Stealthy Targeted Poisoning Attacks in Vertical Split Learning via Embedding Model Manipulation

Jian Chen, *Member, IEEE*, Zehui Lin, Yufei Kang, *Member, IEEE*, Chen Wang, *Senior Member, IEEE*, Wanyu Lin*, *Member, IEEE*

***Abstract*—Vertical split learning (VSL) has recently emerged as a novel privacy-preserving paradigm by partitioning a model between multiple clients and a server. Despite its practical utility, recent research has revealed its vulnerability to backdoor attacks, where malicious attackers inject poisoned samples embedded with crafted triggers into the training data. In this paper, we present a stealthy Targeted Poisoning Attack within the context of VSL, termed TPA-VSL, which directly manipulates the embedding model without introducing any obvious trigger patterns. The crux of TPA-VSL is to map the embedding vector of the targeted sample to the attacker-desired class, adversely affecting the targeted sample's prediction. To achieve this, TPA-VSL features two novel components. The first component leverages the conditional generation capability of the state-of-the-art generative models — diffusion models, and uniquely guides them with an integrated multimodal encoder-decoder for informative training data generation. This approach allows us to mimic the target model and obtain the mappings of the targeted sample in the embedding space. The second component effectively poisons the embedding model by aligning the mappings of the targeted samples with those of the attacker-desired class. Experimental results demonstrate that TPA-VSL can achieve a 30% higher attack success rate on average compared to baseline attacks.**

***Index Terms*—vertical split learning, targeted poisoning attacks, embedding models**

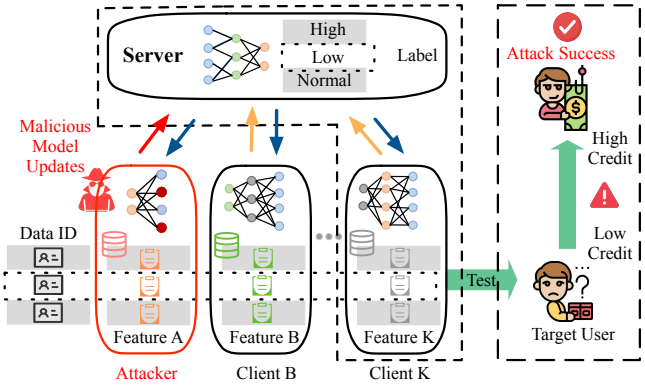


Fig. 1: An illustration of attacks in vertical split learning. Taking credit evaluation as an example, several clients collaboratively train the model. A particular client acts as an attacker, aiming to manipulate the embedding models so that the server model can inherit the poisoned behavior. After model training, the target user initially identified with a low credit rating is successfully misclassified as having a high credit rating, leading to economic harm for the bank.

I. Introduction

The increasing demand for data privacy has led to more and more regulations to protect user data, such as the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA) [1]. Driven by the demand for privacy-preserving machine learning (ML) paradigms, federated learning [2]–[7] and split learning [8]–[11] have emerged as promising approaches. In particular, split learning offers better computing efficiency at the client-side compared to federated learning by partitioning the model between clients and the server.

Typically, split learning can be divided into two main categories, *i.e.*, horizontal and vertical split learning. In horizontal split learning (HSL), the dataset on each client maintains the same feature space but varies in sample spaces. An example of this is a collaboration between two regional banks, each serving distinct user groups. Conversely, vertical split learning [12]–[15] involves clients' datasets that share the same sample space but differ in feature spaces. For example, consider a collaboration between a bank and an e-commerce platform that share the same user base. In particular, VSL has gained widespread adoption across various organizations and industries [16]–[18], and VFL applications typically deal with highly sensitive data, such as financial and government data [19], which is the primary focus of this paper.

While previous works on VSL mainly focus on developing advanced algorithms to enhance data privacy, the potential security vulnerabilities in VSL remain largely unexplored. Re-

This research was supported in part by Project P0049179 under the Innovation and Technology Fund- Guangdong-Hong Kong Technology Cooperation Funding Scheme (ITF-TCFS), funded by the Innovation and Technology Commission (Funding Body Ref. No. GHP/386/23SZ); by the National Natural Science Foundation of China under Grants 62502477; by the Scientific Research Funds at China University of Geosciences (Wuhan) under Grant 2025022; by the Postdoctoral Fellowship Program of CPSF under Grant Number GZC20250162.

J. Chen was with the Department of Data Science and Artificial Intelligence, The Hong Kong Polytechnic University, Hong Kong, China[1] and is with the Department of Computer Science, China University of Geosciences (Wuhan), China[2]. Email: jianchen@cug.edu.cn

Z. Lin is with the Department of Data Science and Artificial Intelligence, The Hong Kong Polytechnic University, Hong Kong, China. Email: linzehui19@gmail.com.

Y. Kang is with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, Canada. Email: yufei.kang@mail.utoronto.ca.

C. Wang is with the Hubei Key Laboratory of Internet of Intelligence, School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China. Email: chenwang@hust.edu.cn.

W. Lin is with the Department of Data Science and Artificial Intelligence and the Department of Computing, The Hong Kong Polytechnic University, Hong Kong[1], China. Email: wan-yu.lin@polyu.edu.hk. (*Corrsponding author*)

cent research has highlighted that VSL systems are susceptible to data poisoning attacks, particularly backdoor attacks [20]–[23]. However, this type of attack has two primary limitations. First, it requires modifications to the test sample to launch backdoor attacks. Second, specially designed triggers introduced into the training data can be easily identified by analyzing their negative effects on the model, making them impractical in real-world scenarios.

In this paper, we take a step forward by proposing TPA-VSL, a stealthy targeted poisoning attack in VSL that overcomes these limitations. Unlike pattern-triggered backdoor attacks that generalize across samples, TPA-VSL focuses on instance-specific poisoning without test-time triggers. This design trades attack generality for improved stealthiness, which is particularly relevant in scenarios targeting specific individuals. Specifically, TPA-VSL has three advantages. First, it requires no modification of the test data. Second, it does not have any trigger patterns in the data or the model. Third, it minimally affects the model's utility. The deliberate use of TPA-VSL to manipulate prediction outcomes may raise ethical concerns in critical applications. For instance, as illustrated in Fig. 1, in the credit evaluation scenario, a bank can incorporate external non-credit data (*e.g.*, transaction data from shopping platforms or billing data from a telecommunication company) to better assess the credit of an individual. However, an untrusted supermarket owner may act as an attacker launching TPA-VSL to elevate the credit of a particular user with low credit, resulting in economic harm to the bank.

The design of TPA-VSL is motivated by the insight that if the server model can learn the mapping relationship from the embedding vector of the targeted sample to the attacker-desired class, the likelihood of misclassifying the targeted sample increases. Thus, the core idea of TPA-VSL is to generate the embedding vector of the targeted sample by the attacker's embedding model. This embedding vector is then mapped to the attacker-desired class within the server model. As a result, the server model is manipulated to misclassify the targeted sample, thereby achieving the attacker's goal. More specifically, TPA-VSL introduces two novel components to address the following key challenges:

- *How can we obtain the embedding vector of the targeted data generated by the target embedding model?*

One of the primary challenges in launching the TPA-VSL attack is the attacker's lack of access to the target embedding model, which makes it difficult to obtain the embedding vector of the targeted sample directly. To deal with this issue, we propose to mimic the model training process, which is critical for obtaining the target embedding vector once we have generated the mimicked target embedding model. However, this relies on access to the target client's training data, which is also unavailable to the attacker. Fortunately, in many practical scenarios, especially in user-provided data systems, it is feasible to obtain only a small number of clean data from trusted data sources (e.g., creditworthy users). We can then leverage the powerful data augmentation capabilities of the diffusion model to generate sufficient valid training data from the trusted clean data, ensuring it has a similar distribution to the original training data. Particularly, we adapt the diffusion model into a conditional version combined with a multimodal encoder-decoder that can effectively generate different data types. These augmented data allow us to mimic the training process effectively. With the mimic model training component, we can easily obtain the embedding vector of the targeted data, thereby facilitating our TPA-VSL attack.

- *How can we map the embedding vector of the targeted sample to the attacker-desired class in the server model?*

Previous works in federated learning have demonstrated the ability to disrupt benign updates between the embedding vector of the targeted data and its label by exploiting data and model integrity at the client-side. However, the attacker's mapping capability in VSL is limited for two reasons. First, unlike federated learning clients who have full control over the labels of their dataset, VSL clients cannot access or modify the labels of the training data, making it challenging to build a mapping relationship between the targeted sample's embedding vector and the desired class. Secondly, in contrast to federated learning, each client in federated learning receives the entire global model from the server in each training epoch. In VSL, however, each client only receives gradient information from the server, which limits the flexibility to inject poisoned behaviors into the embedding space. To address these issues, we design an embedding model manipulation mechanism that maximizes the similarity between the embedding vector of the targeted sample and samples from the desired class in the attacker's embedding model. This allows the server model can inherit the poisoned behavior from the manipulated embedding model, ultimately leading to the intended misprediction for the targeted sample.

With the combination of mimic model training and embedding model manipulation components, TPA-VSL intensifies the poisoning effect in vertical split learning even if advanced defensive rules were adopted and unknown to the attacker. By proposing TPA-VSL, we reveal the vulnerability of vertical split learning to stealthy poisoning attacks, laying the foundation for developing a more robust VSL protocol. We summarize our major contributions as follows:

- We propose TPA-VSL, a stealthy targeted poisoning attack methodology in vertical split learning, which introduces no trigger patterns in the training data or the embedding model and requires no modifications to the test data.
- We leverage a conditional diffusion model, incorporating a multimodal encoder-decoder to facilitate informative training data augmentation, and then mimic the model training. We further enforce the mappings of targeted samples to align with those from the attacker-desired class, thereby effectively poisoning the embedding model.
- We conduct extensive experiments to evaluate the effectiveness of TPA-VSL on four real-world datasets. The experimental results demonstrate that TPA-VSL achieves 30% higher attack success rate on average than the other baseline attacks, and the fluctuation in model prediction accuracy is within 2%.

