be distributed to selected participants in each training epoch, which makes it easier for the attacker to compute an optimal poisoned weight. Nonetheless, in vertical learning, the participants and the server work on different models due to different feature spaces. The server only sends the gradient update information to the participant such that it is difficult for the attacker to poison the uploaded embedding vector to inject the backdoor into the unknown server model.

Our proposed backdoor attack framework, VILLAIN, addresses the above challenges and realizes successful backdoor attacks in vertical split learning.

## 2.3 Threat Model

We define the threat model of backdoor attacks in vertical split learning in terms of the goals, knowledge and capability of the attacker. Without loss of generality, we assume that there is one participant in vertical split learning who is the attacker, denoted as $\mathcal{C}^a$. Note that VILLAIN can be easily extended to the multi-attacker case.

**Attacker's goal.** The attacker aims to inject a backdoor into the model of the server during training. In the inference phase, if the attacker uploads a benign embedding vector to the server, the backdoored model of the server will output the correct label. If the attacker uploads a triggered embedding vector to the server, the backdoored model of the server will output the target label. Note that our targeted backdoor attack is a special case of untargeted backdoor attacks, where the triggered inputs can induce the server to output any wrong label. Compared with untargeted backdoor attacks, targeted backdoor attacks are more challenging.

**Attacker's knowledge.** The attacker has a local dataset $\mathbf{X}^a = \{\tilde{\mathbf{x}}_i^a\}_{i=1}^N$ with correct sample indexes. The attacker knows but cannot alter the indexes of samples used in each epoch (i.e., the training batch). The attacker has no knowledge of the labels of its dataset. Note that the server does not reveal the labels of training samples during the training phase but may return the predicted labels to users during the inference phase. Therefore, the attacker can check the attacker's performance during the inference phase. The attacker originally has access to one target label sample. The attacker acquires corresponding gradient information $\mathbf{g}^a$ back-propagated from the server. Note that the server will coordinate all participants to send the embedding vectors of the same data batch in each epoch [10] [67].

**Attacker's capability.** The attacker can train a local embedding model $f^a$ based on $\mathbf{X}^a$. The output of $f^a$ has a dimension of $M^a$, which is designated by the server. The attacker can upload the embedding vectors to the server. The attacker can execute the attack through multiple epochs. The attacker can select which epochs to poison and which samples to utilize for poisoning. The attacker does not have access to the server model or the local models held by other participants.

## 3 VILLAIN: Detailed Construction

The ultimate goal of VILLAIN is to insert a backdoor in the server model in vertical split learning. To fulfil this goal, as shown in Figure 3, our proposed attack framework consists of two modules, i.e., label inference and data poisoning. The *label inference* module aims to address the challenge of *no label information* for the attacker. By pinpointing the data samples that belong to the target label, the attacker is able to carry out clean-label backdoor attacks. The *data poisoning* module aims to tackle the difficulty of *no global model information* for the attacker. We carefully design the poisoned data to introduce the backdoor into the global model.

*Label inference*. For a participant to launch a backdoor attack in vertical split learning, the lack of label information and the inability to control the labeling process create a dilemma. On one hand, since the attacker cannot falsify the label of training samples, the backdoor attack has to be conducted in a *clean-label* manner. On the other hand, in clean-label attacks, the only way to establish the link between the trigger and the target label is by poisoning the samples of the target label, which, however, requires the attacker to *know* the label of data samples. To resolve this dilemma, we carefully design a novel label inference approach, which is able to pinpoint data samples of the target label.

*Data poisoning*. The lack of server model information, including input dimensions and parameters, prevents the attacker from constructing poisoned training samples that impose the most significant influence on the victim model (i.e., the server model). Furthermore, the server may employ defense strategies, including backdoor detection and backdoor removal, to try to thwart backdoor attacks. To tackle these difficulties, we propose a data poisoning algorithm that is able to perform backdoor injection in an effective and stealthy way.

## 3.1 Label Inference

As we have explained, backdoor attacks in vertical split learning can only be conducted in a clean-label manner, which is much less effective than the dirty-label method adopted in most existing backdoor attacks. In dirty-label attacks, the adversary can willfully assign the wrong label to a training sample and can utilize any sample for poisoning. More specifically, the adversary can add the trigger to any benign sample (of any label) and change its label to the target label such that the backdoor between the trigger and the target label can be learned during training. Nonetheless, in clean-label attacks, a trigger added to a non-target-label sample is nugatory, which means that the adversary can only establish the link between the trigger and the target label by poisoning the samples of the target label. Therefore, it is necessary for the adversary to know if a sample belongs to the target label or not.

Our label inference algorithm consists of three modules,

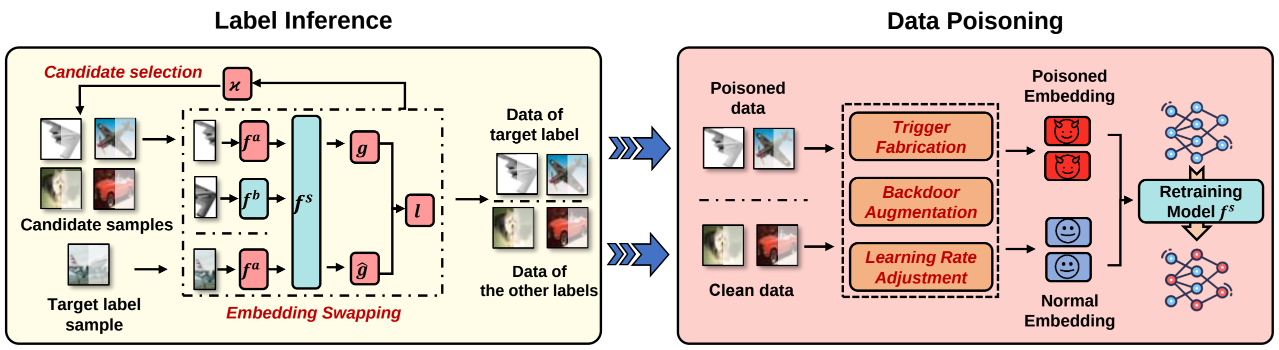


Figure 3: Overview of VILLAIN. VILLAIN consists of the label inference module and the data poisoning module. The label inference module leverages embedding swapping, candidate selection and inference adjustment to pinpoint samples of the target label. The data poisoning module leverages trigger fabrication, backdoor augmentation and learning rate adjustment to hijack the server model.

i.e., embedding swapping, candidate selection, and inference adjustment. In the *embedding swapping* module, we propose a novel algorithm that leverages the relationship between the uploaded embedding and the back-propagated gradient to infer whether a sample belongs to the target label or not. In the *candidate selection* module, we design a selection rule to narrow down the samples that are most likely to be from the target label. In the *inference adjustment* module, we dynamically adjust the embedding vector used for label inference to achieve stealthiness.

### 3.1.1 Embedding Swapping

To make full use of the knowledge available to the attacker, we design a new label inference algorithm that explores the relationship between the forwarded embedding vector (by the attacker) and the back-propagated gradient (from the server). Our intuition is that a forwarded *unaltered* embedding vector from the (well-trained) embedding model of the attacker will induce a small loss; thus, the back-propagated gradient will be small. However, if the attacker intentionally alters the forwarded embedding of a sample to the embedding of another class, the back-propagated gradient will be relatively large.

Let $\mathbf{e}_t^a = f^a(\tilde{\mathbf{x}}_t^a)$ denote the embedding vector of a known (partial) sample $\tilde{\mathbf{x}}_t^a$ of the target label $y_t$, and $\mathbf{e}_i^a = f^a(\tilde{\mathbf{x}}_i^a)$ denote the embedding vector of a sample $\tilde{\mathbf{x}}_i^a$ whose label is unknown. During training, if index $i$ is in the current training batch, the attacker uploads the original embedding vector $\mathbf{e}_i^a$ to the server and obtains a back-propagated gradient $\mathbf{g}_i^a$. The next time that index $i$ is in the training batch again, the attacker instead uploads $\mathbf{e}_t^a$ to the server and obtains a back-propagated gradient $\hat{\mathbf{g}}_i^a$. The label of $\tilde{\mathbf{x}}_i^a$ is likely to be the target label if

$$\frac{||\hat{\mathbf{g}}_i^a||_2}{||\mathbf{g}_i^a||_2} \le \theta, \qquad (2)$$

and

$$||\mathbf{g}_i^a||_2 \le \mu, \qquad (3)$$

where $||\cdot||_2$ represents the $L_2$ norm, $\theta$ and $\mu$ are two threshold parameters. The changing ratio and the gradient norm values can be combined to form the accurate inference label inference judgment conditions. Condition (2) means that replacing the embedding vector of $\tilde{\mathbf{x}}_i^a$ with the embedding vector of $\tilde{\mathbf{x}}_t^a$ will not induce high training loss, which indicates that the label of $\tilde{\mathbf{x}}_i^a$ is very likely to be $y_t$. Note that we do not consider the direction of gradients in condition (2) since the direction of gradients indicates to which direction the model parameters need to be updated to reduce the training loss. No matter to which direction the model parameters need to be updated, a large update indicates a high training loss and a small update indicates a small training loss. Therefore, it is the magnitude but not the direction of gradients that yields the label information. In addition, we withhold embedding swapping during the first several epochs of model training to avoid instability. Condition (3) means the norm of the gradient is small in the previous training, so the norm training loss of $\tilde{\mathbf{x}}_i^a$ is relatively small, which indicates the server model is more confident in predicting $\tilde{\mathbf{x}}_i^a$ as the target label.

### 3.1.2 Candidate Selection

The prior works [16] conduct privacy inference in vertical split learning by training an inference model that predicts the label of a sample based on the output of the attacker's embedding model. The inference model is trained in a supervised or semi-supervised manner with a set of labeled samples. Instead of performing embedding swapping for all training samples (of which many do not belong to the target label), it is wiser for the attacker to select candidate samples that are most likely to belong to the target label for embedding swapping. This will improve the algorithm accuracy and avoid as much as possible to influence the server model continuing learning.

To realize this objective, we build a binary classifier $\mathcal{H}$ adapted from the previous method to predict whether the label of a certain sample $\tilde{\mathbf{x}}_i^a$ is the target label $y_t$ or not. The input of $\mathcal{H}$ is the embedding vector $\mathbf{e}_i^a$ rather than the raw data sample $\tilde{\mathbf{x}}_i^a$ since the embedding model $f^a$ extracts more label-differentiative representations of raw data samples. $\mathcal{H}$ is trained in a semi-supervised manner based on the results of embedding swapping [46]. In each training batch, the attacker selects the top-$n$ samples with the highest prediction results given by $\mathcal{H}$ for embedding swapping, and the results of their embedding swapping can be used to fine-tune $\mathcal{H}$. We leverage the embedding swapping and the $\mathcal{H}$ output for label inference and candidate selection. Our evaluations will demonstrate that VILLAIN achieves a more accurate label inference.

### 3.1.3 Inference Adjustment

If the attacker keeps using the static $\mathbf{e}_t^a$ for embedding swapping for all candidate samples, the server may notice the unchanging embedding for different samples and be alarmed. Also the server model will be influenced. Therefore, we propose to dynamically adjust the embedding for swapping. In a certain batch, given a set of samples $\mathcal{T}$ previously inferred to belong to the target class $y_t$, we choose a subset of samples with confidence to form an inference embedding group. The embedding to swap for a candidate sample is randomly selected from the group.

