URVFL: Undetectable Data Reconstruction Attack on Vertical Federated Learning

Duanyi Yao*, Songze Li†‡, Xueluan Gong§, Sizai Hou*, Gaoning Pan¶

* The Hong Kong University of Science and Technology

† Southeast University

‡ Engineering Research Center of Blockchain Application, Supervision and Management (Southeast University), Ministry of Education

§ Wuhan University

¶ Hangzhou Dianzi University

***Abstract*—Vertical Federated Learning (VFL) is a collaborative learning paradigm designed for scenarios where multiple clients share disjoint features of the same set of data samples. Albeit a wide range of applications, VFL is faced with privacy leakage from data reconstruction attacks. These attacks generally fall into two categories: honest-but-curious (HBC), where adversaries steal data while adhering to the protocol; and malicious attacks, where adversaries breach the training protocol for significant data leakage. While most research has focused on HBC scenarios, the exploration of malicious attacks remains limited.**

Launching effective malicious attacks in VFL presents unique challenges: 1) Firstly, given the distributed nature of clients' data features and models, each client rigorously guards its privacy and prohibits direct querying, complicating any attempts to steal data; 2) Existing malicious attacks alter the underlying VFL training task, and are hence easily detected by comparing the received gradients with the ones received in honest training. To overcome these challenges, we develop URVFL**, a novel attack strategy that evades current detection mechanisms. The key idea is to integrate a *discriminator with auxiliary classifier* that takes a full advantage of the label information and generates malicious gradients to the victim clients: on one hand, label information helps to better characterize embeddings of samples from distinct classes, yielding an improved reconstruction performance; on the other hand, computing malicious gradients with label information better mimics the honest training, making the malicious gradients indistinguishable from the honest ones, and the attack much more stealthy. Our comprehensive experiments demonstrate that** URVFL **significantly outperforms existing attacks, and successfully circumvents SOTA detection methods for malicious attacks. Additional ablation studies and evaluations on defenses further underscore the robustness and effectiveness of** URVFL**. Our code will be available at https://github.com/duanyiyao/URVFL.**

I. Introduction

Federated learning (FL) is an emerging privacy-preserving collaborative learning paradigm, which allows multiple distributed clients to securely train a model without sharing private data [46]. FL can be categorized into horizontal FL (HFL) and vertical FL (VFL) [24]. In HFL, different clients possess distinct sets of training samples, yet all samples share the same feature space. In contrast, VFL is applicable to scenarios where clients have the same sample identifiers but possess disjoint feature spaces, e.g., an finance company may hold investment records for a set of customers, while a bank may have the expenditure details for the same set of customers. Due to its capability to integrate diverse data sources, VFL is increasingly gaining attention and has been applied in various fields, including finance [3], healthcare [29], and recommendation systems [45].

In VFL [3], [8], [22], [23], [35], [41], [44], an active client holds both labels and partial features, while multiple passive clients hold disjoint features for the same samples. VFL typically employs a split learning framework [2], [19], [35], [36], and utilizes separate models to process these disjoint features. Specifically, a split VFL system comprises one top model and several bottom models: each passive client operates a bottom model, whereas the active client manages both a bottom model and a top model (see Figure 1). During the training phase, for each batch of training samples, all passive and active clients compute embeddings using their respective bottom models; these embeddings are then transmitted to the active client, who outputs the predictions with the top model and computes the loss from the labels. Following this, the gradients for embeddings are sent back to all clients to update their bottom models.

Data reconstruction attacks on VFL: While VFL emphasizes privacy by sharing embeddings instead of raw data, emerging data reconstruction attacks have revealed potential vulnerabilities that could lead to privacy leakage within VFL models. Such attacks are categorized into honest-but-curious (HBC) and malicious attacks. HBC adversaries stealthily extract private features while adhering to the training protocol, whereas malicious adversaries actively manipulate or violate the protocol to steal private features. Notable methods like GRNA [25] and GIA [18] focus on HBC scenarios. In GRNA, under the

Network and Distributedx System Security (NDSS) Symposium 2025
24-28 February 2025, San Diego, CA, USA
ISBN 979-8-9894372-8-3
https://dx.doi.org/10.14722/ndss.2025.240046
www.ndss-symposium.org

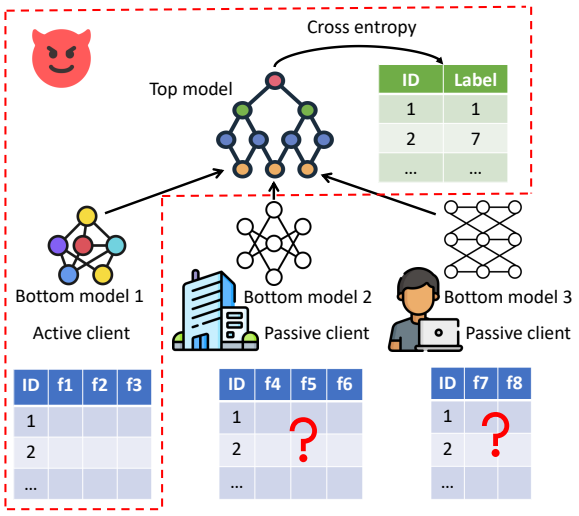


Fig. 1: Illustration of a VFL system with one active client and two passive clients. The area enclosed by the red dashed lines contains the information accessible to and the actions taken by the active client. In a data reconstruction attack, the malicious active client intends to recover the private features of the target passive clients.