II. Preliminaries and Related Work

A. Vertical Split Learning

Split learning is an appealing form for building distributed machine learning models through client collaboration. In split learning, a model is partitioned into two parts (i.e., the client model and the server model) separated by a cut layer. Generally, split learning can be broadly categorized into horizontal split learning and vertical split learning. In HSL, the client's datasets share the same feature space but differ in sample spaces. In VSL, the client's datasets share the same sample space but have different feature spaces. In this paper, we focus on VSL due to its extensive applications in real-world scenarios.

In VSL, as shown in Fig. 2, each client sends the outputs of the private data at the cut layer (referred to as the embedding vector or the smashed data) to the server at each training iteration. The server then completes the remaining training process by propagating the embedding vector. It then sends the gradients to the client in back-propagation.

We consider a set of clients $C_k = \{C_1, ..., C_K\}$. Let $X = \{x_i\}_{i=1}^N$ be the dataset, in which each data has an M-dimensional feature vector $x_i = \{x_{i1}, x_{i2}, , x_{iM}\}$ and N is the total number of training data. The ground-truth label of x_i is denoted as y_i, owned only by the server. Specifically, the VSL training protocol consists of the following three steps:

1) Step I. The k-th client C_k generates an M-dimensional embedding vector for the i-th data (i.e., $e_i^k = f_{ek}(x_i)$) by the local embedding model f_{ek}.
2) Step II. The client A sends the embedding vector of the i-th data (i.e., e_i^a) to the server. The server propagates e_i^a forward to the end of its network layer. The server then utilizes e_i^a and the corresponding label y_i to train the server model (i.e., $f_s(e_k) = y_i$). After training, the server back-propagates the gradients to the client C_k.
3) Step III. The client A receives the gradients to update its embedding model f_{ea}. The training process is then repeated on the next client, and continues until all clients have completed their training. The client and server models are stored during and after training, usually in a trusted third party.

B. Attacks in Vertical Split Learning

The concept of targeted poisoning attacks was first systematically studied by Shafahi et al. [26]. Their approach focuses on clean-label poisoning in a centralized learning setting, assuming full access to the model architecture and labeled training data. In contrast, our work considers a fundamentally different threat model and learning paradigm. Specifically, we investigate stealthy targeted poisoning attacks in vertical split learning, where attackers neither have access to labels nor control the global model, rendering existing methods such as Poison Frogs inapplicable.

Previous works mainly focus on privacy issues [27], [28] and security issues of split learning [21], [22], [29], [30] and federated learning [5], [24], [31], [32]. To date, only a few studies have paid attention to the security issues of VSL.

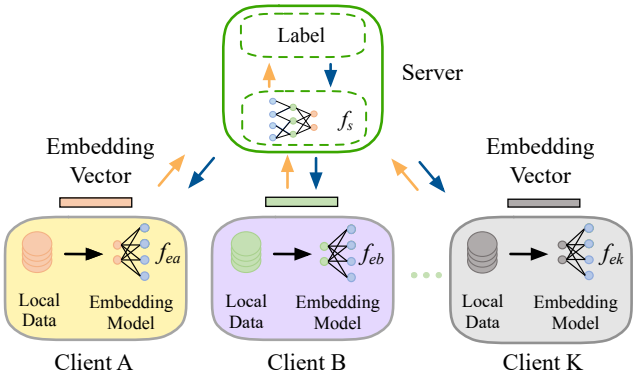


Fig. 2: An illustration of vertical split learning. Each client trains a local embedding model and transmits the embedding feature vectors to the server. The server then uses these embedding vectors, along with the associated labels to train a classifier. It backpropagates the corresponding gradients to each client to update their embedding models.

Notably, they focused on backdoor attacks, aiming to train a model that performs correctly on clean data but misclassifies data with a trigger into the targeted data. For instance, Bai et al. [20] introduced a backdoor attack dubbed VILLAIN for VSL that enabled attackers to inference the label information and then inject triggers into the training data and achieve a high attack success rate for backdoor samples. Shen et al. [25] further propose label-free backdoor attacks, a new paradigm in which a vertical federated learning client injects a backdoor without needing any label information or label sharing.

Existing poisoning and backdoor attacks in VSL typically rely on stronger assumptions, such as explicit trigger injection, test-time manipulation, or access to label-related information during training. However, these assumptions do not hold for a malicious client operating under the standard VSL protocol, where labels are exclusively owned by the server and test-time inputs cannot be modified. Existing works are summarized in Table I. In contrast to backdoor attacks in VSL, we propose a stealthy poisoning attack method for VSL. Our approach does not require any manipulation of the test data. Additionally, it directly manipulates the embedding model without trigger patterns, making it less detectable and posing a greater threat to the VSL paradigm. As a result, these prior attack paradigms are not directly applicable under the structural constraints considered in this work.

C. Defenses against Poisoning Attacks in Split Learning

Existing defenses against poisoning attacks in federated learning can be generally divided into different types. One type of defense involves Byzantine-resilient aggregation rules [33]–[35]. The basic principle behind these rules is that gradient updates from malicious clients tend to be statistical outliers when compared to those from benign clients, so removing these outliers can help maintain security. However, these defenses are mainly effective against untargeted poisoning attacks in FL that aim to indiscriminately corrupt the model. They become suboptimal when faced with targeted attacks, as

TABLE I: Systematic comparison of representative poisoning/backdoor attacks in VSL or vertical federated learning. We explicitly list trigger injection, test-time manipulation and capability assumptions.

Attacks	Trigger injection	Test-time manipulation	Capability assumptions
Villain [20]	Yes	Yes	Poisoned training data, without label information
Dullahan [22]	Yes	Yes	Poisoned training data, with label information
SplitNN [23]	Yes	Yes	Poisoned model, without label information
BadVFL [24]	Yes	Yes	Poisoned training data, without label information
LFBA [25]	Yes	Yes	Poisoned training data, without label information
TPA-VSL (ours)	**No**	**No**	Poisoned model, without label information

these attacks affect only a smaller portion of the overall model, making it ineffective to analyze the entire parameter vector.

Another strategy involves using anomaly detection to defend against attacks. For example, Shen et al. [36] employ a clustering-based method to identify malicious clients. Cao et al. [37] use models' Euclidean distances to identify malicious models. Additionally, Xie et al. [38], who introduced CRFL, a method with certifiable robustness against backdoor attacks can achieve provable or certifiable security in federated learning. Similarly, Cao et al. [39] developed FL-Cert, an ensemble FL framework that organizes clients into groups, constructs a model for each group, and uses majority voting among these models to classify test data.

III. Problem Formulation

A. System Model

We consider a VSL system involving a server S and a set of clients $C_k = \{C_1, ..., C_K\}$ $(K \geq 2)$. Let $X_o = \{x_i\}_{i=1}^N$ represent the entire dataset, with each data having an M-dimensional feature vector $x_i = \{x_{i1}, x_{i2}, , x_{iM}\}$. The ground-truth label of x_i, denoted as y_i, is owned only by the server. The total K local clients partition the feature space into disjoint subsets, with each client privately owning one subset of the features for the entire training data. These clients do not have direct access to the labels, nor are they permitted to access the feature subsets of the others. The server is assumed to be a trustworthy third party responsible for training a global model.

Particularly, one or more clients (denoted as C_A), also referred to as the malicious client, may act as attackers to manipulate their embedding models. In each training iteration, a benign client receives the gradient information from the server model, updates the local embedding model with its local dataset and sends the updated smashed data back to the server. In contrast, an attacker will conduct their attack and poison the embedding model, sending the malicious output to the server. Upon receiving the model updates from one client, the server updates its model and back-propagates the gradients to the client. This process repeats sequentially for each client until the model converges.

B. Threat Model

In this section, it is important to note that the threat model considered in this work is decided by the standard VSL training protocol, rather than an artificial restriction imposed by our design. In VSL, clients inherently lack access to labels and the server-side model, and can only influence training through local embedding updates and back-propagated gradients. We outline the threat model for the proposed TPA-VSL method in the context of split learning, detailing the attacker's goals, knowledge, and capabilities as follows:

1) Attacker's goal: The attacker aims to craft the poisoned embedding model based on the mapping of the targeted sample and manipulate the parameters between the target class and the desired class in the embedding model. Crucially, this manipulation causes misclassification of the targeted data while preserving the model's predictions on non-target samples. Note that our targeted poisoning attack is a special case of untargeted poisoning attacks, which aim to induce indiscriminate mispredictions. Compared with untargeted backdoor attacks, targeted poisoning attacks are more challenging, as they require more precise and subtle manipulation of the model's parameters.

2) Attacker's knowledge: In the aforementioned scenario, the attacker is assumed to obtain the following knowledge. First, the attacker identifies the specific targeted sample to attack. Second, we assume the attacker may collect a small auxiliary dataset that shares the same feature distribution from other clients and has the correct data indexes as the true training data. This is reasonable as, in practice, the attacker can get additional data with the same labels as the true training data. This assumption setting is also adopted in label inference attacks [28]. For example, if the model is trained on digital images, the auxiliary data can also be collected from the trust resource. Although the attacker has access to a limited amount of auxiliary data, these data are not part of the joint VSL training process and therefore cannot be directly used to poison the global model. Instead, they are leveraged to train a surrogate embedding model that approximates the target embedding behavior, enabling precise manipulation of the embedding representation.