## 3.2 Data Poisoning

After inferring the samples that belong to the target class, the attacker needs to poison these samples to inject the backdoor into the server model. To realize effective and stealthy backdoor injection, we develop three strategies, i.e., trigger fabrication, backdoor augmentation, and learning rate adjustment.

### 3.2.1 Trigger Fabrication

Conventional backdoor attacks usually adopt a replacement trigger, i.e., within the trigger mask area, a fixed trigger pattern replaces the original sample. Nonetheless, a replacement trigger with a fixed pattern is easy to be detected by the server. Therefore, we propose to use an additive trigger $\mathcal{E}$ to poison the embedding vector a sample as

$$\hat{\mathbf{e}}^a = f^a(\tilde{\mathbf{x}}^a) \oplus \mathcal{E}, \tag{4}$$

where $\oplus$ denotes element-wise addition. Note that we directly add the trigger to the embedding vector instead of the raw data sample since the attacker only has to upload the embedding vector to the server.

The trigger $\mathcal{E}$ is formed as

$$\mathcal{E} = \mathcal{M} \otimes (\beta \cdot \Delta), \tag{5}$$

where $\mathcal{M}$ is the trigger mask that has value 1 at the trigger area and value 0 at other areas, $\otimes$ is the element-wise multiplication, $\beta$ is a parameter that controls the trigger magnitude, and $\Delta = [+\delta, +\delta, -\delta, -\delta, \cdots, +\delta, +\delta, -\delta, -\delta]$ (a pattern of two positive values followed by two negative values). We design the trigger based on the strips based backdoor (SIG) [6] to improve the backdoor effectiveness. $\delta$ is the average standard deviation of elements in the backdoor dimension of all samples. Let $m$ denote the number of 1s in the trigger mask $\mathcal{M}$. We choose the $m$ elements in the embedding vector with the highest standard deviation as the trigger area mainly due to two reasons. It is harder for the server to detect the added trigger on elements with a larger standard deviation.

### 3.2.2 Backdoor Augmentation

The backdoor injection in vertical split learning is more difficult than the normal backdoor because the attacker can not control the embedding updates from other benign participants. To augment the backdoor injection, we introduce randomness to poisoned data during training. Therefore, we develop two randomization strategies for backdoor augmentation.

*Dropout.* Backdoor attacks, as a special kind of feature learning, are also possible to suffer from overfitting [19] [53]. Inspired by the dropout method commonly used to mitigate the overfitting problem [58], we randomly set a small number of 1s to 0s in the trigger mask for data poisoning.

*Shifting.* We randomly multiply the trigger mask by $\gamma$ that is uniformly distributed in the range $[\overline{\gamma}, \underline{\gamma}]$ to slightly shift the trigger magnitude. Shifting introduces randomness into the trigger, improving the trigger robustness and generalizability [30] [50]. The trigger shifting may also help evade trigger detection in backdoor defenses [14] [37].

Note that we do not apply the randomization strategies to the trigger during the inference phase.

### 3.2.3 Learning Rate Adjustment

In vertical split learning, the attack power of the malicious participant is likely to be diluted by other benign participants. To cope with this problem, we propose to increase the learning rate of the embedding model owned by the attacker to enhance the influence of the attack model to the final classification result during the normal training phase before inference and backdoor attack. Note that the malicious participant only adjusts its own learning rate but not the learning rate of the server model. Learning rate adjustment is valid in our attack since the participant has full control over the local training process. Although the server may assign learning rates to participants in vertical split learning, it is difficult for the server to check whether a participant follows the designated learning rate during the local training or not. The learning rate is adjusted to the smaller value in the attack phase. Learning rate adjustment has been adopted by existing works for poisoning attacks in federated learning [5] [16]. If there are more than one attackers, they may collude to improve the overall attack performance. For example, the attackers can cooperatively

Table 1: Attack performance of VILLAIN compared with baselines.

| DS[†] | Metric | ExPLoit repl. tgr. | ExPLoit add. tgr. | pasv. Fu repl. tgr. | pasv. Fu add. tgr. | act. Fu repl. tgr. | act. Fu add. tgr. | ES repl. tgr. | VILLAIN [‡] |
|---|---|---|---|---|---|---|---|---|---|
| MN | ASR | 16.51 ± 5.14% | 18.43 ± 4.50% | 98.02 ± 2.21% | 100.00 ± 0.00% | 97.66 ± 3.57% | 99.94 ± 0.13% | 96.53 ± 5.11% | **100.00 ± 0.00%** |
| | CDA | 96.10 ± 0.22% | 95.73 ± 0.16% | 95.99 ± 0.19% | 96.14 ± 0.08% | 96.01 ± 0.12% | **96.18 ± 0.07%** | 95.47 ± 0.33% | 96.11 ± 0.22% |
| | LIA | 12.48 ± 0.73% | 12.48 ± 0.73% | 89.39 ± 6.99% | 89.39 ± 6.99% | 93.70 ± 4.48% | 93.70 ± 4.48% | 94.03 ± 2.56% | **94.03 ± 2.56%** |
| CF | ASR | 8.26 ± 2.02% | 16.93 ± 3.76% | 13.61 ± 0.86% | 78.99 ± 6.23% | 14.45 ± 1.44% | 84.96 ± 8.28% | 23.66 ± 6.48% | **98.68 ± 0.59%** |
| | CDA | 76.66 ± 0.38% | 75.94 ± 0.36% | 76.75 ± 0.27% | 76.96 ± 0.35% | **76.90 ± 0.14%** | 77.09 ± 0.38% | 76.49 ± 0.40% | 76.87 ± 0.25% |
| | LIA | 18.96 ± 2.19% | 18.96 ± 2.19% | 68.12 ± 6.09% | 68.12 ± 6.09% | 76.35 ± 5.26% | 76.35 ± 5.26% | 96.08 ± 4.28% | **96.08 ± 4.28%** |
| IN | ASR | 13.94 ± 4.8% | 12.55 ± 1.79% | 26.73 ± 2.73% | 76.03 ± 9.59% | 27.71 ± 2.44% | 79.48 ± 6.09% | 32.39 ± 12.26% | **92.79 ± 1.58%** |
| | CDA | 71.21 ± 0.39% | 70.82 ± 0.93% | 70.55 ± 0.18% | 70.08 ± 0.22% | 70.91 ± 0.50% | 70.19 ± 0.74% | **71.64 ± 0.89%** | 71.54 ± 0.98% |
| | LIA | 14.53 ± 1.70% | 14.53 ± 1.70% | 80.28 ± 8.94% | 80.28 ± 8.94% | 86.54 ± 6.68% | 86.54 ± 6.68% | 90.41 ± 2.18% | **90.41 ± 2.18%** |
| CN | ASR | 5.13 ± 3.95% | 8.98 ± 4.39% | 26.63 ± 5.30% | 86.56 ± 6.45% | 33.95 ± 10.22% | 85.01 ± 15.82% | 64.56 ± 6.36% | **99.55 ± 0.62%** |
| | CDA | 61.90 ± 0.28% | 61.64 ± 0.48% | 62.65 ± 0.17% | **62.86 ± 0.08%** | 62.68 ± 0.31% | 62.72 ± 0.47% | 62.67 ± 0.08% | 62.78 ± 0.11% |
| | LIA | 12.55 ± 1.91% | 12.55 ± 1.91% | 66.83 ± 8.01% | 66.83 ± 8.01 % | 72.09 ± 7.26% | 72.09 ± 7.26% | 93.19 ± 3.95% | **93.19 ± 3.95%** |
| BM | ASR | 9.15 ± 3.90% | 14.38 ± 1.93% | 40.19 ± 4.31% | 90.28 ± 10.19% | 39.46 ± 2.53% | 86.79 ± 10.56% | 59.43 ± 12.10% | **97.84 ± 2.57%** |
| | CDA | 91.36 ± 0.77% | 90.37 ± 0.51% | 92.11 ± 0.94% | 91.22 ± 2.71% | **92.79 ± 0.25%** | 88.83 ± 2.55% | 91.80 ± 1.46% | 90.00 ± 2.34% |
| | LIA | 46.18 ± 2.39% | 46.18 ± 2.39% | 92.11 ± 4.49% | 92.11 ± 4.49% | 88.78 ± 4.64% | 88.78 ± 4.64% | 94.05 ± 4.82% | **94.05 ± 4.82%** |
| GM | ASR | 12.01 ± 3.54% | 17.87 ± 5.83% | 67.69 ± 1.04% | 100.00 ± 0.00% | 67.43 ± 1.22% | 100.00 ± 0.00% | 92.27 ± 15.41% | **100.00 ± 0.00%** |
| | CDA | 78.02 ± 0.77% | 77.81 ± 0.42% | 78.55 ± 0.24% | 78.41 ± 0.06% | 78.53 ± 0.20% | 78.32 ± 0.24% | **78.68 ± 0.09%** | 78.37 ± 0.14% |
| | LIA | 55.78 ± 2.33% | 55.78 ± 2.33% | 77.66 ± 0.72% | 77.66 ± 0.72% | 77.52 ± 0.60% | 77.52 ± 0.60% | 95.18 ± 5.69% | **95.18 ± 5.69%** |

[†] MN: MNIST, CF: CIFAR-10, IN: ImageNette, CN: CINIC-10.

[‡] ASR: attack success rate. CDA: clean data accuracy. LIA: label inference accuracy. ExPLoit: label inference adapted from [28]. pasv. Fu: the passive label inference method in Fu [16]. act. Fu: the active label inference method in Fu [16]. repl. tgr.: replacement trigger without backdoor augmentation. add. tgr.: additive trigger with backdoor augmentation. ES: embedding swapping.

conduct label inference based on the joint gradient information. More advanced collusion strategies for backdoor attacks in vertical split learning will be our future directions.

## 4 Evaluations

### 4.1 Experiment Setup

**Dataset**. We evaluate VILLAIN on six datasets with different neural network architectures. Specifically, we choose four image datasets (unstructured datasets), i.e., MNIST [32], CIFAR-10 [31], CINIC-10 [12], ImageNette [23], and two tabular datasets (structured datasets), i.e., Bank Marketing (BM) [41], Give-Me-Some-Credit (GM) [2].

**Vertical split learning settings**. Our default experiment setting consists of two participants, each owning half of the feature vector. For MNIST, the embedding model of each participant is a 4-layer fully-connected neural network, and the server model is a 3-layer fully-connected neural network. The accuracy on the test data is 96.11%. For CIFAR-10, the embedding model of each participant is a VGG16 network, and the server model is a 3-layer fully-connected neural network. The accuracy on the test data is 76.87%. For CINIC-10, the embedding model of each participant is a VGG16 network, and the server model is a 3-layer fully-connected neural network. The accuracy on the test data is 62.78%. For ImageNette, the embedding model of each participant is a VGG16 network, and the server model is a 3-layer fully-connected neural network. The accuracy on the test data is 71.54%. For BM and GM, the embedding model of each participant is a 4-layer fully-connected neural network, and the server model is a 3-layer fully-connected neural network. The accuracy on the test data is 90.00% on BM and 78.37% on GM.