assumption that the adversary can query the passive clients' models, target features could be inferred by training a generative regression network to learn correlations between target features and the adversary's partial features, thus enabling generation of target features. GIA [18], presupposing the adversary can inject adversarial features to the target clients, employs optimization techniques to reconstruct the target features. A malicious data reconstruction attack in VFL, called AGN, is developed in [38], where the adversary transmits malicious gradients to passive clients to induce more significant feature leakage.

A class of closely related attacks have also been developed for split learning (SL), which can be viewed as a special case of VFL. In SL, there is a clear division of roles where an active client, holding a top model and labels, but without partial features, collaborates with a single passive client who manages a bottom model and all data samples with complete features. In HBC scenario, attacks like PCAT [12] and SDAR [50] use the top model to train a shadow model on an auxiliary dataset, which is subsequently used to train a decoder for data reconstruction. In the malicious scenario, FSHA [27] manipulates the training objective, generating malicious gradients to achieve more accurate data reconstruction.

Challenges of existing data reconstruction attacks on VFL: While malicious attacks could be more detrimental to privacy leakage in VFL, research works on malicious attacks remains limited. We identify two primary challenges to launch effective malicious attacks: 1) *Distributed features and limited model access.* The heterogeneous nature of feature sources across distributed clients, together with the lack of access to passive clients' local models, makes it difficult for the adversary to infer target features using its own partial features; 2) *Powerful detections.* Existing malicious attacks, such as FSHA and AGN, replace the honest training task with a malicious task, causing the gradients returned to passive clients to appear different from those in honest training, which renders these attacks easily detectable. SOTA detection strategies for malicious attacks, e.g., SplitGuard [6] and Gradient Scrutinizer [10], monitor incoming gradients to identify potential attacks and significantly diminish the success of malicious data reconstruction attacks.

Our attack URVFL: In response to these challenges, we investigate whether a malicious attacker in VFL can circumvent detection measures to maintain effective data reconstruction. We answer this question affirmatively, by developing a novel undetectable data reconstruction attack on VFL (referred to as URVFL). URVFL begins with the adversary pretraining an encoder, a decoder, and its bottom model to achieve high reconstruction performance on an auxiliary data. Following this, the adversary freezes the encoder and bottom model, and trains a discriminator with auxiliary classifier (DAC) [16] that integrates labels information to generate malicious gradients transmitted to passive clients. These malicious gradients can transfer embeddings distribution from the encoder to passive clients' models. In data reconstruction phase, the trained decoder is leveraged to reconstruct target features from the target clients, as these passive clients' models are meticulously guided to mimic the encoder by DAC.

To address challenge 1, URVFL captures correlations between the adversary's partial features and the target features without direct queries or data injections, using DAC to generate malicious gradients that indirectly guide the passive clients' model to leak feature information. To handle challenge 2, we use DAC to incorporate label information into the computation of the malicious loss function, generating malicious gradients almost *indistinguishable* from the gradients received in honest training, which helps URVFL to effectively circumvent current detection strategies. Additionally, these label information helps to better transfer the embedding distribution from the encoder to the passive clients' models compared with a traditional discriminator, and provide extra information for achieving a smaller reconstruction error. As a result, over extensive empirical evaluations, URVFL is shown to achieve better reconstruction performance than SOTA attacks, and successfully circumvents all existing detections against malicious attacks. Further ablation studies and evaluations against defense strategies underscore the robustness and effectiveness of URVFL.

In summary, this paper makes the following main contributions:

1) We introduce URVFL, a novel undetectable data reconstruction attack on VFL, designed to bypass existing detection measures against malicious attacks and effectively steal private features.
2) We develop a DAC as part of URVFL, which:
   - Significantly enhances the effectiveness of embedding distribution transfer and overall attack performance.
   - Generates malicious gradients that are indistinguishable from those produced during honest training.

3) We conduct rigorous evaluations of URVFL using five representative datasets, demonstrating that it:
   - Circumvents SOTA detection strategies against malicious data reconstruction.
   - Outperforms SOTA data reconstruction attacks on VFL, both with and without detection mechanisms in place.
   - Achieves high-quality reconstruction despite defense methods designed for both malicious and HBC attacks in VFL.

II. Background and Related works

In this section, we review SOTA data reconstruction attacks on VFL, and existing detection and defense mechanisms for these attacks.