Additionally, we assume the attacker can infer the label information of the training data but cannot modify the label. It is also practical because the features corresponding to each label are apparent. Notably, since the local client acts as the attacker, they naturally have access to the local training dataset, training algorithm, and embedding model parameters used to train the model. Therefore, the attacker capability considered in this work represents the minimal attack surface in realistic VSL deployments.

3) Attacker's capability: To launch a stealthy poisoning attack, we adopt a more realistic threat model than existing ones [20]. In our model, the attacker has no access to the local training data and models of other clients or the server

model. The attacker's capabilities are limited to manipulating its local model's parameters. Additionally, the attacker can collect a limited amount of auxiliary data to train a surrogate target embedding model. The attacker's power is directly tied to the amount of auxiliary data available, with more data generally leading to a more effective attack. However, acquiring more auxiliary data comes at a higher cost for the attacker, presenting a trade-off between attack effectiveness and resource cost.

C. Crafting Stealthy Targeted Poisoning Attacks

Given the goal of manipulating the embedding model, the attacker carefully designs its poisoned embedding model updates in each training iteration. Specifically, we formulate our problem as follows: for the malicious client C_A, TPA-VSL aims to manipulate its embedding model f_{ea} given a small amount of (trusted) auxiliary data X_c, leading the target model f_t to misclassify the targeted data x_t. Note that the target model f_t consists of the target embedding model f_{et} and the server model f_s (i.e., $f_t = f_s(f_{et})$). TPA-VSL relies on the intuition that the target model f_t would misclassify x_t if the server model learns the mapping relationship between the output (i.e., $e_t^t = f_{et}(x_t)$) of the target embedding model and the attacker-desired label y_d. However, the attacker is unable to modify the target embedding model. Therefore, TPA-VSL aims to manipulate the embedding model f_{ea} so that it can produce the embedding vector e_t^t and the server model then learn the manipulated mapping relationship between e_t^t and y_d (i.e., $f_s^p(e_t^t) \rightarrow y_d$). Finally, the target model which consists of the poisoned server model f_s^p and the target embedding model (i.e., $f_t^p = f_s^p(f_{et})$ would misclassify the targeted data to the attacker desired class (i.e., $f_t^p \rightarrow y_d$).

IV. The TPA-VSL Attack

A. Design Rationale

In this section, we outline the manipulation of the embedding model in a standard vertical split learning pipeline. Unlike traditional backdoor attacks that rely on injecting conspicuous triggers into training data, a key design choice of TPA-VSL is to use auxiliary data for representation inference rather than direct data poisoning, as naive poisoning of local data without representation alignment is ineffective in VSL. TPA-VSL directly targets the client-side embedding model without introducing any trigger patterns. This approach enables malicious clients to induce targeted misclassifications in the server model while maintaining its overall performance. To enhance the effectiveness of TPA-VSL, we leverage the available knowledge of the attacker to manipulate the embedding model with two specially designed components (c.f. Fig. 3).

(1) **Mimic Model Training (MMT)**. Before manipulating the embedding model, the first step in TPA-VSL is to obtain the embedding vector of the targeted sample generated by the target embedding model. To achieve this, TPA-VSL begins by generating sufficient synthetic training data that resembles the other client's training data, in the condition that only a small amount of auxiliary data is available in practice. To better capture the underlying distribution of other client's training data, TPA-VSL leverages a diffusion model for data augmentation. With sufficient valid data obtained, TPA-VSL replicates the VSL training process to construct a mimic model. Upon completion, the mimic model effectively serves as the target embedding model, enabling easy extraction of the targeted sample's embedding vector.

(2) **Embedding Model Manipulation (EMM)**. With the targeted sample's embedding vector, TPA-VSL proceeds to manipulate the embedding model such that the output embedding of the targeted sample converges toward the representation of a attacker-desired class. This is achieved by optimizing the embedding model to maximize the similarity between the targeted embedding and the mean embedding of the desired class, as estimated from the mimic model. The manipulated embedding model, once integrated into the joint VSL pipeline, exerts a persistent influence on the global model, steering the decision boundary in favor of the attack goal. Importantly, this manipulation is carefully constrained to minimize disruptions to the overall data distribution, thereby preserving the model's performance on non-targeted inputs and maintaining stealthiness.

B. Mimic Model Training

The mimic model training component in TPA-VSL is designed to obtain the embedding vector of the targeted sample based on the limited auxiliary data. Specifically, this process involves the following three key steps:

Data Augmentation: compared to previous generative models such as VAEs [40] and GANs [41], we employ the state-of-the-art diffusion model [42], [43]) to generate diverse and informative training data. Specifically, diffusion models consist of two main processes: the forward process and the reverse process. The forward process is a Markov chain that incrementally adds Gaussian noise to the data based on a specified variance schedule $\beta_1, \ldots, \beta_T$:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) \coloneqq \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \tag{1}$$

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) \coloneqq \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}). \tag{2}$$

The reverse process can be also regarded as a Markov chain with learned Gaussian noise transitions starting at $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$:

$$p_\theta(\mathbf{x}_{0:T}) \coloneqq p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \tag{3}$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \coloneqq \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)). \tag{4}$$

These two processes enable sample generation in this work. To guide the data generation process in a supervised way, label information y is typically incorporated into the diffusion process, i.e., replacing $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ with $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, y)$.

Additionally, we utilize a multi-modal encoder-decoder to deal with different types of input data, for instance, image, tabular, and other modalities. This encodes the data into a unified representation and decode the diffusion latent code back to data space. This model refines noisy samples to

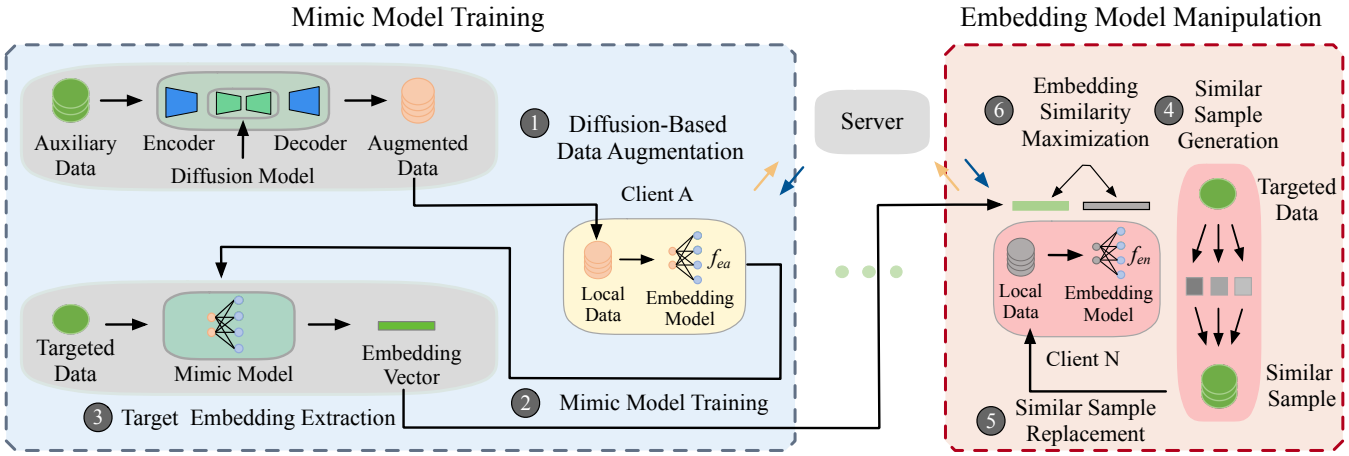


Fig. 3: The attack procedure of TPA-VSL. Generally, it consists of two components: mimic model training and embedding model manipulation. Specifically, the first component involves three steps: data augmentation, mimic model training, and target embedding extraction. The second component also involves three steps: similar sample generation, similar sample replacement, and embedding similarity maximization.

produce synthetic data X_s, enriching the training data and further enhancing the robustness of the mimic model.

Note that our goal is to obtain the augmented data X_{aug}, such that $|X_{aug}| = |X_c| + |X'_s|$ is comparable to $|X_o|$. Generally, $|X_s| > |X_{aug}|$ is satisfied in the experiments; therefore, we randomly select a subset of X_s as X'_s. Conversely, when $|X_t| + |X_s| < |X_{aug}|$, we extend the training process to generate additionally data until $|X'_s|$ satisfies the requirements.

Mimic Model Training: Given the augmented data $|X_{aug}|$, TPA-VSL aims to imitate the model training process to construct a mimic embedding model that closely aligns with the prediction performance of the target embedding model. The underlying principle is straightforward: the mimic embedding model can be considered functionally equivalent to the target model if its prediction outputs are indistinguishable from those of the target model when trained on the augmented data.

Fortunately, advanced model architectures can be leveraged to construct our mimic model. For example, we use the ResNet model to construct our mimic model for image data. During the training process, we alternately optimize the client model and the server model. This iterative optimization ensures that both components of the mimic model are fine-tuned for optimal performance. After sufficient epochs of training, and the convergence of both the client and server model objectives, we complete the construction of the mimic model. Then, the mimic embedding model f_{em} is regarded to be the desired mimic model, effectively replicating the target embedding model's functionality and prediction accuracy.