Table 2: Potential side-effects. ori. acc.: the clean data accuracy without the attack. CDA.L: clean data accuracy with only the label inference module. CDA.B: clean data accuracy with the entire attack.

| Dataset | ori. acc. | CDA.L | CDA.B |
|---|---|---|---|
| **MNIST** | 94.82 ± 0.19% | 94.74 ± 0.28% | 95.08 ± 0.24% |
| **CIFAR-10** | 79.26 ± 0.17% | 78.10 ± 0.15% | 77.74 ± 0.57% |
| **ImageNette** | 66.92 ± 4.35% | 69.46 ± 2.60% | 70.54 ± 2.93% |
| **CINIC-10** | 63.07 ± 0.25% | 59.35 ± 2.92% | 62.78 ± 1.74% |
| **BM** | 89.78 ± 0.28% | 90.98 ± 1.04% | 91.80 ± 1.36% |
| **GM** | 77.22 ± 0.04% | 77.69 ± 0.03% | 78.36 ± 0.18% |

**Attack settings**. We validate our attack in the two-participant scenario as default. Different from horizontal federated learning that is usually used in the crowdsourcing scenario with thousands of participants [3] [67], vertical split learning is usually used by a few big companies with complementary features of large datasets with the same sample IDs [8]. We follow the experiment settings in existing works on vertical federated learning with two participants as default [16] [54]. The default embedding aggregation method in the server model is embedding concatenation. We also test different embedding aggregation methods in Section 4.2. The default poisoning rate is 1% on each dataset. For MNIST, CIFAR-10, ImageNette, CINIC-10, BM and GM, the number of each training batch is 128, 128, 50, 64, 100, 1,000, respectively, and the number selected for embedding swapping is $n = 14, 14, 10, 8, 6, 40$, respectively. The threshold $\mu$ in label

Table 3: Attack on different aggregation methods.

| DS | M[†] | ori. acc. | LIA | ASR | CDA |
|---|---|---|---|---|---|
| **MN** | C | 95.82 ± 0.29% | 94.03 ± 2.56% | 100.00 ± 0.00% | 96.11 ± 0.22% |
| | A | 96.69 ± 0.35% | 99.00 ± 0.19% | 100.00 ± 0.00% | 95.97 ± 0.27% |
| | M1 | 95.97 ± 0.38% | 89.48 ± 2.99% | 100.00 ± 0.00% | 95.13 ± 0.30% |
| | M2 | 95.61 ± 0.69% | 94.05 ± 3.65% | 100.00 ± 0.00% | 94.56 ± 0.48% |
| | M3 | 96.11 ± 0.16% | 99.51 ± 0.17% | 95.22 ± 1.13% | 95.59 ± 0.37% |
| **CF-10** | C | 78.29 ± 0.42% | 96.08 ± 4.28% | 98.68 ± 0.59% | 76.87 ± 0.25% |
| | A | 78.79 ± 0.22% | 99.85 ± 0.22% | 94.55 ± 0.28% | 79.90 ± 0.58% |
| | M1 | 77.83 ± 0.27% | 99.86 ± 0.32% | 94.85 ± 0.51% | 79.17 ± 0.18% |
| | M2 | 76.44 ± 0.37% | 99.98 ± 0.02% | 91.33 ± 0.48% | 78.09 ± 0.70% |
| | M3 | 76.94 ± 0.05% | 99.29 ± 0.44% | 82.98 ± 3.81% | 78.54 ± 0.10% |
| **IN** | C | 71.59 ± 0.84% | 90.41 ± 2.18% | 92.79 ± 1.58% | 71.54 ± 0.98% |
| | A | 71.93 ± 1.06% | 88.56 ± 2.63% | 100.00 ± 0.00% | 68.84 ± 0.74% |
| | M1 | 59.99 ± 1.94% | 82.30 ± 4.48% | 99.29 ± 0.12% | 56.64 ± 3.57% |
| | M2 | 66.95 ± 1.44% | 84.30 ± 2.31% | 100.00 ± 0.00% | 64.56 ± 0.79% |
| | M3 | 65.59 ± 1.57% | 86.69 ± 3.74% | 100.00 ± 0.00% | 63.49 ± 1.30% |
| **CN** | C | 62.10 ± 0.08% | 93.19 ± 3.95% | 99.55 ± 0.62% | 62.78 ± 0.11% |
| | A | 63.36 ± 1.37% | 94.97 ± 4.22% | 95.84 ± 3.82% | 62.81 ± 1.59% |
| | M1 | 63.19 ± 0.27% | 88.61 ± 2.90% | 96.81 ± 2.27% | 61.76 ± 0.23% |
| | M2 | 60.16 ± 1.51% | 85.18 ± 3.07% | 94.43 ± 6.10% | 62.83 ± 0.59% |
| | M3 | 63.29 ± 0.37% | 88.47 ± 3.58% | 96.81 ± 2.53% | 64.11 ± 0.20% |
| **BM** | C | 90.98 ± 0.52 % | 94.05 ± 4.82% | 97.84 ± 2.57% | 90.57 ± 2.14% |
| | A | 90.35 ± 0.36% | 99.58 ± 0.37% | 92.50 ± 5.83% | 90.83 ± 0.28% |
| | M1 | 92.68 ± 0.78% | 99.89 ± 0.10% | 70.68 ± 8.54% | 92.70 ± 0.81% |
| | M2 | 92.31 ± 0.35% | 99.80 ± 0.12% | 92.45 ± 3.61% | 90.15 ± 0.96% |
| | M3 | 91.94 ± 0.56% | 99.90 ± 0.11% | 84.32 ± 5.31% | 90.31 ± 0.53% |
| **GM** | C | 78.91 ± 0.28% | 95.18 ± 5.69% | 100.00 ± 0.00% | 78.37 ± 0.14% |
| | A | 75.04 ± 0.30% | 84.64 ± 6.17% | 96.10 ± 1.70% | 77.96 ± 0.25% |
| | M1 | 76.80 ± 0.36% | 93.13 ± 4.51% | 98.37 ± 0.52% | 77.04 ± 0.58% |
| | M2 | 77.39 ± 0.28% | 95.70 ± 6.98% | 96.17 ± 1.24% | 77.20 ± 0.32% |
| | M3 | 77.54 ± 0.55% | 95.27 ± 6.13% | 97.99 ± 1.49% | 76.69 ± 0.45% |

[†] We evaluate five different vertical split learning aggregation methods. C: CON, embedding concatenation. A: ADD, element-wise addition. M1: MEAN, element-wise average. M2: MAX, element-wise maximum. M3: MIN, element-wise minimum.

Table 4: Data-domain triggers. TS: Trigger Size.

| DS | TS | ASR | CDA | ori. acc. | DS | TS | ASR | CDA | ori. acc. |
|---|---|---|---|---|---|---|---|---|---|
| **MN** | 2 | 92.04% | 96.72% | 94.66% | **CF** | 2 | 95.36% | 78.82% | 76.78% |
| | 3 | 99.92% | 96.65% | 94.71% | | 3 | 99.70% | 78.95% | 76.58% |
| | 4 | 99.97% | 96.79% | 94.40% | | 4 | 98.53% | 79.31% | 75.65% |
| | 5 | 99.94% | 96.80% | 94.57% | | 5 | 99.27% | 79.43% | 76.75% |
| | 6 | 99.99% | 96.63% | 94.99% | | 6 | 99.55% | 79.27% | 77.76% |
| **IM** | 14 | 41.69% | 74.19% | 73.06% | **CN** | 2 | 46.60% | 63.43% | 61.00% |
| | 21 | 51.11% | 74.51% | 70.45% | | 3 | 98.59% | 63.84% | 62.26% |
| | 28 | 77.58% | 74.87% | 70.05% | | 4 | 96.85% | 64.12% | 62.74% |
| | 35 | 90.11% | 75.25% | 72.53% | | 5 | 99.17% | 64.01% | 62.11% |
| | 42 | 98.66% | 74.37% | 71.47% | | 6 | 96.92% | 63.87% | 62.16% |
| **BM** | 1 | 98.69% | 92.40% | 90.18% | **GM** | 1 | 100.00% | 78.52% | 77.82% |
| | 2 | 97.79% | 92.76% | 88.25% | | 2 | 100.00% | 78.76% | 77.82% |
| | 3 | 99.74% | 93.28% | 90.33% | | 3 | 100.00% | 78.76% | 77.73% |
| | 4 | 99.35% | 92.89% | 86.23% | | 4 | 100.00% | 78.54% | 77.65% |
| | 5 | 99.80% | 93.12% | 90.72% | | 5 | 100.00% | 78.73% | 77.80% |

inference model is set based on the average value of the gradient L2 norm value. The default value of $\beta$ is 0.4. We set $\bar{\gamma}$ to 0.6 and $\underline{\gamma}$ to 1.2. We perform attack on all the labels to test the overall effectiveness. The dropout ratio we use in the experiment is 0.75. We use *Attack success rate* (ASR) and *Clean data rate* (CDA) to evaluate the performance of backdoor attacks [17]. We also use *Label inference accuracy* (LIA) to evaluate the performance of the label inference module. LIA measures the percentage of correctly-inferred samples in all the samples we classified as the target label, since we only use the target label data for the backdoor attack.

**Baselines**. As far as we know, there are no backdoor attacks against vertical split learning. Therefore, we compare the attack performance of VILLAIN with a baseline that adopts a traditional replacement trigger [1, -1, 1, -1, 1]. We compare the label inference module of VILLAIN with the passive and the active label inference methods in Fu [16]. We also compare our work with ExPLoit [28]. ExPLoit considers a two-party split learning scenario where the participant owns all features and the server owns all labels. The participant learns an embedding model based on all features, while the server learns a classification model based on embeddings. To conduct a label inference attack, ExPLoit enables the malicious participant to optimize a surrogate model with similar functionalities of the server's classification model. We adapt ExPLoit to our threat model where the malicious participant only owns partial features.

The experiments are carried out on our workstations equipped with Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz, and NVIDIA GTX 3090 GPU cards running Ubuntu 18.04 system.

## 4.2 Overall Performance

We run each experiment five times for the overall performance and record the mean and standard deviation. As shown in Table 1, VILLAIN outperforms the baseline backdoor attack with significant improvement. The clean data accuracy of VILLAIN is less than 1% lower than the benign model. As shown in Table 1, our label inference algorithm achieves the highest precision on each dataset. Our proposed label inference strategy outperforms baselines [16] by as high as 20%. The attack performance of ExPLoit [28] is not ideal under our threat model. The possible reason is that ExPLoit requires all features to reconstruct a substitute server model. Under our threat model with only partial features, the server model cannot be restored with high reliability, thus the label inference attack of ExPLoit is less effective. The main task accuracy of the vertical split learning is lower than that of centralized models since model split negatively affects the overall learning ability. We evaluate the runtime efficiency of the label inference. The ExPLoit takes much more time since the optimization is time-consuming. VILLAIN without candidate selection has a much lower time efficiency and takes the shortest time for label inference than baselines. For backdoor attack, it takes 4, 5, 4, 2, 6, 9 epochs to establish the stable backdoor into the server model for MNIST, CIFAR-10, ImageNette, CINIC-10, BM, and GM respectively. Our label inference results are impressive mainly because we have made use of the gradient information, which has been ignored in existing papers. The gradients indicate the training loss on the training sample, which enables us to design embedding swapping for label inference. Moreover, our proposed candidate selection further improves inference efficiency by sifting the most promising candidate for label inference.