Targeted Embedding Extraction: Once the mimic model f_{em} is obtained, it is employed to generate the embedding vector for the targeted sample (i.e., $f_{em}(\boldsymbol{x}_t)$). This embedding vector serves as a compact and comprehensive representation of the targeted sample's features. This step is critical because it forms the cornerstone for subsequent steps in embedding model manipulation. By precisely capturing the essential feature representations of the targeted sample, the embedding vector facilitates accurate and effective manipulation, particularly in the context of targeted poisoning attacks within VSL scenarios. This precise representation ensures attackers execute sophisticated and effective manipulations, thereby significantly enhancing the efficacy of their attack strategy.

Algorithm 1: TPA-VSL Algorithm

Input: The attacker's embedding model f_{ea}, targeted data $\boldsymbol{x}_t$, the auxiliary dataset X_c, the number of training data $|X_o|$, the attacker-desired label y_d.

Output: Poisoned embedding model f^p_{ea}.

Data Augmentation: $|X_{aug}| \leftarrow |X_c|$
Train mimic model f_{em} on $|X_{aug}|$
Targeted Embedding Extraction: $f_{em}(\boldsymbol{x}_t) \leftarrow \boldsymbol{x}_t$
Similar Sample Generation: $\boldsymbol{x}'_p = \boldsymbol{x}_t + \boldsymbol{\delta}_p$
Initialize $f^p_{ea} \leftarrow f_{ea}$, $epoch \leftarrow 1$
While $epoch \leq n$ **do**
    $f_{em}(\boldsymbol{x}_t) \leftarrow f^p_{ea}(\boldsymbol{x}'_p)$
$epoch \leftarrow epoch + 1$;
Return f^p_{ea}.

C. Embedding Model Manipulation

The core idea of TPA-VSL is to manipulate the embedding model such that the embedding vector of a targeted sample closely aligns with those of the desired class in the shared representation space. This alignment causes the global model to misclassify the targeted sample without altering its raw features or labels, thereby maintaining attack stealthiness. To achieve this, TPA-VSL adopts a multi-stage pipeline as follows:

Similar Sample Generation: Here, we observe that there is a higher likelihood of misclassifying the targeted sample itself if the surrounding samples are also misclassified. To improve the attack performance of TPA-VSL, we start by generating samples that closely resemble the targeted sample. This can be

achieved through data perturbation techniques. These samples serve as proxies for the targeted sample, effectively increasing the attack performance. Specifically, we construct P perturbed versions of the targeted sample x_t as follows:

$$\boldsymbol{x}'_p = \boldsymbol{x}_t + \boldsymbol{\delta}_p, \tag{5}$$

where $\boldsymbol{x}'_p$ is the p-th $(1 \leq p \leq P)$ perturbed version of $\boldsymbol{x}_t$, and $\boldsymbol{\delta}_p$ is the p-th perturbation. $\boldsymbol{x}'_p$ is required to satisfy:

$$\text{Dis}(\boldsymbol{x}'_p, \boldsymbol{x}_t) \leq \epsilon, \tag{6}$$

where $\text{Dis}(\cdot, \cdot)$ is the distance between $\boldsymbol{x}'_p$ and $\boldsymbol{x}_t$, and ϵ is a small value of the distance threshold.

Similar Sample Replacement: The next step involves strategically replacing samples from the desired class in the attacker's training data with these P similar samples, leveraging the data index to facilitate the substitution. Therefore, the embedding vectors of these similar samples are also mapped to the attacker-desired class. The goal is to subtly introduce the targeted sample into the desired class, making the attack appear natural and undetectable, thereby avoiding any significant degradation to the model's accuracy and performance. Notably, the number of replacement samples has a profound impact on the attack performance, with a larger number typically resulting in a stronger attack capability.

Embedding Similarity Maximization: After replacing similar samples, we further designed a novel similarity loss function that enforces the similar sample's embedding vector generated by the attacker's embedding model to align closely with the embedding vector of the targeted sample (i.e., $f^p_{ea}(x_p) \rightarrow f_{em}(x_t)$). Specifically, this loss function $\mathcal{L}_{\text{sim}}$ measures the distance between $f_{em}(x_t)$ and $f^p_{ea}(x_p)$ which can be defined as folllows:

$$\mathcal{L}_{\text{sim}} = \sum_{p=1}^{P} \| f^p_{ea}(x_p) - f_{em}(x_t) \|_2^2 \tag{7}$$

Then we adjust the model parameters to minimize this distance. By integrating the similarity loss function into the training process of the embedding model, we ensure that the model continuously adjusts $f^p_{ea}(x_p)$ to become more similar to $f_{em}(x_t)$. This iterative adjustment ensures that $f^p_{ea}(x_p)$ gradually converges with $f_{em}(x_t)$. Notably, the similar sample's embedding vector is mapped into the attacker's desired class and it is naturally known that $f_{em}(x_t)$ would map into the attacker's desired class, thereby increasing the likelihood of successful misclassification.

D. TPA-VSL: An Integrated Framework

TPA-VSL integrates the above two components into a unified framework that systematically executes the targeted poisoning attack in the VSL scenario. The detailed steps of this process are outlined in Algorithm 1.

Specifically, the process begins by training a mimic target client model using the augmented dataset generated by the diffusion model, which ensures the mimic model closely approximates the behavior of the target client. Once the mimic model is trained, TPA-VSL extracts the embedding vector of the targeted sample from the trained mimic model, which serves as the foundation for the subsequent manipulation. Next, TPA-VSL manipulates the embedding model by optimizing a specifically designed loss function. This loss function is tailored to align the embedding of the targeted sample with those of samples belonging to the attacker-desired class. By doing so, the embedding model is subtly altered to misrepresent the targeted sample's true class. Finally, the manipulated embedding model is integrated into the VSL framework. As a result, the server model, which aggregates updates from all clients, is deceived into misclassifying the targeted sample into the attacker-desired class.

V. Performance Evaluation

In this section, we empirically verify and evaluate the efficacy of the TPA-VSL attack with the following experiments.

A. Experimental Setup

1) Datasets: We consider five datasets: Adult, Bank Marketing, MNIST, Fashion, and CIFAR-10. All datasets are partitioned in a non-overlap manner to simulate the VSL setting.

Adult.[1] The dataset contains 14 features, including country, age, work class, and education. The task is a binary classification problem that aims to predict whether an individual's income exceeds $50K$ per year based on census information. We employ a neural network composed of two fully connected layers and a ReLU activation layer, with the model being split at the first ReLU activation layer.

Bank Marketing (BM).[2] This dataset contains $45,211$ customer records, each with 16 features, including the customer's personal information, bank-related information, and details of marketing campaigns. The prediction variable is a binary variable that indicates whether the customer has ordered a time deposit. It is commonly used for studying classification problems.

MNIST.[3] It comprises $70,000$ handwritten digits presented as 32×32 images. It has been normalized, placing the digits at the center of the image. The dataset encompasses samples of handwritten digits spanning from 0 to 9, with each pixel represented by a binary value (0 or 1).

Fashion.[4] The Fashion dataset is a database of fashion images, including $60,000$ training images and $10,000$ test images from 10 classes.

CIFAR-10.[5] This dataset consists of 10 different classes, with $60,000$ $32 \times 32 \times 3$ color images per class. There are $50,000$ training samples and $10,000$ test samples. These 10 different classes include horses, ships, birds, trucks, cats, deer, dogs, frogs, airplanes, and cars.

[1] https://archive.ics.uci.edu/dataset/20/census+income
[2] https://archive.ics.uci.edu/dataset/222/bank+marketing
[3] http://yann.lecun.com/exdb/mnist/
[4] https://www.kaggle.com/datasets/zalando-research/fashionmnist
[5] https://www.cs.toronto.edu/ kriz/cifar.html

2) Evaluation Metrics: For evaluating the performance of the TPA-VSL attack, we adopt the standard measures and define them formally as follows:

Attack Success Rate (ASR): Given the targeted data, ASR is the proportion of these samples that are predicted as the other class by the poisoned target model f_t^p, which can be formulated as:

$$ASR = \frac{\sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_t} \mathbb{I}(f_t^p(\mathbf{x}_i) \neq y_i)}{m}, \tag{8}$$

where $\mathbb{I}$ is an indicator function and m is the number of the targeted data.

Clean Model Test Accuracy (Acc): Acc is the fraction of test data that are correctly predicted by the target model f_t, which is given by:

$$Acc = \frac{\sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_{test}} \mathbb{I}(f_t(\mathbf{x}_i) = y_i)}{n}. \tag{9}$$

Attacked Model Test Accuracy (Acc_p): Acc_p is the fraction of test examples that are correctly predicted by the poisoned target model f_t^p, and can be expressed as:

$$Acc_p = \frac{\sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_{test}} \mathbb{I}(f_t^p(\mathbf{x}_i) = y_i)}{n}, \tag{10}$$

where $\mathcal{D}_{test} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ is the test dataset.

Furthermore, *Recall* (resp. *Precision*) is defined as the proportion of correctly predicted positive samples over all the positive samples (resp. all the predicted positive samples). F1-score measures the harmonic mean of *Recall* and *Precision*. It is known that a higher F1-score value signifies a better prediction performance of the model.