Table 5: Multi-participant scenario. # pa.: number of participants. ori. acc.: original model accuracy.

| DS | # pa. | ori. acc. | LIA | ASR | CDA |
|---|---|---|---|---|---|
| MN | 2 | 95.82 ± 0.29% | 94.03 ± 2.56% | 100.00 ± 0.00% | 96.11 ± 0.22% |
| | 4 | 92.87 ± 0.24% | 94.71 ± 4.71% | 100.00 ± 0.00% | 93.72 ± 0.49% |
| | 8 | 93.18 ± 0.17% | 90.07 ± 3.15% | 94.86 ± 1.04% | 92.87 ± 0.36% |
| | 16 | 90.49 ± 0.29% | 81.59 ± 5.82% | 87.16 ± 7.76% | 89.65 ± 0.15% |
| CF | 2 | 78.29 ± 0.42% | 96.08 ± 4.28% | 98.68 ± 0.59% | 76.87 ± 0.25% |
| | 4 | 70.61 ± 0.33% | 91.17 ± 4.90% | 98.57 ± 0.69% | 69.39 ± 0.89% |
| | 8 | 67.38 ± 0.48% | 72.31 ± 6.75% | 65.59 ± 4.68% | 68.76 ± 0.15% |
| | 16 | 65.56 ± 0.45% | 65.39 ± 9.45% | 23.57 ± 6.03% | 66.69 ± 0.25% |
| IN | 2 | 71.59 ± 0.84% | 90.41 ± 2.18% | 92.79 ± 1.58% | 71.54 ± 0.98% |
| | 4 | 67.58 ± 0.29% | 84.20 ± 3.85% | 80.64 ± 2.76% | 68.68 ± 0.70% |
| | 8 | 66.73 ± 0.31% | 82.28 ± 6.08% | 62.19 ± 6.01% | 64.17 ± 0.96% |
| | 16 | 58.15 ± 0.57% | 73.68 ± 5.15% | 36.89 ± 10.81% | 60.82 ± 0.44% |
| CN | 2 | 62.10 ± 0.08% | 93.19 ± 3.95% | 99.55 ± 0.62% | 62.78 ± 0.11% |
| | 4 | 59.19 ± 0.28% | 90.62 ± 3.89% | 84.98 ± 3.17% | 61.87 ± 1.33% |
| | 8 | 58.44 ± 0.26% | 70.69 ± 2.09% | 59.12 ± 8.79% | 58.83 ± 0.67% |
| | 16 | 53.90 ± 1.06% | 62.96 ± 9.51% | 37.32 ± 7.48% | 54.43 ± 0.79% |
| BM | 2 | 90.98 ± 0.52 % | 94.05 ± 4.82% | 97.84 ± 2.57% | 90.57 ± 2.14% |
| | 3 | 90.71 ± 0.61% | 92.26 ± 4.52% | 92.26 ± 3.64% | 91.46 ± 0.29% |
| | 4 | 90.74 ± 0.51% | 87.29 ± 6.48% | 91.26 ± 4.09% | 91.45 ± 0.92% |
| | 6 | 91.18 ± 0.13% | 82.93 ± 8.54% | 76.91 ± 6.55% | 92.59 ± 0.44% |
| GM | 2 | 78.91 ± 0.28% | 95.18 ± 5.69% | 100.00 ± 0.00% | 78.37 ± 0.14% |
| | 3 | 79.22 ± 0.84% | 93.17 ± 8.50% | 92.14 ± 2.58% | 77.64 ± 0.17% |
| | 4 | 76.04 ± 0.18% | 96.58 ± 6.59% | 89.55 ± 6.87% | 78.42 ± 0.61% |
| | 6 | 77.24 ± 0.30% | 78.57 ± 9.62% | 66.17 ± 5.58% | 78.27 ± 0.28% |
| AM | 2 | 97.05 ± 0.57% | 96.93 ± 3.09% | 98.14 ± 1.07% | 97.46 ± 0.72% |
| | 4 | 81.27 ± 0.63% | 93.27 ± 5.24% | 95.14 ± 2.82% | 80.73 ± 0.90% |
| | 8 | 71.29 ± 0.23% | 87.58 ± 11.42% | 88.53 ± 5.04% | 73.36 ± 0.55% |
| | 16 | 68.09 ± 0.42% | 82.57 ± 5.71% | 69.17 ± 4.65% | 67.68 ± 0.28% |

Table 6: Impact of server models. dep.: model depth.

| dep. | MNIST | | CIFAR-10 | |
|---|---|---|---|---|
| | LIA | ASR | LIA | ASR |
| 3 | 94.03 ± 2.56% | 100.00 ± 0.00% | 96.08 ± 4.28% | 98.68 ± 0.59% |
| 4 | 95.89 ± 2.95% | 100.00 ± 0.00% | 96.63 ± 3.55% | 96.97 ± 0.45% |
| 5 | 94.92 ± 2.63% | 99.53 ± 0.24% | 97.55 ± 3.97% | 96.83 ± 0.24% |
| 6 | 92.85 ± 4.10% | 100.00 ± 0.00% | 97.06 ± 1.73% | 98.03 ± 0.58% |
| 7 | 95.73 ± 2.66% | 100.00 ± 0.00% | 98.53 ± 2.66% | 97.86 ± 0.13% |
| dep. | CINIC-10 | | BM | |
| | LIA | ASR | LIA | ASR |
| 3 | 93.19 ± 3.05% | 99.55 ± 0.62% | 94.05 ± 4.82% | 97.84 ± 2.57% |
| 4 | 94.10 ± 2.56% | 97.27 ± 1.43% | 95.03 ± 5.93% | 96.91 ± 0.92% |
| 5 | 93.68 ± 1.41% | 98.03 ± 0.20% | 98.23 ± 0.96% | 98.35 ± 0.47% |
| 6 | 96.14 ± 3.02% | 95.82 ± 3.94% | 94.76 ± 2.59% | 92.47 ± 1.69% |
| 7 | 95.16 ± 3.97% | 96.29 ± 3.46% | 95.91 ± 2.49% | 95.10 ± 0.82% |
| dep. | ImageNette | | GM | |
| | LIA | ASR | LIA | ASR |
| 3 | 90.41 ± 2.18% | 92.79 ± 1.58% | 95.18 ± 5.69% | 100.00 ± 0.00% |
| 4 | 92.14 ± 3.06% | 93.01 ± 1.65% | 98.62 ± 0.63% | 100.00 ± 0.00% |
| 5 | 95.52 ± 3.45% | 96.68 ± 0.94% | 96.28 ± 3.10% | 99.35 ± 0.20% |
| 6 | 87.05 ± 7.49% | 90.93 ± 3.69% | 93.60 ± 4.60% | 100.00 ± 0.00% |
| 7 | 94.11 ± 2.46% | 92.04 ± 0.75% | 94.04 ± 3.63% | 98.80 ± 0.94% |

Table 7: Impact of participant models.

| Model | MNIST | | BM | | GM | |
|---|---|---|---|---|---|---|
| | LIA | ASR | LIA | ASR | LIA | ASR |
| FCNN-4 | 94.03 ± 2.56% | 100.00 ± 0.00% | 94.05 ± 4.82% | 97.84 ± 2.57% | 95.18 ± 5.69% | 100.00 ± 0.00% |
| FCNN-5 | 95.23 ± 3.09% | 99.51 ± 0.31% | 96.47 ± 2.85% | 98.75 ± 0.48% | 94.33 ± 4.60% | 100.00 ± 0.00% |
| FCNN-6 | 97.33 ± 2.40% | 96.50 ± 2.67% | 95.48 ± 3.54% | 96.34 ± 1.93% | 95.05 ± 3.63% | 100.00 ± 0.00% |
| FCNN-7 | 94.49 ± 1.87% | 98.87 ± 0.30% | 98.80 ± 0.84% | 97.02 ± 1.66% | 97.12 ± 2.40% | 100.00 ± 0.00% |
| FCNN-8 | 96.91 ± 2.84% | 99.17 ± 0.65% | 96.03 ± 3.44% | 98.17 ± 0.64% | 93.15 ± 3.56% | 98.81 ± 0.53% |
| Model | CIFAR-10 | | ImageNette | | CINIC-10 | |
| | LIA | ASR | LIA | ASR | LIA | ASR |
| CNN | 94.14 ± 3.10% | 95.30 ± 0.69% | 92.77 ± 3.29% | 93.16 ± 0.84% | 94.86 ± 2.82% | 98.83 ± 0.57% |
| VGG-16 | 96.08 ± 4.28% | 98.68 ± 0.59% | 90.41 ± 2.18% | 92.79 ± 1.58% | 93.19 ± 3.95% | 99.55 ± 0.62% |
| VGG-19 | 95.23 ± 2.68% | 97.20 ± 0.82% | 95.41 ± 2.43% | 95.25 ± 1.65% | 95.25 ± 1.49% | 98.32 ± 1.35% |
| ResNet-50 | 97.91 ± 2.44% | 95.02 ± 1.24% | 92.11 ± 3.58% | 94.65 ± 2.26% | 96.53 ± 1.71% | 97.25 ± 1.08% |
| DenseNet | 97.72 ± 3.34% | 98.24 ± 0.97% | 97.68 ± 5.65% | 92.89 ± 2.15% | 95.89 ± 3.38% | 98.73 ± 0.86% |

**Potential side-effects**. The embedding swapping of label inference module and the data poisoning module may both have side-effects on learning the central model. We evaluate the clean data accuracy without any attack, with only label inference, and with both label inference and data poisoning. As shown in Table 2, the label inference only slightly downgrades the main task accuracy. This is because the candidate selection process greatly reduces the number of samples for embedding swapping. The low poisoning rate ensures that the backdoor does not degrade clean data accuracy.

**Different embedding aggregation methods**. We evaluate VILLAIN on different vertical split learning aggregation methods on the server model that adopt addition, average pooling, max pooling, and min pooling for embedding aggregation. The default embedding aggregation is embedding concatenation. The server splices the embedding from different clients into one embedding. The element-wise addition vertical split learning method adds the embedding from different participants together to form the input for the server model. The element-wise average pooling vertical split learning mode conducts average pooling on each element of the embedding vector from the participants. The max/min pooling vertical split learning methods choose the maximum/minimum value of each element of the embedding vector from the participants. As shown in Table 3, VILLAIN performs well on different vertical split learning methods.

**Data-domain triggers**. In VILLAIN, the trigger can be added in the data domain or the embedding domain. We test the performance of VILLAIN with data-domain triggers. As shown in Table 4, the ASR is higher than 98% for MNIST, CIFAR-10, CINIC-10, BM, and GM at a trigger size of 3, which proves that the data-domain trigger is effective. The results also show that the CDA is hardly influenced by the trigger in the data domain.

**Multi-participant scenario**. VILLAIN can be extended to vertical split learning with more than 2 participants. We conduct experiments on MNIST, CIFAR-10, ImageNette, and CINIC-10 with 2, 4, 8, and 16 participants. We conduct experiments on BM and GM with 2, 3, 4, and 6 participants since their feature dimensions are small. Since public tabular datasets usually have limited numbers of samples and feature dimensions, we construct a simulated tabular dataset, AM, consisting of 60,000 samples with a feature dimension of 1,024. The samples belong to 10 classes, each with a relatively balanced number of samples. We generate the value of the $i$-th feature of a sample that belongs to class $j$ as $v_{i,j} = \overline{v}_{i,j} + \sigma$, where $\overline{v}_{i,j}$ is the average value of the corresponding feature according to real financial datasets BM and GM. $\sigma$ is a random adaptation factor generated by an auto-encoder [22] [49]. The data is equally divided among all participants and we choose the middle participant as the attacker. As shown in Table 5, the attack success rate remains high when the number of

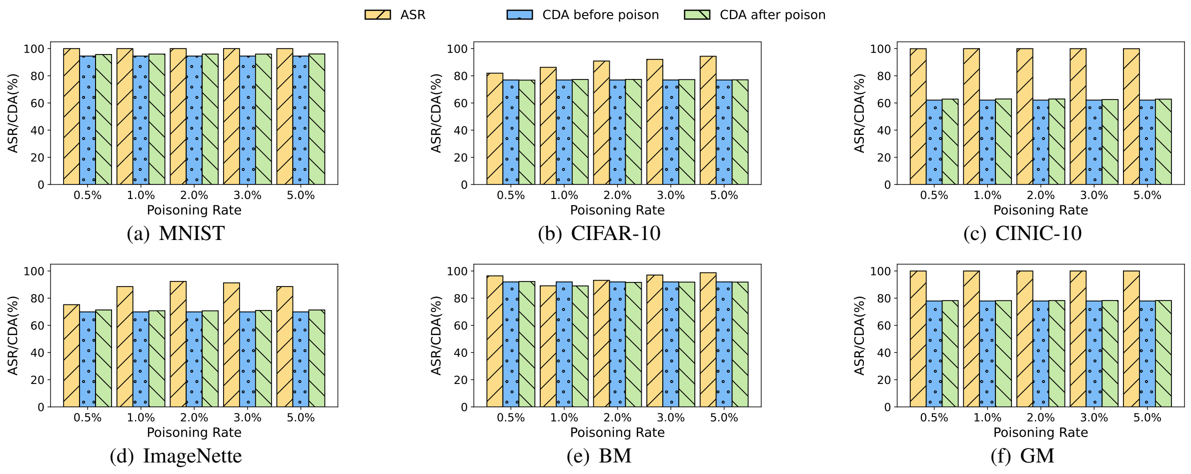


Figure 4: Impact of poisoning rate.

participants is no more than 8. The attack success rate drops noticeably when the number of participants is 16, which is reasonable since the attacker only controls 1/16 of the embedding vector uploaded to the server. When the number of participants increases, the clean data accuracy drops due to data splitting. For the tabular dataset, the attack is still strong for the 6-client scenario. The ASR of BM drops because of the limited control of the overall embedding. We conduct the attack with 2, 4, 8, and 16 participants on AM. As shown in Table 5, when there are as many as 16 participants, the LIA is above 80%, and the ASR is above 70%, which proves the effectiveness of VILLAIN under multiple participants.

We also conduct ablation studies to examine the contribution of the *candidate selection*, the *trigger fabrication* and the *backdoor augmentation* modules to the attack performance. We show the results in the Appendix A.1.

## 4.3 Impact of Hyperparameters

**Impact of poisoning rate.** The poisoning rate is defined as the number of poisoned embeddings to all embeddings submitted by the malicious participant to the server. Achieving a high attack success rate with a low poisoning rate is desirable since the clean data accuracy will be less affected and the attack is more stealthy. As shown in Figure 4, a higher poisoning rate brings a higher attack success rate because the model is more likely to learn the backdoor pattern from the poisoned data. The backdoor attack still works even with a low poisoning rate of only 0.5%.

**Impact of trigger magnitude.** We vary the magnitude of the trigger $\beta$ from 0.2 to 1. As shown in Figure 10 in the appendix, a larger trigger magnitude improves the attack success rate, especially for more complex datasets like CIFAR-10 and ImageNette. The trigger magnitude hardly changes the prediction accuracy on clean samples.

**Impact of server & participant models.** For real-world attack scenarios, the attacker has no knowledge of the server model. We change the number of layers of the server model to evaluate the attack performance. We set the target label to 0 for each dataset. As shown in Table 6, we highlight the best and the worst results in each dataset. The results show that VILLAIN is robust to different server structures overall. We also conduct experiments on participants with different models. For small datasets like MNIST, BM, and GM, we use fully connected neural network as the participant model and change the model layer from 4 to 5, 6, 7, 8. For CIFAR-10, ImageNette, and CINIC-10, we use CNN, VGG-16, VGG-19, ResNet-50 [21], DenseNet [24] as the participant model. The target label is 0 for each dataset. As shown in Table 7, the label inference and backdoor attack are hardly influenced by the participant model structures.

**Impact of trigger size.** Since the output dimension of the embedding model of the attacker is 64 in our experiments, we vary the trigger length to 4, 8, 16, 32, and 64 to evaluate the attack performance. As shown in Figure 8 in the appendix, VILLAIN maintains a high attack success rate with a trigger length of no less than 16 on all datasets. The experiment results also show that the clean data accuracy is hardly influenced by the trigger size. The backdoor randomization is effective when the trigger sizes are large.

**Impact of learning rate.** We set the learning rate of the attacker model to be 0.01, 0.03, 0.05, 0.08, 0.1, 0.15, and 0.2 during the normal training phase. As demonstrated in Figure 9, the learning rate shows impact on the ASR. We also conduct evaluation of cases when the learning rate provided by the server is already large and the attacker attempts to further increase the learning rate. As shown in Figure 16, further increasing a large learning rate may neither improve nor harm the attack performance.

**Impact of number of candidates.** We vary the number of candidates selected for label inference in each batch. As shown in Figure 11 in the appendix, for MNIST, CIFAR-10,

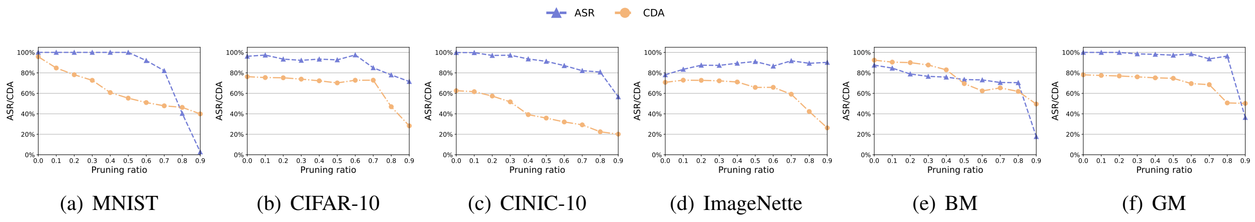


Figure 5: Backdoor attack against defense with pruning.

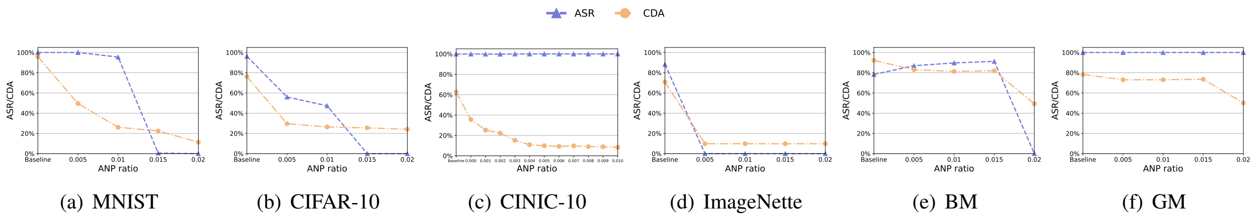


Figure 6: Backdoor attack against defense with ANP.

CINIC-10, BM, and GM, the number of selected candidate samples per batch has little influence on the attack performance, as ASR is nearly 100% on these datasets. For ImageNette, the attack success rate goes up when we increase the number of samples for label inference per batch thanks to more inferred data of the target label. With the accurate label inference method, a larger number of candidates for label inference helps find more samples of the target label.

## 4.4 Resistance to Defense

The server may employ countermeasures to defend against VILLAIN.

### 4.4.1 Defenses Against Label Inference Attack

Since the only information we use for label inference is the back-propagated gradient from the server, we consider possible defenses that process the gradients to prevent information leakage without degrading the learning task. We utilize three possible defense methods against label inference, i.e., DP-SGD, gradient compression, and Privacy-preserving Deep Learning. For DP-SGD, the server disrupts the gradients following differential privacy [52] [64] for different privacy levels. A lower ε indicates that the privacy leakage is smaller with a larger magnitude of added noises, vice versa. For gradient compression [56], the server sends a subset of gradients with the largest absolute values to participants. The compression ratio represents the fraction of preserved elements to the dimension of gradients. We also test the label inference against Privacy-preserving Deep Learning (PPDL) [56], in which the server first adds random noises to the gradients and then only keeps a fraction of gradients. We consider the fraction of preserved gradients as the defense parameter.

*Results*. As shown in Table 9 in the appendix, under DP-SGD, our label inference attack is still effective when ε is larger than 1. If ε is smaller than 1, meaning stricter privacy protection with larger noises, the label inference accuracy drops, but the clean data accuracy also falls due to disturbed gradients. Under gradient compression, our label inference attack performs well since the gradients with large absolute values still preserve most label information. Under PPDL, the defense degrades both clean data accuracy and label inference accuracy.

### 4.4.2 Defenses Against Backdoor Attacks

We test VILLAIN under different defenses following the systemic categorization on backdoor defenses in [36].

*Model reconstruction*. Model reconstruction based defenses aim at purifying the backdoored model. Model Pruning [39] and Adversarial Neuron Pruning (ANP) [63] intend to remove the backdoor via pruning. As shown in Figures 5, 12, and 13, VILLAIN maintains a high attack success rate when the model is pruned with a medium ratio. The defense aims at achieving a point where the ASR is low and the CDA is nearly not affected. Although the attack success rate drops with a high pruning ratio, the model's functionality is also damaged by the pruning operation. The pruning is proven to be insufficient. The impact of ANP is shown in Figure 6, 14, and 15. For MNIST, CINIC-10 and GM, the backdoor attack is hardly influenced while the model functionality on clean data is damaged. For CIFAR-10, ImageNette and BM, the backdoor attack success rate drops along with the model accuracy. Both trends prove the defense can not keep high CDA while reducing the ASR.

*Sample preprocessing*. Sample preprocessing based defenses process the samples before feeding the samples into the model, which aims to destroy the trigger pattern. We apply the state-of-the-art sample preprocessing defense, i.e., Backdoor Defense via Transformations [37], which uses transformation $T(\cdot)$ to preprocess the input. We apply two transformations, namely embedding noise, and left-right flipping. For embedding noise, we add random noise in 5 different levels to the embedding vectors. As shown in Figure 7, the ASR withstands the embedding noise while the model accuracy on clean data drops even faster. For the left-right flipping, we swap the left and the right part of the embedding from the attacker. As shown in Table 10 in the appendix, the flipping mostly influences the model accuracy on the clean data.

*Trigger synthesis*. These defenses aim to synthesize the backdoor used in the backdoor attack. We use the generalizable detection method based on the intuition that the backdoor trigger can be recovered by computing the perturbation needed for a clean sample to be misclassified into the target label. We apply the method to the server model, and the results show that VILLAIN is always below the threshold.

*Poison suppression*. These defense methods aim at depressing the effectiveness of poisoned samples during the training process to prevent backdoor injection. Anti-Backdoor Learning (ABL) [35] detects poisoned samples first and then suppresses these samples to mitigate the backdoor attack. As shown in Table 11 in the appendix, the detection accuracy is rather low because the detection is based on the assumption that poisoned samples have a lower loss value but VILLAIN in fact makes poisoned samples more learnable.

#### 4.4.3 Adaptive Defenses

We also validate VILLAIN's feasibility under two possible adaptive defenses that attempt to thwart the label inference and the data poisoning modules respectively.