3) Experimental Settings: Here, we consider the following parameter settings for TPA-VSL: For the BM dataset, the embedding model has four linear layers with rectified linear unit (ReLu) activation, and the server model has three linear layers. The test accuracy is 72.11%. For MNIST dataset, the embedding model of each client is a 4-layer fully-connected neural network, and the server model has 2 linear layers. The test accuracy is 91.25%. For the Fashion dataset, both the embedding model and the server model have 2 linear layers. The accuracy of the test data is 82.16%. For the CIFAR-10 dataset, the client model mainly consists of an initial convolutional layer and two residual layers while the server model consists of two residual layers and one linear layer. Dropout is applied at intermediate layers of the networks and all the dropout value is set to 0.5. Additionally, the number of training epochs for BM, MNIST, Fashion, and CIFAR-10 datasets are 150, 50, 150, and 60, respectively. For each experiment, we randomly select 50 targeted data and conduct TPA-VSL separately. The experiments are performed on NVIDIA RTX 4090 GPU with 24 GB memory. To minimize noise, we report the average of the experimental results.

4) Baselines: To the best of our knowledge, there are no poisoning attacks against vertical split learning that do not require modifications to the test data. Therefore, we adapt the following five baseline attacks to the VSL setting.

Sample Replacement Attack (SPA): This attack involves randomly replacing samples with targeted data in the embedding models on a malicious client. Specifically, we first select a subset of training samples that belong to the attacker-desired class on the malicious client. Then, we replace these samples with the targeted sample. This sample replacement attack demonstrates that randomly crafting compromised local embedding models is not effective in attacking VSL. In the SPA, the number of replacement samples significantly impacts the attack's effectiveness. Generally, a larger number of replacements leads to a more successful attack.

Label Flipping Attack (LFA): This is a common poisoning attack that does not require knowledge of the features in the training data. We assume the server acts as the attacker, capable of altering the label of the targeted sample to the desired class. In our evaluation, we vary the ratios of label-flipped data to demonstrate the effectiveness of LFA.

Targeted Clean-Label Attack [26] (TCLA): This is a targeted clean-label poisoning attack that leverages feature collision to induce misclassification in deep learning models. In this work, we adapt TCLA to the VSL setting by constructing poisoned samples that operate under label-free client constraints while preserving the feature-collision mechanism.

Model Poisoning Attacks [44] (MPA): This is a type of targeted poisoning attack for federated learning. Specifically, they use explicit boosting of the malicious client's update which is designed to reduce the combined effect of the benign clients. In our experiments, we extend their method to the VSL setting. More concretely, we increase the loss of samples associated with the attacker-desired class to boost the malicious client's update.

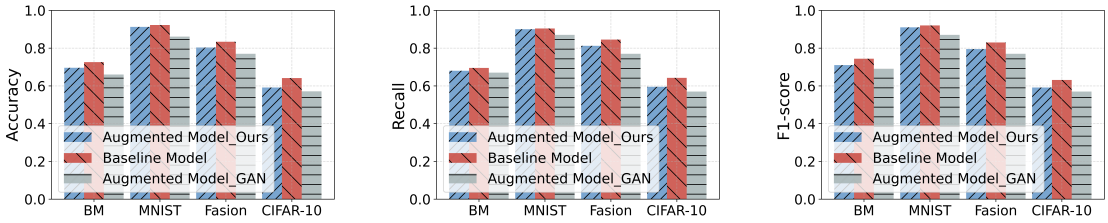
ACE [45]: This is the state-of-the-art model poisoning attack for federated learning. Specifically, they enable malicious clients to manipulate their local model parameters to falsely appear as high contributors. In the VSL scenario, we adapt this method by designing targeted manipulations of local embeddings to exploit the contribution evaluation metrics used by the server.

B. Effectiveness of Data Augmentation

We first assess the effectiveness of data augmentation, by comparing the accuracy, recall and F1-score of three models: our augmented model, which is trained on the augmented dataset, the augmented model, trained on augmented dataset using a GAN-based method, and the baseline model which is trained on a clean dataset of the same size as X_{aug}.

As illustrated in Fig. 4, the augmented model consistently exhibits comparable performance to the baseline model across all evaluation metrics and datasets. This close alignment highlights that our data augmentation strategy effectively enhances the trusted clean dataset without introducing significant noise or bias. Notably, the differences in accuracy and recall across the two models remain within 2%, and F1-scores follow a similar trend, further validating the semantic consistency and utility of the generated data. Even on CIFAR-10, which poses challenges due to high inter-class visual similarity, the performance gap between the two models is negligible, underscoring the generalizability of the approach.

In contrast, the GAN-based augmented model shows a slight but consistent degradation across most datasets and metrics.

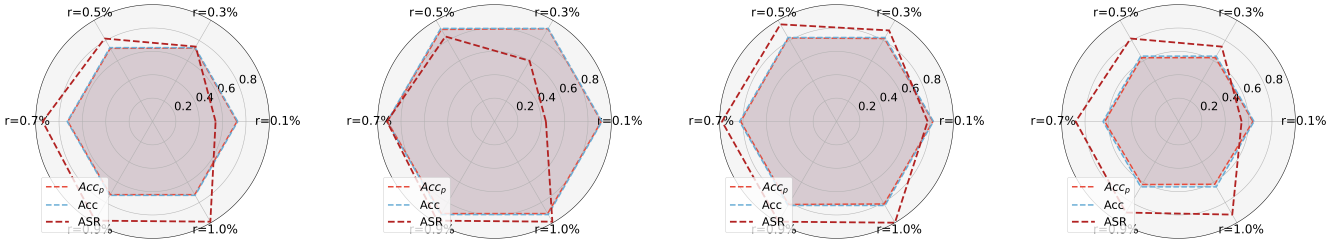


(a) Accuracy (b) Recall (c) F1-score

Fig. 4: Effectiveness of data augmentation.

TABLE II: Comparison analysis with baseline attacks across different datasets.

Dataset	No Attack	SPA		LFA		TCLA		MPA		ACE		TPA-VSL	
	Acc (%)	ASR	Acc_p(%)	ASR	Acc_p(%)	ASR	Acc_p(%)	ASR	Acc_p(%)	ASR	Acc_p(%)	ASR	Acc_p(%)
Adult	81.56 ± 0.17	0.61	81.13 ± 0.26	0.59	80.92 ± 0.35	0.64	80.56 ± 0.30	0.71	80.89 ± 0.33	0.78	81.08 ± 0.23	**0.84**	81.09 ± 0.30
BM	72.58 ± 0.30	0.68	72.36 ± 0.35	0.64	72.25 ± 0.34	0.66	72.16 ± 0.28	0.68	72.22 ± 0.28	0.74	72.20 ± 0.25	**0.82**	72.35 ± 0.28
MNIST	92.23 ± 0.14	0.36	91.17 ± 0.26	0.40	91.18 ± 0.18	0.48	91.06 ± 0.24	0.42	91.33 ± 0.18	0.64	91.38 ± 0.20	**0.84**	91.41 ± 0.16
Fashion	83.35 ± 0.12	0.54	82.79 ± 0.18	0.50	82.20 ± 0.15	0.62	82.04 ± 0.28	0.82	82.40 ± 0.15	0.88	82.34 ± 0.16	**0.94**	82.52 ± 0.14
CIFAR-10	64.13 ± 0.35	0.58	63.56 ± 0.41	0.62	63.42 ± 0.34	0.68	63.54 ± 0.30	0.72	63.78 ± 0.30	0.74	63.75 ± 0.28	**0.80**	63.46 ± 0.34
Average	78.77 ± 0.22	0.55	78.20 ± 0.29	0.55	77.99 ± 0.27	0.62	77.87 ± 0.28	0.67	78.12 ± 0.24	0.75	78.15± 0.22	**0.85**	78.16 ± 0.24



(a) BM (b) MNIST (c) Fashion (d) CIFAR-10

Fig. 5: Impact of modification rate on ASR, Acc_p, and Acc under the two-party scenario.

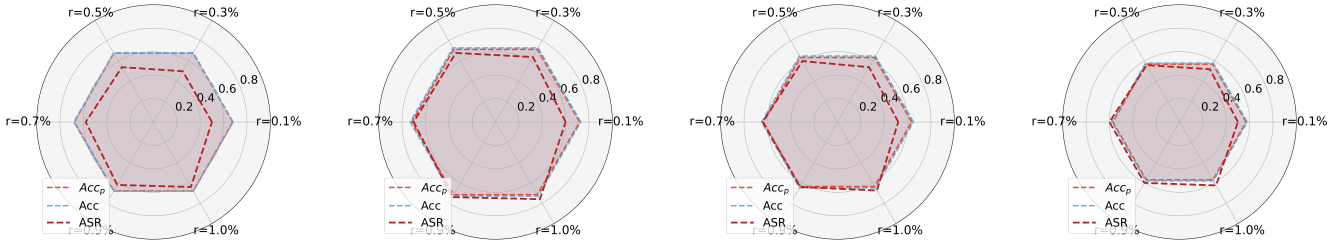
This performance drop is particularly evident on more complex datasets such as Fashion-MNIST and CIFAR-10, suggesting that GAN-based augmentation may introduce distributional artifacts that negatively affect downstream performance. These results highlight that not all data augmentation strategies are equally effective in preserving task-relevant semantics.

These findings are particularly critical for TPA-VSL, as the effectiveness of the mimic model training depends heavily on the representativeness of the augmented dataset. By generating data that closely mirrors the distribution of the original training data held by other clients, the attacker is able to construct a surrogate embedding model that approximates the behavior of the target embedding model.