*Embedding detection*. The server may adapt its defense against the label inference module of VILLAIN by detecting embedding swapping. Since we swap the embedding of training samples with the embedding of a known sample of the target label, different training samples may have the identical embedding result, which may be detected by the server as the sign of label inference attacks. VILLAIN can bypass this adaptive defense by introducing randomness into the embedding swapping process. We consider two methods, i.e., adding random noises or randomly smoothing some elements of the embedding vector. We compute the distance between two embeddings of any two different benign samples and find that the minimal distance is less than the lowest bar 0.01. The server may use the minimum distance as the detection threshold, i.e., an embedding with a smaller distance to others will be flagged as abnormal. As shown in Table 12 in the appendix, the introduced randomness does not affect the attack performance of VILLAIN, but helps evade the adaptive defense of identical embedding detection.

*Embedding smoothing*. The server may adapt its defense against the data poisoning module of VILLAIN by trying to neutralize the unknown trigger with the convolutional operation. We assume that the server uses a 1 convolutional kernel to process the embedding vector. All the elements in the convolutional kernel add up to 1. We use the pooling ratio to denote the value of the central element in the convolutional kernel. The other two elements are equal. For example, if the pooling ratio is 0.2, the convolutional kernel will be $[0.4, 0.2, 0.4]$. A large pooling ratio indicates a stronger smoothing level. As shown in Table 8, the attack success rate is relatively stable under embedding smoothing, but the clean data accuracy will be greatly reduced.

Table 8: The performance of VILLAIN under adaptive defense of embedding smoothing.

| | MNIST | | | CIFAR-10 | | | ImageNette | |
|---|---|---|---|---|---|---|---|---|
| Ratio | CDA | ASR | Ratio | CDA | ASR | Ratio | CDA | ASR |
| 1.0 | 96.24% | 100.00% | 1.0 | 76.08% | 98.43% | 1.0 | 71.80% | 92.94% |
| 0.8 | 95.62% | 100.00% | 0.8 | 75.17% | 97.62% | 0.8 | 70.85% | 89.54% |
| 0.6 | 93.72% | 95.31% | 0.6 | 73.73% | 97.46% | 0.6 | 65.91% | 90.39% |
| 0.4 | 74.95% | 92.86% | 0.4 | 71.38% | 96.80% | 0.4 | 65.31% | 87.60% |
| 0.2 | 37.89% | 91.92% | 0.2 | 68.22% | 95.93% | 0.2 | 62.12% | 85.99% |
| | CINIC-10 | | | BM | | | GM | |
| Ratio | CDA | ASR | Ratio | CDA | ASR | Ratio | CDA | ASR |
| 1.0 | 62.17% | 99.66% | 1.0 | 91.85% | 97.69% | 1.0 | 77.84% | 100.00% |
| 0.8 | 61.99% | 97.32% | 0.8 | 83.95% | 95.56% | 0.8 | 77.06% | 100.00% |
| 0.6 | 59.88% | 96.86% | 0.6 | 82.47% | 96.52% | 0.6 | 75.06% | 100.00% |
| 0.4 | 55.03% | 95.99% | 0.4 | 81.21% | 96.51% | 0.4 | 62.03% | 100.00% |
| 0.2 | 47.83% | 95.27% | 0.2 | 80.23% | 96.42% | 0.2 | 46.96% | 100.00% |

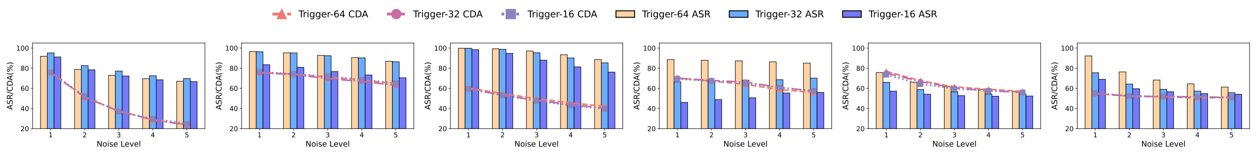
## 5 Discussions

We discuss possible extensions and defenses of VILLAIN.

*Split learning of other structures.* Split learning framework may be materialized with different structures for different tasks, e.g., graph neural networks (GNN) [11] [55], to which VILLAIN may not apply due to different forms of interactions between the participant and the server. The input to a GNN model includes both the node attributes and the link topology, which entails a special vertical data partition among participants in split learning. Backdoor attacks against GNN-based split learning is a possible future direction.

*Secure vertical split learning.* Homomorphic Encryption (HE) [43], Secure Multi-Party Computation (MPC) [29] and Trusted Execution Environment (TEE) [13] have been introduced to vertical split learning to provide guaranteed security at high computational cost or hardware requirement. Backdoor attacks under secure vertical split learning framework is also a future direction.

*Possible defenses.* Existing defenses have all been proposed regarding backdoor attacks in centralized learning or horizontal federated learning, thus failing to detect VILLAIN, as demonstrated by our experiments. Preventing backdoor attacks in vertical federated learning poses new challenges since the updates from different participants are based on different features, thus outlier detection cannot be used to figure



(a) MNIST (b) CIFAR-10 (c) CINIC-10 (d) ImageNette (e) BM (f) GM

Figure 7: Backdoor attack against defense with transformation with embedding noise.

out malicious participants. There are two possible ways to improve the security of vertical split learning. On the one hand, the server can mask gradients with more advanced techniques to mitigate label inference. On the other hand, since the malicious attacker may not conduct data poisoning in all epochs, the server may perform outlier detection based on historical updates of a participant.

## 6 Related Work

**Vertical split learning**. Vertical split learning, as a new machine learning paradigm for VFL, splits the model into segments, and the segments are distributed to the VFL participators to be trained separately. The neural network model in vertical split learning is called the Split Neural Network, or SplitNN [54]. Each data holder converts its original data into an intermediate embedding with the model segment [27]. The vertical split learning model can adopt different aggregation methods, including average, element-wise maximum, element-wise sum, element-wise multiplication, and concatenation [8]. Splitfed Learning (SFL) [59] applied previous privacy protection and model training methods to vertical split learning scenarios to improve data privacy and training efficiency. Secure computation methods like secure 2-party computation and Homomorphic Encryption can also be used in split learning [48] [66].

**Backdoor attacks**. Backdoor attack aims to inject the backdoor into victim machine learning models [36]. Attackers usually utilize data poisoning as the backdoor injection method. BadNets [20] introduces the first backdoor attack to deep neural networks by a visible backdoor trigger. To make the attack stealthy, invisible backdoor attacks [14] [45] and semantic backdoor attacks [4] [38] were proposed. All these methods above are dirty-label backdoor attacks because their poisoned samples are mislabelled by the attacker. In contrast, clean-label backdoor attacks aim to poison the target model without controlling the labeling process of poisoned data [60]. Attackers may use adversarial perturbation or image scaling to inject invisible clean-label poisoned data samples [69] [51]. Federated learning is especially vulnerable to data poisoning based backdoor attacks [18]. The attacker can embed the backdoor into the tail of the input distribution [62]. Since federated learning involves multiple participants, the attacker can also make use of a composite global trigger, which is divided among different participants and injected individually [65]. New loss functions [7] and model construction methods [15] were developed to backdoor the global model effectively. However, the above backdoor attacks all aim at horizontal federated learning.

**Privacy and security in federated learning**. There are various security threats in federated learning [42]. Researchers show that membership inference attacks can be used to trace training data records in federated learning in both black-box and white-box scenarios [44]. Attackers can infer the training data membership in a passive and an active manner in federated learning [40]. By optimizing the loss between the recovered gradients and the real gradients, the attacker can recover the training samples and the labels with the gradients [72]. For split learning, label inference attacks [34] have been proposed for the participant to infer the label based on the information transmitted between participants and the server. The label inference can also be formalized as a supervised learning problem with a loss function of gradient-matching [28]. The adversary server in [47] manages to recover the training data in the participant model by hijacking its learning process. Fu [16] provides passive and active label inference attacks by semi-supervised learning and gradient analysis.

## 7 Conclusion

In this paper, we make the first attempt to explore the security risks of backdoor attacks against vertical split learning. Our proposed attack framework, VILLAIN, overcomes the challenges of no label and no server model information in vertical split learning. More specifically, we have developed a novel label inference algorithm to locate samples of the target label. In addition, we have designed effective data poisoning strategies to strengthen the link between the trigger and the backdoor in the server model. Extensive experiments have validated the effectiveness and robustness of VILLAIN.

## 8 Acknowledgments

We sincerely thank our shepherd and all the anonymous reviewers for their valuable comments. This work was sponsored by CCF-AFSG Research Fund. Yanjiao Chen is the corresponding author.

## References

[1] FedAI. https://www.fedai.org/.

[2] Give me some credit dataset. https://www.kaggle.com/c/GiveMeSomeCredit.

[3] Google Gboard. https://apps.apple.com/us/app/gboard-the-google-keyboard/id1091700242.

[4] Eugene Bagdasaryan and Vitaly Shmatikov. Blind backdoors in deep learning models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1505–1521, 2021.

[5] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020.

[6] Mauro Barni, Kassem Kallas, and Benedetta Tondi. A new backdoor attack in cnns by training set corruption without label poisoning. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 101–105. IEEE, 2019.

[7] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, pages 634–643. PMLR, 2019.

[8] Iker Ceballos, Vivek Sharma, Eduardo Mugica, Abhishek Singh, Alberto Roman, Praneeth Vepakomma, and Ramesh Raskar. SplitNN-driven vertical partitioning. *arXiv preprint arXiv:2008.04137*, 2020.

[9] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[10] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, Dimitrios Papadopoulos, and Qiang Yang. Secureboost: a lossless federated learning framework. *IEEE Intelligent Systems*, 36(6):87–98, 2021.

[11] Tsz-Him Cheung, Weihang Dai, and Shuhan Li. Fedsgc: Federated simple graph convolution for node classification. In *International Workshop on Federated and Transfer Learning for Data Sparsity and Confidentiality in Conjuncation with IJCAI*, 2021.

[12] Luke N Darlow, Elliot J Crowley, Antreas Antoniou, and Amos J Storkey. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018.

[13] Ghada Dessouky, Tommaso Frassetto, and Ahmad-Reza Sadeghi. {HybCache}: Hybrid {Side-Channel-Resilient} caches for trusted execution environments. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 451–468, 2020.

[14] Khoa Doan, Yingjie Lao, Weijie Zhao, and Ping Li. Lira: Learnable, imperceptible and robust backdoor attacks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11966–11976, 2021.

[15] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. Local model poisoning attacks to {Byzantine-Robust} federated learning. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1605–1622, 2020.

[16] Chong Fu, Xuhong Zhang, Shouling Ji, Jinyin Chen, Jingzheng Wu, Shanqing Guo, Jun Zhou, Alex X Liu, and Ting Wang. Label inference attacks against vertical federated learning. In *USENIX Security Symposium*, 2022.

[17] Xueluan Gong, Yanjiao Chen, Jianshuo Dong, and Qian Wang. ATTEQ-NN: attention-based QoE-aware evasive backdoor attacks. In *Network and Distributed System Security Symposium*. The Internet Society, 2022.

[18] Xueluan Gong, Yanjiao Chen, Qian Wang, and Weihan Kong. Backdoor attacks and defenses in federated learning: State-of-the-art, taxonomy, and future directions. *IEEE Wireless Communications*, 2022.