C. Effectiveness of TPA-VSL

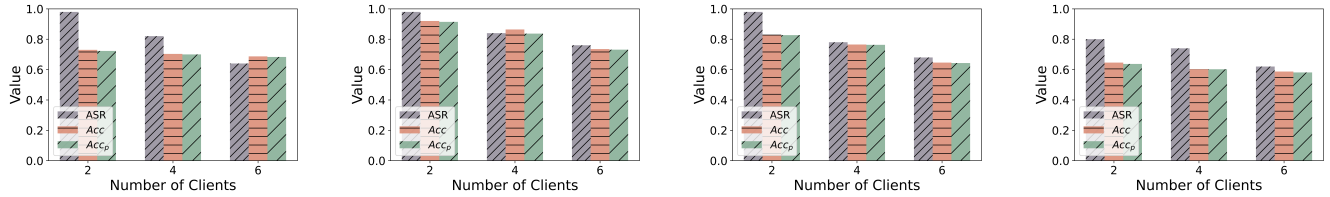
We first conduct a comparative study to assess the attack effectiveness of TPA-VSL in comparison to baseline attacks in a two-party scenario. The performance comparison of TPA-VSL with SPA, LFA, MPA, and ACE on different datasets is summarized in Table II. We evaluate ASR, Acc and Acc_p for a given dataset. We set the ratio of auxiliary data to 5%. For the TPA-VSL attack, we set the modification rate to 0.5%. For the SPA and MPA, we set the ratio of sample replacement to 0.5%. For the LFA, we set the ratio of label-flipped samples to 0.5%. From the result, we can observe that the no-attack scenario yields an accuracy of 78.77% on average. The SPA, LFA, TCLA, MPA, and ACE result in an average of 78.2%, 77.99%,77.87%, 78.12%, and 78.15% accuracy, respectively, with ASRs of 0.55, 0.55, 0.62, 0.67, and 0.75.

Specifically, on the MNIST dataset, the SPA reduces the accuracy to 91.17%, with an ASR of 0.36. The LFA and MPA show slight reductions in accuracy to 91.18% and 91.33%, respectively, with ASRs of 0.40 and 0.42. For the Fashion dataset, the SPA reduces the accuracy to 82.79%, with an ASR of 0.54. The LFA and MPA achieve 82.20% and 82.40% accuracy, respectively, with ASRs of 0.50 and 0.82. The TPA-VSL attack achieves the highest ASR of 0.94, with an accuracy of 82.52%. On the CIFAR-10 dataset, the SPA reduces the accuracy to 63.56%, with an ASR of 0.58. The LFA and MPA show 63.42% and 63.78% accuracy, respectively, with ASRs of 0.62 and 0.72. The TPA-VSL attack demonstrates the highest ASR of 0.85, with an average accuracy of 77.43%, which consistently indicates that TPA-VSL achieves the best attack performance among all the baseline attacks.



(a) BM (b) MNIST (c) Fashion (d) CIFAR-10

Fig. 6: Impact of modification rate on ASR, Acc_p, and Acc under the multi-party scenario.



(a) BM (b) MNIST (c) Fashion (d) CIFAR-10

Fig. 7: Effectiveness of TPA-VSL under the multi-party scenario.

TABLE III: Impact of Ratio of Auxiliary Data.

Dataset	**Ori**	$\|X\|_t$									
		1%		**5%**		**10%**		**15%**		**20%**	
	Acc **(%)**	**ASR**	Acc **(%)**	**ASR**	Acc_p**(%)**	**ASR**	Acc_p**(%)**	**ASR**	Acc_p**(%)**	**ASR**	Acc_p**(%)**
Adult	81.56 ± 0.17	0.71	75.95 ± 0.37	0.79	78.45 ± 0.33	0.83	80.01 ± 0.29	0.86	80.98 ± 0.26	0.88	81.35± 0.28
BM	72.58 ± 0.30	0.74	65.02 ± 0.42	0.78	67.87 ± 0.40	0.82	72.01 ± 0.38	0.82	72.31 ± 0.34	0.84	72.42 ± 0.35
MNIST	92.23 ± 0.14	0.78	85.10 ± 0.29	0.82	88.05 ± 0.25	0.84	91.27 ± 0.20	0.86	91.89 ± 0.22	0.86	91.88 ± 0.20
Fashion	83.35 ± 0.12	0.85	77.05 ± 0.85	0.90	79.01 ± 0.22	0.94	82.21 ± 0.18	0.96	83.33 ± 0.18	0.96	83.23 ± 0.23
CIFAR-10	64.13 ± 0.35	0.72	58.03 ± 0.45	0.76	60.32 ± 0.42	0.80	63.48 ± 0.38	0.84	63.88 ± 0.30	0.84	63.86 ± 0.32
Average	78.77 ± 0.21	0.76	72.23 ± 0.47	0.81	74.74 ± 0.25	0.85	77.79 ± 0.28	0.87	78.47 ± 0.26	0.876	78.55 ± 0.27

D. Impact of Modification Rate

In this section, we evaluate the impact of modification rate in TPA-VSL for different datasets. We set the ratio of auxiliary data to 5%. Fig. 5 illustrates the effect of varying modification rates on ASR, Acc and Acc_p. Specifically, we can observe that as the modification rate increases from 0.1% to 1.0%, the ASR shows a significant upward trend, indicating that higher modification rates lead to more successful attacks for different datasets. Conversely, both Acc and Acc_p remain relatively stable, suggesting that the clean and overall accuracy are not substantially affected by the modification rate.

Specifically, for the BM dataset, as the modification rate increases, the ASR rises significantly from approximately 0.6 to nearly 1.0. Meanwhile, Acc and Acc_p remain almost unchanged, staying around 0.8 and 0.85, respectively. Similarly, for other datasets like CIFAR-10, the ASR increases from about 0.5 to over 0.9, showing a clear upward trend. Acc and Acc_p also remain stable when the modification rate reaches 1.0%. Overall, the results indicate that while the ASR increases with higher modification rates across all datasets, the Acc and Acc_p are generally stable for different datasets. This demonstrates the effectiveness of TPA-VSL in maintaining accuracy while varying the modification rate.

Additionally, Fig. 6 illustrates the effect of varying modification rates on ASR, Acc and Acc_p under multi-party scenario, where the number of clients is fixed at 6. We can observe that ASR are comparatively lower across all modification rates, reflecting increased robustness due to the model update diversity in multi-party setting. However, the overall trend remains consistent. Increasing modification rate boosts ASR while only marginally degrading Acc and Acc_p. Notably, even at r=1.0%, the performance degradation in Acc and Acc_p is minimal, indicating that the attack remains stealthy. These results demonstrate the robustness of TPA-VSL across varying modification rates and its capacity to maintain efficacy without significantly compromising utility.

E. Performance on Multi-Party Setting

In this section, we evaluate the effectiveness of TPA-VSL in a multi-party scenario. TPA-VSL can be extended to VSL with more than 2 clients. We conduct experiments on BM, MNIST, Fashion, and CIFAR-10 datasets with 2, 4, 6 clients. As shown in Fig. 7, the ASR remains relatively high when the number of clients increases. We observe a noticeable drop

in ASR when the number of participants exceeds 6, which is reasonable since the attacker controls a smaller ratio of the embedding vector uploaded to the server. Additionally, when the number of clients increases, the Acc and Acc_p have slight drops due to the data splitting setting. On average, as the number of clients increases, the ASR can achieve 60%, which proves the effectiveness of TPA-VSL under multiple clients scenario. Specifically, we notice that the ASR on MNIST and Fashion datasets remains relatively stable even with 6 clients, while the ASR on CIFAR-10 dataset experiences a slight drop. This is largely due to the increased complexity of the CIFAR-10 dataset, which makes it more challenging to attack. Furthermore, the MNIST dataset exhibits the highest ASR across all scenarios, indicating that TPA-VSL is particularly effective in this setting.

F. Impact of Ratio of Auxiliary Data

In this section, we evaluate the impact of varying ratios of auxiliary data on TPA-VSL's attack performance, where the auxiliary data ratio ranges from 1% to 20%. The results presented in Table. III show that the accuracy Acc remains stable, ranging from 72.23% to 78.55% on average, with fluctuations of less than 2% as the ratio of auxiliary data increases from 1% to 20%. The fluctuations are within a narrow margin suggesting that the embedding model manipulation introduced by TPA-VSL does not significantly compromise the overall model utility.

More importantly, we explicitly evaluate the extreme low-data regime. As shown in Table III, even when only 1% of auxiliary data is available, TPA-VSL is still able to achieve non-trivial ASR across all datasets, although with a moderate reduction in both ASR and Acc_p. This performance drop is expected, as extremely limited auxiliary data constrain the surrogate embedding model's ability to approximate the target representation space. Nevertheless, the attack remains effective and does not collapse, demonstrating that TPA-VSL does not rely on large amounts of auxiliary data.

Moreover, the ASR under the TPA-VSL attack remains relatively stable, indicating that our model is resilient to attacks even with varying amounts of auxiliary data. Notably, even when the ratio of $|\boldsymbol{X}|_t$ reaches 5%, the Acc and ASR can still maintain a high level, demonstrating the robustness of our approach. Meanwhile, the ASR also exhibits a slight increase as the ratio of $|\boldsymbol{X}|_t$ increases, and it can be seen that at 5%, the ASR has already achieved a relatively attack effect. These findings demonstrate that our data augmentation component is robust and effective, performing well under different ratios of auxiliary data. This analysis is crucial, as in real-world scenarios, an adversary may only have limited access to auxiliary data that conducts the attack.

G. Performance on Multi-Target Setting

In this section, we evaluate the effectiveness of TPA-VSL in simultaneously attacking multiple targets across four different datasets. By exploring scenarios with multiple targets, we aim to provide a more comprehensive understanding of TPA-VSL's performance in practical federated learning environments. In our experiments, we initially set the ratio of the trusted clean dataset to 10% and assess the effectiveness of TPA-VSL under these conditions. The results are illustrated in Fig. 8, which show a slight decrease in the ASR across the four datasets as the number of targets increases from 1 to 10. On average, there is an approximate 8% reduction in ASR. This degradation is expected, as manipulating the embedding model to simultaneously attack multiple targets introduces additional optimization challenges among targets in federated learning scenarios. This analysis underscores the importance of considering multiple target scenarios to better understand the robustness and adaptability of TPA-VSL in real-world applications.

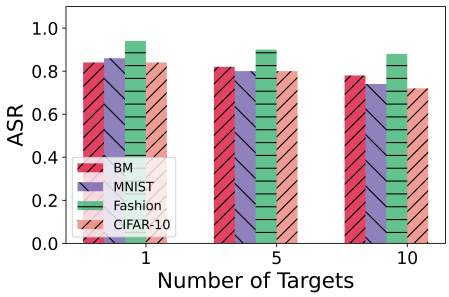
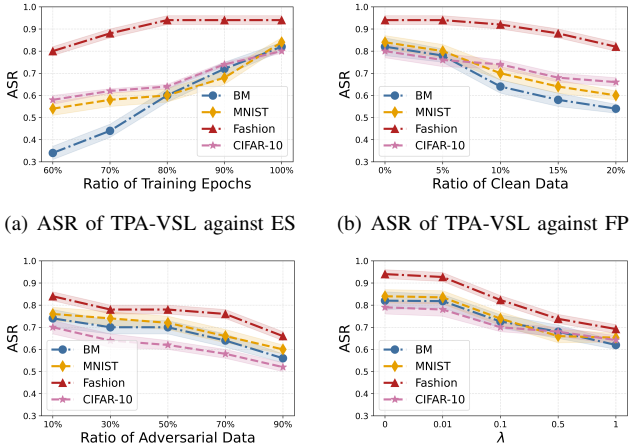


Fig. 8: ASR of TPA-VSL under different numbers of targets.



(a) ASR of TPA-VSL against ES (b) ASR of TPA-VSL against FP

(c) ASR of TPA-VSL against FAT (d) ASR of TPA-VSL against ER

Fig. 9: ASR of TPA-VSL against baseline defenses.

H. Countermeasures

In this section, we discuss existing defenses against poisoning attacks and evaluate their effectiveness against our TPA-VSL attack.

1) Leveraging Existing Defenses Against Poisoning Attacks: To date, few defense strategies have been explicitly designed for the VSL setting, leaving this area relatively unexplored. To bridge this gap, we adapt and extend two simple yet effective defense mechanisms to the VSL setting.

- Early Stopping (ES): Early stopping is a widely adopted defense strategy that limits unnecessary training iterations, thereby enhancing the robustness of machine learn-

ing models. Given that our TPA-VSL attack relies on extensive training iterations to effectively compromise the model, implementing early stopping can potentially mitigate the impact of TPA-VSL. If the server model is trained for fewer iterations, it may reduce the attack's effectiveness by preventing the model from fully adapting to the poisoned data.

- Fine-Pruning (FP): Fine-pruning [46] serves as a post-training defense against poisoning attacks. It aims to alleviate the detrimental effects of a potentially compromised model by fine-tuning it with a subset of clean training data. In the context of VSL, defenders can leverage this approach by incorporating clean data to refine and adjust the server model, thereby diminishing the influence of the poisoned samples. This method not only restores model integrity but also enhances overall performance by ensuring that the model's predictions remain accurate and reliable.
- Federated adversarial training (FAT): FAT [47] is designed to enhance model robustness against adversarial attacks in FL setting. Since the poisoned embedding vectors generated by TPA-VSL can be regarded as anomalous model updates, FAT may serve as a potential defense mechanism by identifying and mitigating these anomalous updates during the model aggregation process.
- Embedding Regularization (ER): ER [48] is designed to strengthen the robustness against embedding-level manipulation attacks. Specifically, ER penalizes embedding drift by imposing an L_2-norm regularization on the client-side embedding representations. The overall objective is formulated as:

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda \mathcal{L}_{\text{ER}}, \tag{11}$$

where $\mathcal{L}_{\text{task}}$ denotes the original task loss, λ controls the strength of regularization, and the ER term is defined as the ℓ_2 penalty:

$$\mathcal{L}_{\text{ER}} = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{w}_i\|_2^2, \tag{12}$$

with $\mathbf{w}_i = f_w(\mathbf{x}_i)$ being the embedding vector generated by the client-side embedding model $f_w(\cdot)$ for sample $\mathbf{x}_i$ and N the number of training samples in an epoch.

2) Performance Results: Fig. 9(a) demonstrates the performance of the ES defense against TPA-VSL under different ratios of training epochs. We set $|\boldsymbol{X}|_t = 10\%$ and the modification rate to 0.5%. The results show that ES achieves an average reduction of 35% in ASR when the ratio decreases from 100% to 60%. Notably, the ASR reduction is most pronounced on the BM dataset as the ratio decreases. This is largely because BM is a simpler dataset compared to others, which makes it more susceptible to underfitting when trained with fewer epochs. However, it is indicated that ES also reduces the prediction accuracy of the model. Thus, this method sacrifices the utility of the model. We further observe that reducing the number of training epochs consistently degrades test accuracy. While early stopping can partially reduce ASR by limiting the server model's ability to internalize manipulated embeddings, it does so at the cost of noticeable utility loss, highlighting an inherent robustness–utility trade-off rather than a cost-free defense.

TABLE IV: Time overhead for TPA-VSL's components (s).

Time overhead \ Dataset / Component	BM	MNIST	Fashion	CIFAR-10
MMT	309.5	426.6	409.1	1023.5
EMM	31.2	62.1	47.9	89.5

Fig. 9(b) demonstrates the performance of the FP defense against TPA-VSL under different ratios of clean data. We report the ASR of the FP defense when ratios change. We can observe that the ASR is reduced by 20% on average when the ratio of clean data increases to 20%. This suggests that the FP defense can mitigate TPA-VSL by leveraging a larger proportion of clean data. However, manually gathering a large amount of clean data is time-consuming and may not be feasible. Notably, as the proportion of clean data increases, the Fashion dataset's ASR decreases at the slowest rate. This is because the Fashion dataset is more complex and diverse, making it more challenging to attack.

The effectiveness of FAT is illustrated in Fig. 9(c), where increasing the proportion of adversarial data from 10% to 90% results in a substantial reduction in ASR across all datasets. Notably, CIFAR-10 and BM demonstrate the most significant improvements, with ASR declining from 0.75 to 0.55 and from 0.70 to 0.52, respectively. In contrast, the Fashion dataset maintains an ASR above 0.65 even under aggressive adversarial training, underscoring the robustness of TPA-VSL. These observations indicate that while FAT enhances model resilience by mitigating adversarial perturbations, it remains insufficient to mitigates targeted embedding manipulation within vertical split learning frameworks.

Fig. 9(d) shows the ASR of TPA-VSL under different embedding regularization parameters λ. As λ increases, the ASR consistently decreases across all datasets, indicating that L_2-based embedding regularization effectively suppresses abnormal embedding drift caused by poisoning. Overall, ER provides a practical and VSL-compatible defense with a clear robustness and utility trade-off controlled by λ.

I. Evaluation on Run-time Overhead

Lastly, we evaluate the run-time overhead of TPA-VSL, which was tested on an NVIDIA Geforce RTX 4090. The experimental results are detailed in Table IV. The run-time for TPA-VSL is divided into two main parts: MMT and EMM.

In the MMT phase, we evaluate the time required to generate data using a 20% portion of trusted clean data. Notably, TPA-VSL takes the longest time with the CIFAR-10 dataset, requiring approximately 1023.5 seconds. In contrast, the BM dataset requires significantly less time than CIFAR-10. This difference is primarily due to the complexity and size of the datasets, as larger datasets like CIFAR-10 naturally require more computational resources and time for processing. It is evident that the run-time overhead is significantly influenced by the size and complexity of the dataset.

In the EMM phase, we set the modification rate to 0.5% and record the time needed to manipulate the embedding model. For the BM, MNIST, and Fashion datasets, TPA-VSL requires only 31.2 seconds, 62.1 seconds, and 47.9 seconds, respectively. The significantly reduced time in the EMM phase can be attributed to the nature of the task, which involves merely altering the features of the data. This process is inherently faster as it eliminates the extensive computations required for sample generation and model training.

VI. Conclusion

Existing poisoning attacks in vertical split learning typically inject fixed poison patterns into the data or the model, which can be easily detected by analyzing their negative impact on the model. In this paper, we propose a stealthy poisoning attack in vertical split learning, dubbed TPA-VSL that does not require modification of the test data and lacks trigger patterns. We present a novel two-step method that first utilizes the diffusion model to obtain sufficient training data and further trains an effective target mimic model. Additionally, we manipulate the embedding model by maximizing the similarity between the embedding vector of the targeted sample and those of samples from the desired class. Thus, we enable the targeted sample's misclassification. Extensive experimental results on five real-world datasets show that TPA-VSL achieves a high ASR, verifying the effectiveness of our TPA-VSL attack.

References

[1] S. L. Pardau, "The california consumer privacy act: Towards a european-style privacy regime in the united states," *J. Tech. L. & Pol'y*, vol. 23, p. 68, 2018.

[2] J. Zhang, X. Liu, J. Niu, H. Yang, W. Lin, and X. Wu, "The aggregated model is a confounder: Enabling deconfounded federated learning for ood generalization," in *Proceeding of IEEE INFOCOM*. IEEE, 2026, pp. 1–10.