[19] Kathrin Grosse, Taesung Lee, Youngja Park, Michael Backes, and Ian Molloy. A new measure for overfitting and its implications for backdooring of deep learning. 2020.

[20] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[22] Xianxu Hou, Linlin Shen, Ke Sun, and Guoping Qiu. Deep feature consistent variational autoencoder. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 1133–1141. IEEE, 2017.

[23] Jeremy Howard and Sylvain Gugger. Fastai: a layered API for deep learning. *Information*, 11(2):108, 2020.

[24] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[25] W Ronny Huang, Jonas Geiping, Liam Fowl, Gavin Taylor, and Tom Goldstein. Metapoison: practical general-purpose clean-label data poisoning. In *Advances in Neural Information Processing Systems*. PMLR, 2020.

[26] Praveen Joshi, Chandra Thapa, Seyit Camtepe, Mohammed Hasanuzzaman, Ted Scully, and Haithem Afli. Performance and information leakage in splitfed learning and multi-head split learning in healthcare data and beyond. *Methods and Protocols*, 5(4):60, 2022.

[27] Praveen Joshi, Chandra Thapa, Seyit Camtepe, Mohammed Hasanuzzamana, Ted Scully, and Haithem Afli. Splitfed learning without client-side synchronization: Analyzing client-side split network portion size to overall performance. *arXiv preprint arXiv:2109.09246*, 2021.

[28] Sanjay Kariyappa and Moinuddin K Qureshi. Exploit: Extracting private labels in split learning. In *First IEEE Conference on Secure and Trustworthy Machine Learning*, 2021.

[29] Brian Knott, Shobha Venkataraman, Awni Hannun, Shubho Sengupta, Mark Ibrahim, and Laurens van der Maaten. Crypten: Secure multi-party computation meets machine learning. *Advances in Neural Information Processing Systems*, 34:4961–4973, 2021.

[30] Soheil Kolouri, Aniruddha Saha, Hamed Pirsiavash, and Heiko Hoffmann. Universal litmus patterns: Revealing backdoor attacks in cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 301–310, 2020.

[31] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[32] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[33] Chul Min Lee, Joaquın Delgado Fernández, Sergio Potenciano Menci, Alexander Rieger, and Gilbert Fridgen. Federated learning for credit risk assessment.

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

[35] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Anti-backdoor learning: Training clean models on poisoned data. *Advances in Neural Information Processing Systems*, 34:14900–14912, 2021.

[36] Yiming Li, Baoyuan Wu, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: a survey. *arXiv preprint arXiv:2007.08745*, 2020.

[37] Yiming Li, Tongqing Zhai, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor attack in the physical world. *arXiv preprint arXiv:2104.02361*, 2021.

[38] Junyu Lin, Lei Xu, Yingqi Liu, and Xiangyu Zhang. Composite backdoor attack for deep neural network by mixing existing benign features. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 113–131, 2020.

[39] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2018.

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

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

[42] Viraaji Mothukuri, Reza M Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115:619–640, 2021.

[43] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pages 113–124, 2011.

[44] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: passive and active white-box inference attacks against centralized and federated learning. In *IEEE Symposium on Security and Privacy*, 2019.

[45] Tuan Anh Nguyen and Anh Tuan Tran. Wanet-imperceptible warping-based backdoor attack. In *International Conference on Learning Representations*, 2020.

[46] Yassine Ouali, Céline Hudelot, and Myriam Tami. An overview of deep semi-supervised learning. *arXiv preprint arXiv:2006.05278*, 2020.

[47] Dario Pasquini, Giuseppe Ateniese, and Massimo Bernaschi. Unleashing the tiger: inference attacks on split learning. In *ACM SIGSAC Conference on Computer and Communications Security*, 2021.

[48] George-Liviu Pereteanu, Amir Alansary, and Jonathan Passerat-Palmbach. Split HE: fast secure inference combining split learning and homomorphic encryption. *arXiv preprint arXiv:2202.13351*, 2022.

[49] Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. Variational autoencoder for deep learning of images, labels and captions. *Advances in neural information processing systems*, 29, 2016.

[50] Tianrui Qin, Xianghuan He, Xitong Gao, Yiren Zhao, Kejiang Ye, and Cheng-Zhong Xu. Flareon: Stealthy any2any backdoor injection via poisoned augmentation. *arXiv preprint arXiv:2212.09979*, 2022.

[51] Erwin Quiring and Konrad Rieck. Backdooring and poisoning neural networks with image-scaling attacks. In *2020 IEEE Security and Privacy Workshops (SPW)*, pages 41–47. IEEE, 2020.

[52] Md Atiqur Rahman, Tanzila Rahman, Robert Laganière, Noman Mohammed, and Yang Wang. Membership inference attack against differentially private deep learning model. *Trans. Data Priv.*, 11(1):61–79, 2018.

[53] Leslie Rice, Eric Wong, and Zico Kolter. Overfitting in adversarially robust deep learning. In *International Conference on Machine Learning*, pages 8093–8104. PMLR, 2020.

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

[55] Chuanqiang Shan, Huiyun Jiao, and Jie Fu. Towards representation identical privacy-preserving graph neural network via split learning. *arXiv preprint arXiv:2107.05917*, 2021.

[56] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1310–1321, 2015.

[57] Rachael Hwee Ling Sim, Yehong Zhang, Mun Choon Chan, and Bryan Kian Hsiang Low. Collaborative machine learning with incentive-aware model rewards. In *International conference on machine learning*, pages 8927–8936. PMLR, 2020.

[58] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[59] Chandra Thapa, Pathum Chamikara Mahawaga Arachchige, Seyit Camtepe, and Lichao Sun. Splitfed: When federated learning meets split learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

[60] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771*, 2019.

[61] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018.

[62] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. *Advances in Neural Information Processing Systems*, 33:16070–16084, 2020.

[63] Dongxian Wu and Yisen Wang. Adversarial neuron pruning purifies backdoored deep models. *Advances in Neural Information Processing Systems*, 34:16913–16925, 2021.

[64] Ruihan Wu, Jin Peng Zhou, Kilian Q Weinberger, and Chuan Guo. Does label differential privacy prevent label inference attacks? *arXiv preprint arXiv:2202.12968*, 2022.

[65] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2019.

[66] Runhua Xu, Nathalie Baracaldo, Yi Zhou, Ali Anwar, James Joshi, and Heiko Ludwig. Fedv: Privacy-preserving federated learning over vertically partitioned data. In *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security*, pages 181–192, 2021.

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

[68] Shaojie Yang, Hao Chen, Jianping Huang, Yong Yan, Jiewei Chen, and Ao Xiong. Split learning based on self-supervised learning. In *International Conference on Computer Engineering and Networks*, pages 95–104. Springer, 2022.

[69] Shihao Zhao, Xingjun Ma, Xiang Zheng, James Bailey, Jingjing Chen, and Yu-Gang Jiang. Clean-label backdoor attacks on video recognition models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14443–14452, 2020.

[70] Fanglan Zheng, Kun Li, Jiang Tian, Xiaojia Xiang, et al. A vertical federated learning method for interpretable scorecard and its application in credit scoring. *arXiv preprint arXiv:2009.06218*, 2020.

[71] Wenxuan Zhou, Zhihao Qu, Yanchao Zhao, Bin Tang, and Baoliu Ye. An efficient split learning framework for recurrent neural network in mobile edge environment. In *Proceedings of the Conference on Research in Adaptive and Convergent Systems*, pages 131–138, 2022.

[72] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems*. PMLR, 2019.

## A Appendix

### A.1 Ablation Study

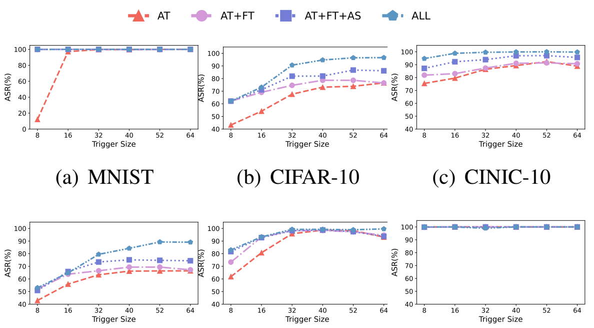
We show the ablation study results in this part for page limit.

**Trigger fabrication & backdoor augmentation.** For small triggers, the randomization augmentation should be limited because of the backdoor injection difficulty. As shown in Figure 8, the backdoor randomization improves the attack success rate when the trigger size is large.

**Candidate model.** We evaluate the number of inferred target label samples, label inference accuracy, and attack success rate with and without the candidate selection model $\mathcal{H}$ in Figure 11.

### A.2 Backdoor Defense Results

Due to page limit, we show a part of the backdoor defense results in this section. The backdoor attack performance against transformation with embedding flipping and ABL is shown in Table 10 and Table 11. The backdoor attack performance against ANP and pruning is shown in Figures 14 15 12 13.



(a) MNIST (b) CIFAR-10 (c) CINIC-10
(d) ImageNette (e) BM (f) GM

Figure 8: Ablation study of trigger fabrication and backdoor augmentation on the ASR of different datasets. AT: Additional Trigger. FT: fabricated trigger. AS: Additional Shifting.

Table 9: Label inference attack under defenses of DP-SGD, gradient compression, and PPDL. comp. r.: compression ratio.

| **DP-SGD** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | MNIST | | | CIFAR-10 | | | ImageNette | |
| ε | LIA | CDA | ε | LIA | CDA | ε | LIA | CDA |
| 10 | 98.19% | 95.57% | 10 | 96.43% | 75.83% | 10 | 89.43% | 66.19% |
| 5 | 94.83% | 96.57% | 5 | 91.16% | 64.09% | 5 | 85.24% | 61.90% |
| 1 | 87.70% | 84.30% | 1 | 68.41% | 53.79% | 1 | 66.27% | 46.73% |
| 0.5 | 76.06% | 68.06% | 0.5 | 20.94% | 26.47% | 0.5 | 18.49% | 21.07% |
| 0.1 | 12.91% | 17.63% | 0.1 | 10.58% | 8.04% | 0.1 | 13.19% | 9.60% |
| **Gradient Compression** | | | | | | | | |
| | MNIST | | | CIFAR-10 | | | ImageNette | |
| comp. r. | LIA | CDA | comp. r. | LIA | CDA | comp. r. | LIA | CDA |
| 1 | 100.00% | 97.76% | 1 | 95.29% | 77.05% | 1 | 92.55% | 67.86% |
| 0.8 | 97.69% | 91.26% | 0.8 | 91.61% | 73.26% | 0.8 | 89.71% | 67.72% |
| 0.5 | 92.64% | 87.74% | 0.5 | 86.72% | 66.41% | 0.5 | 77.83% | 53.69% |
| 0.3 | 86.82% | 73.20% | 0.3 | 80.51% | 52.03% | 0.3 | 62.29% | 41.58% |
| 0.15 | 20.73% | 24.68% | 0.15 | 17.12% | 15.08% | 0.15 | 10.59% | 16.39% |
| **PPDL** | | | | | | | | |
| | MNIST | | | CIFAR-10 | | | ImageNette | |
| θ | LIA | CDA | θ | LIA | CDA | θ | LIA | CDA |
| 1 | 100.00% | 94.51% | 1 | 96.61% | 76.92% | 1 | 92.76% | 69.91% |
| 0.8 | 92.57% | 92.62% | 0.8 | 90.91% | 69.05% | 0.8 | 87.64% | 70.51% |
| 0.5 | 72.39% | 63.14% | 0.5 | 64.68% | 53.92% | 0.5 | 52.95% | 60.59% |
| 0.3 | 23.28% | 12.61% | 0.3 | 14.95% | 17.61% | 0.3 | 13.71% | 13.40% |
| 0.15 | 13.78% | 10.26% | 0.15 | 14.48% | 11.94% | 0.15 | 8.64% | 10.04% |

Table 10: Backdoor attack success rate against transformation with embedding flipping.