[3] H. Zeng, J. Lou, K. Li, C. Wu, G. Xue, Y. Luo, F. Cheng, W. Zhao, and J. Li, "Esfl: Accelerating poisonous model detection in privacy-preserving federated learning," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–15, 2025.

[4] G. Chen, W. Wang, Y. Wu, C. Li, G. Xu, S. Ji, T. Li, M. Shen, and Y. Han, "Robustpfl: Robust personalized federated learning," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–16, 2025.

[5] X. Wu, J. Niu, X. Liu, G. Zhu, S. Tang, W. Lin, and J. Cao, "The diversity bonus: Learning from dissimilar clients in personalized federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 36, no. 10, pp. 18 613–18 627, 2025.

[6] Y. Lu, L. Yang, H.-R. Chen, J. Cao, W. Lin, and S. Long, "Federated class-incremental learning with dynamic feature extractor fusion," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 12 969–12 982, 2024.

[7] J. Tu, J. Huang, L. Yang, and W. Lin, "Personalized federated learning with layer-wise feature transformation via meta-learning," *ACM Transactions on Knowledge Discovery from Data*, vol. 18, no. 4, pp. 1–21, 2024.

[8] R. Joeres, D. B. Blumenthal, and O. V. Kalinina, "Data splitting to avoid information leakage with datasail," *Nature Communications*, vol. 16, no. 1, p. 3337, 2025.

[9] Y. Oh, J. Lee, C. G. Brinton, and Y.-S. Jeon, "Communication-efficient split learning via adaptive feature-wise compression," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2025.

[10] W. Lin, H. Lan, H. He, and B. Li, "Graph-relational federated learning: Enhanced personalization and robustness," *IEEE Transactions on Dependable and Secure Computing*, vol. 22, no. 5, pp. 5773–5785, 2025.

[11] J. Liu, X. Lyu, Q. Cui, and X. Tao, "Similarity-based label inference attack against training and inference of split learning," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 2881–2895, 2024.

[12] M. Ye, W. Shen, B. Du, E. Snezhko, V. Kovalev, and P. C. Yuen, "Vertical federated learning for effectiveness, security, applicability: A survey," *ACM Computing Surveys*, vol. 57, no. 9, pp. 1–32, 2025.

[13] Y. Liu, Y. Kang, T. Zou, Y. Pu, Y. He, X. Ye, Y. Ouyang, Y.-Q. Zhang, and Q. Yang, "Vertical federated learning: Concepts, advances, and challenges," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 7, pp. 3615–3634, 2024.

[14] D. Romanini, A. J. Hall, P. Papadopoulos, T. Titcombe, A. Ismail, T. Cebere, R. Sandmann, R. Roehm, and M. A. Hoeh, "Pyvertical: A vertical federated learning framework for multi-headed splitnn," *arXiv preprint arXiv:2104.00489*, 2021.

[15] C. Thapa, P. C. M. Arachchige, S. Camtepe, and L. Sun, "Splitfed: When federated learning meets split learning," in *Proceedings of AAAI*, vol. 36, no. 8, 2022, pp. 8485–8493.

[16] S. Li, D. Yao, and J. Liu, "Fedvs: Straggler-resilient and privacy-preserving vertical federated learning for split models," in *Proceedings of ICML*, 2023, pp. 20 296–20 311.

[17] M. Ye, W. Shen, E. Snezhko, V. Kovalev, P. C. Yuen, and B. Du, "Vertical federated learning for effectiveness, security, applicability: A survey," *arXiv preprint arXiv:2405.17495*, 2024.

[18] X. Liu, Y. Deng, and T. Mahmoodi, "Wireless distributed learning: A new hybrid split and federated learning approach," *IEEE Transactions on Wireless Communications*, vol. 22, no. 4, pp. 2650–2665, 2022.

[19] B. Gu, A. Xu, Z. Huo, C. Deng, and H. Huang, "Privacy-preserving asynchronous vertical federated learning algorithms for multiparty collaborative learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 11, pp. 6103–6115, 2021.

[20] Y. Bai, Y. Chen, H. Zhang, W. Xu, H. Weng, and D. Goodman, "{VILLAIN}: Backdoor attacks against vertical split learning," in *Proceedings of USENIX Security Symposium*, 2023, pp. 2743–2760.

[21] F. Yu, B. Zeng, K. Zhao, Z. Pang, and L. Wang, "Chronic poisoning: Backdoor attack against split learning," in *Proceedings of AAAI*, vol. 38, no. 15, 2024, pp. 16 531–16 538.

[22] Y. Pu, Z. Ding, J. Chen, C. Zhou, Q. Li, C. Hu, and S. Ji, "A stealthy backdoor attack for without-label-sharing split learning," *arXiv preprint arXiv:2405.12751*, 2024.

[23] Y. He, Z. Shen, J. Hua, Q. Dong, J. Niu, W. Tong, X. Huang, C. Li, and S. Zhong, "Backdoor attack against split neural network-based vertical federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 748–763, 2023.

[24] M. Naseri, Y. Han, and E. de Cristofaro, "Badvfl: Backdoor attacks in vertical federated learning," in *Proceedings of IEEE Symposium on Security and Privacy (SP)*, 2024, p. 8.

[25] W. Shen, W. Huang, G. Wan, and M. Ye, "Label-free backdoor attacks in vertical federated learning," in *Proceedings of AAAI*, vol. 39, no. 19, 2025, pp. 20 389–20 397.

[26] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," in *Proceedings of NeurIPS*, vol. 31, 2018.

[27] D. Zhu, J. Chen, X. Zhou, W. Shang, A. E. Hassan, and J. Grossklags, "Vulnerabilities of data protection in vertical federated learning training and countermeasures," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 3674–3689, 2024.

[28] C. Fu, X. Zhang, S. Ji, J. Chen, J. Wu, S. Guo, J. Zhou, A. X. Liu, and T. Wang, "Label inference attacks against vertical federated learning," in *Proceedings of USENIX Security Symposium*, 2022, pp. 1397–1414.

[29] L. Zhang, X. Gao, Y. Li, and Y. Liu, "Functionality and data stealing by pseudo-client attack and target defenses in split learning," *IEEE Transactions on Dependable and Secure Computing*, vol. 22, pp. 84–100, 2024.

[30] B. Tajalli, O. Ersoy, and S. Picek, "On feasibility of server-side backdoor attacks on split learning," in *Proceedings of IEEE Security and Privacy Workshops (SPW)*, 2023, pp. 84–93.

[31] J. Chen, Z. Lin, W. Lin, W. Shi, X. Yin, and D. Wang, "Fedmua: Exploring the vulnerabilities of federated learning to malicious unlearning attacks," *IEEE Transactions on Information Forensics and Security*, vol. 20, pp. 1665–1678, 2025.

[32] Z. Xiang, T. Wang, W. Lin, and D. Wang, "Practical differentially private and byzantine-resilient federated learning," *Proceedings of the ACM on Management of Data*, vol. 1, no. 2, pp. 1–26, 2023.

[33] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proceedings of ICML*, 2018, pp. 5650–5659.

[34] B. Zhao, P. Sun, T. Wang, and K. Jiang, "Fedinv: Byzantine-robust federated learning by inversing local model updates," in *Proceedings of AAAI*, vol. 36, no. 8, 2022, pp. 9171–9179.

[35] P. Zhao, J. Jiang, and G. Zhang, "Fedsuper: A byzantine-robust federated learning under supervision," *ACM Transactions on Sensor Networks*, vol. 20, no. 2, pp. 1–29, 2024.

[36] S. Shen, S. Tople, and P. Saxena, "Auror: Defending against poisoning attacks in collaborative deep learning systems," in *Proceedings of the 32nd annual conference on computer security applications*, 2016, pp. 508–519.

[37] D. Cao, S. Chang, Z. Lin, G. Liu, and D. Sun, "Understanding distributed poisoning attack in federated learning," in *Proceedings of IEEE 25th international conference on parallel and distributed systems (ICPADS)*, 2019, pp. 233–239.

[38] C. Xie, M. Chen, P.-Y. Chen, and B. Li, "Crfl: Certifiably robust federated learning against backdoor attacks," in *Proceedings of ICML*, 2021, pp. 11 372–11 382.

[39] X. Cao, Z. Zhang, J. Jia, and N. Z. Gong, "Flcert: Provably secure federated learning against poisoning attacks," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3691–3705, 2022.

[40] D. P. Kingma, "Auto-encoding variational bayes," in *Proceedings of ICLR*, 2014.

[41] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Proceedings of NeurIPS*, vol. 27, 2014.

[42] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *Proceedings of ICML*, 2015, pp. 2256–2265.

[43] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *Proceedings of ICML*, 2021, pp. 8162–8171.

[44] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *Proceedings of ICML*, 2019, pp. 634–643.

[45] Z. Xu, F. Jiang, L. Niu, J. Jia, B. Li, and R. Poovendran, "{ACE}: A model poisoning attack on contribution evaluation methods in federated learning," in *Proceedings of USENIX Security Symposium*, 2024, pp. 4175–4192.

[46] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *Proceedings of International symposium on research in attacks, intrusions, and defenses*, 2018, pp. 273–294.

[47] G. Zizzo, A. Rawat, M. Sinn, and B. Buesser, "Fat: Federated adversarial training," in *Proceedings of NeurIPS Workshop on SpicyFL*, 2020.

[48] F. Li, K. Li, H. Wu, J. Tian, and J. Zhou, "Towards robust learning via core feature-aware adversarial training," *IEEE Transactions on Information Forensics and Security*, 2025.