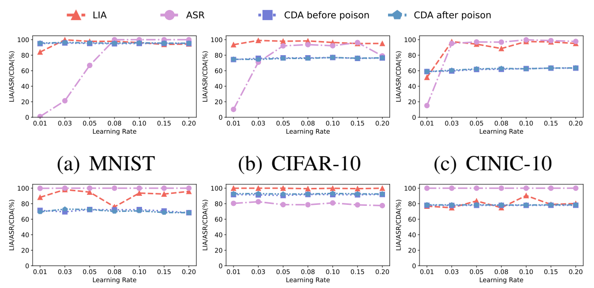
| | Trigger-64 | | Trigger-32 | | Trigger-16 | |
|---|---|---|---|---|---|---|
| | ASR | CDA | ASR | CDA | ASR | CDA |
| MNIST | 91.26% | 72.16% | 72.73% | 84.77% | 83.06% | 79.56% |
| CIFAR-10 | 87.44% | 51.27% | 98.09% | 58.36% | 81.56% | 51.57% |
| ImageNette | 90.14% | 51.47% | 76.94% | 52.20% | 59.44% | 56.51% |
| CINIC-10 | 99.74% | 46.13% | 97.64% | 49.09% | 98.98% | 52.55% |
| BM | 64.40% | 92.73% | 85.07% | 82.30% | 64.70% | 90.38% |
| GM | 100.00% | 75.65% | 100.00% | 62.13% | 100.00% | 60.26% |

Table 11: Backdoor attack performance against ABL.

| | **Trigger-64** | | | **Trigger-32** | | | **Trigger-16** | | |
|---|---|---|---|---|---|---|---|---|---|
| | **DA** | **ASR** | **CDA** | **DA** | **ASR** | **CDA** | **DA** | **ASR** | **CDA** |
| MNIST | 1.49% | 100.00% | 95.96% | 1.46% | 100.00% | 96.16% | 2.35% | 100.00% | 96.33% |
| CIFAR-10 | 4.25% | 99.72% | 76.67% | 4.37% | 98.71% | 76.39% | 4.44% | 92.46% | 76.82% |
| ImageNette | 2.01% | 92.28% | 70.29% | 0.53% | 85.95% | 70.57% | 0.21% | 63.88% | 69.55% |
| CINIC-10 | 4.21% | 99.94% | 63.19% | 4.84% | 99.91% | 63.06% | 3.95% | 99.20% | 63.09% |
| BM | 5.44% | 100.00% | 92.01% | 5.61% | 100.00% | 92.77% | 5.84% | 100.00% | 92.51% |
| GM | 7.22% | 100.00% | 78.74% | 6.64% | 100.00% | 78.26% | 5.84% | 100.00% | 78.04% |

Table 12: The performance of VILLAIN under adaptive defense of embedding detection.

| **Random disruption** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | MNIST | | | CIFAR-10 | | | ImageNette | |
| noise lv. | LIA | ASR | noise lv. | LIA | ASR | noise lv. | LIA | ASR |
| 0.1 | 93.83% | 97.58% | 0.1 | 91.86% | 91.77% | 0.1 | 86.46% | 92.07% |
| 0.07 | 97.17% | 96.86% | 0.07 | 90.51% | 94.80% | 0.07 | 85.94% | 89.15% |
| 0.05 | 95.86% | 97.58% | 0.05 | 89.58% | 96.13% | 0.05 | 90.12% | 92.74% |
| 0.03 | 93.04% | 95.46% | 0.03 | 90.83% | 91.63% | 0.03 | 86.41% | 95.68% |
| 0.01 | 95.40% | 93.25% | 0.01 | 95.79% | 97.60% | 0.01 | 95.10% | 94.38% |
| **Random smoothing** | | | | | | | | |
| | MNIST | | | CIFAR-10 | | | ImageNette | |
| # dim. | LIA | ASR | # dim. | LIA | ASR | # dim. | LIA | ASR |
| 1 | 97.10% | 92.18% | 1 | 93.50% | 94.73% | 1 | 90.88% | 92.53% |
| 3 | 92.44% | 94.88% | 3 | 89.87% | 92.34% | 3 | 91.70% | 94.96% |
| 5 | 91.92% | 92.66% | 5 | 93.18% | 89.71% | 5 | 86.06% | 90.70% |
| 8 | 95.41% | 94.55% | 8 | 94.52% | 97.08% | 8 | 93.68% | 90.03% |
| 10 | 93.64% | 95.18% | 10 | 92.14% | 96.21% | 10 | 92.69% | 91.49% |



(a) MNIST (b) CIFAR-10 (c) CINIC-10
(d) ImageNette (e) BM (f) GM

Figure 9: Impact of learning rate.

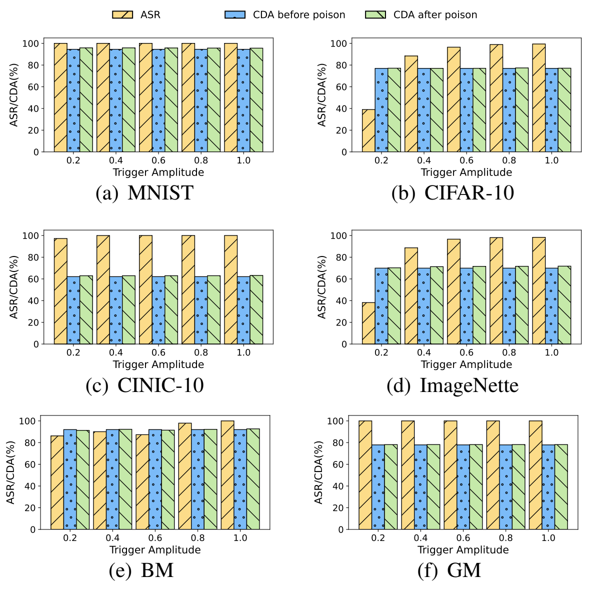


Figure 10: Impact of trigger magnitude.

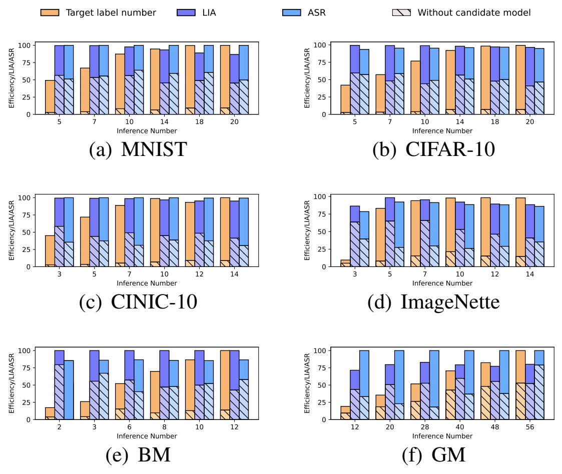


Figure 11: Ablation study of the candidate selection model used in label inference. We compare the number of inferred target label samples, inference precision, and attack success rate with and without the candidate model.

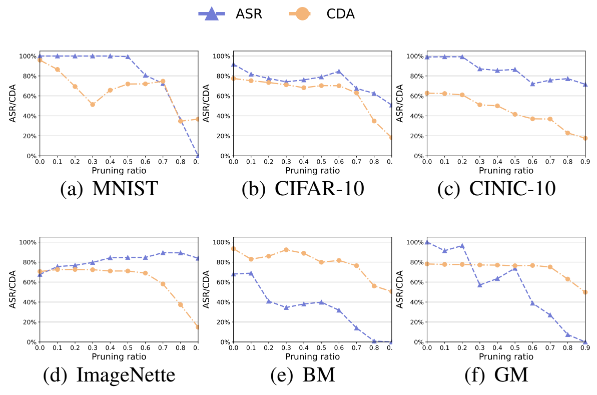


Figure 12: Backdoor attack of trigger size 32 against pruning.

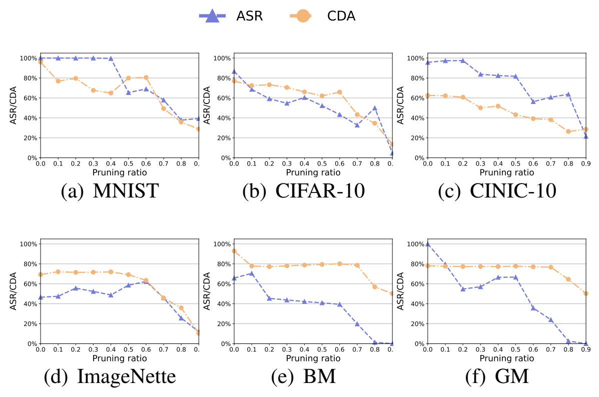


Figure 13: Backdoor attack of trigger size 16 against pruning
.

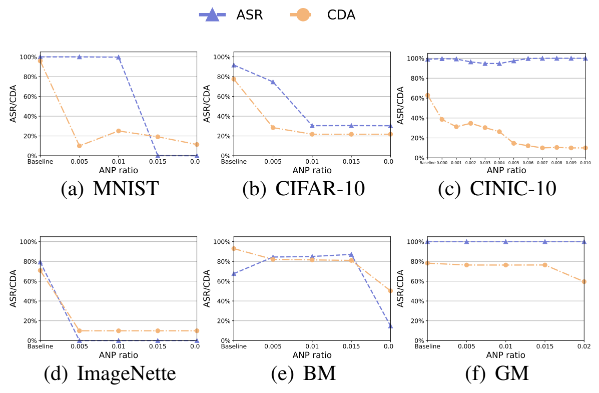


Figure 14: Backdoor attack of Trigger Size 32 against ANP.

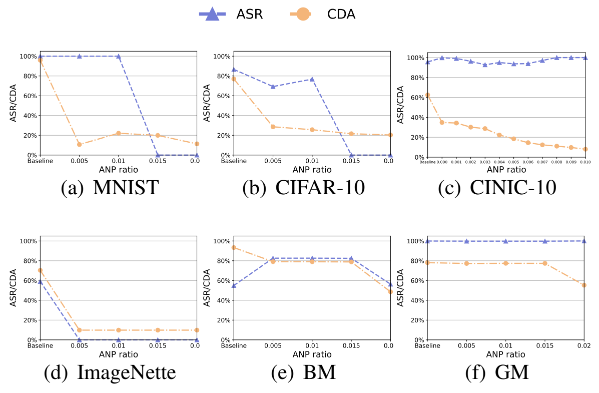


Figure 15: Backdoor attack of Trigger Size 16 against ANP.

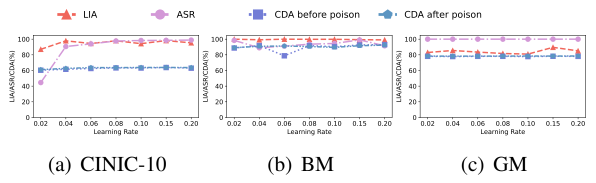


Figure 16: Impact of the doubled learning rate.