A. Data reconstruction attacks on VFL

Although VFL safeguards feature privacy by exchanging embeddings and intermediate gradients instead of raw features, recent research has revealed its vulnerability to data reconstruction attacks that can lead to private feature leakage. In HBC setting, the attacks described in [25] and [47] assume that an HBC adversary can query the passive clients' bottom model. During the inference phase, the adversary covertly trains a generative model that is fed random vectors and partial data features available to the adversary, thereby producing synthetic inputs. By minimizing the discrepancy of the bottom models' output between synthetic inputs and target features, the generative model is trained to approximate these features accurately. While the adversary in [25] can access the gradients of the target bottom model directly, the adversary in [47] employs zeroth-order optimization to estimate these gradients, thereby bypassing the need for direct gradient access.

The approach in [18], GIA, operates under a black-box scenario where the adversarial active client lacks direct access to the passive clients' bottom model. Despite this limitation, the adversary can strategically insert a batch of auxiliary data features into the datasets of the passive clients and subsequently collect the resulting output embeddings. The adversary then trains a shadow model using the collected auxiliary features and embeddings pairs. After training, the adversary optimizes noise to closely approximate the target features by querying this shadow model. It is essential to emphasize that these attack methodologies, while effective in reconstructing private data, depend on the adversary having either direct access to the passive clients' models or the capability to inject data into these clients.

In malicious setting, the attack method, AGN [38], utilizes a generator to create fake target features from passive clients' embeddings. These fake features are then evaluated by a discriminator, which distinguishes between real and fake features to generate malicious gradients. These gradients are sent back to the passive clients, aiming to train embeddings that facilitate the reconstruction of target features. Despite the innovative approach of AGN, our evaluations in the section of experiments reveal that the data reconstructed by this method is relatively coarse and remains susceptible to existing defense mechanisms.

Data reconstruction attacks on split learning. A parallel strand of privacy attacks on SL is adaptable to VFL with certain modifications. Unlike VFL which involves multiple clients, SL typically includes only one passive client and an active client managing the bottom and top models, respectively, which can be viewed as a special case of VFL.

For HBC settings, He et al. pioneer privacy attacks on SL in [14], exploring various levels of adversarial knowledge. In a black-box scenario, where the adversary lacks the knowledge of the bottom model but can make queries, an inverse model is used to reconstruct private features. In scenarios without query capabilities, the adversary utilizes a shadow model trained with auxiliary data under the guidance of the top model. Similar with GIA, the shadow model is then used to reconstruct target features through optimization. Unsplit [7] presents a scenario where the adversary knows the bottom model's structure but lacks auxiliary data. This method involves optimizing synthetic features and the shadow model concurrently to reduce disparities between real and synthetic features' embeddings. PCAT [12] depicts an adversary with access to a batch of auxiliary data but unaware of the bottom model's structure and parameters. The strategy involves covertly training a shadow model with the auxiliary data, guided by the top model in each training round. Then an inverse model is trained on the shadow model for data reconstruction. SDAR [50], assuming knowledge of the bottom model's architecture, utilizes auxiliary data and adversarial regularization to train both a shadow model and its inverse model. SDAR employs two discriminators for adversarial regularization: one differentiates between the embeddings of synthetic and genuine features, while the other identifies the data's origin, whether it comes from the inverse model or the auxiliary data. For malicious setting, the attack method, FSHA [27], replaces the top model with a discriminator for aligning the bottom model with a local encoder. Concurrently, an decoder trained on the encoder can apply to reconstruct target features.

B. Defenses and detections against data reconstruction attacks

Defenses: As data leakage risks in VFL escalate, a variety of defensive strategies have been developed to defend data reconstruction attacks. These defenses, particularly against both HBC and malicious adversary, can be broadly classified into cryptographic methods, perturbation-based methods, and adversarial training methods. Cryptographic strategies include homomorphic encryption (HE) [9], [13], [43], secret sharing (SS) [17], [22], and hashing [30]. Employed to protect the privacy of individual embeddings in VFL, these methods, while effective, often lead to significant computation and communication overheads. Perturbation methods involve adding noise to embeddings to disturb potential reconstruction attacks, as detailed in [3], [15], [39]. Another variation treats noise as trainable parameters, aiming at minimizing privacy leakage without significant degradation of model performance, as explored in [26]. While these methods can mitigate data

reconstruction attacks, they might reduce model accuracy significantly. Adversarial training techniques, as discussed in [33], [37], train models in such a way that embeddings are less likely to leak raw features information. Although these methods are effective at reducing features leakage, they increase the computational burden and may affect the pefromance of the intended learning tasks.

Detections against malicious attacks: As malicious attacks such as FSHA and AGN substitute the intended training task, causing significant private features leakage, recent advancements in detection methods, notably SplitGuard [6] and Gradient Scrutinizer [10], have been developed to specifically address these types of attacks. SplitGuard operates by requiring passive clients to intermittently replace labels in a batch with randomized labels, referred to as fake batches, during training. The idea behind SplitGuard is that if the top model is genuinely engaged in learning the intended task, the angles and distances between the gradients from fake and regular batches will differ significantly from those between two regular batches. Consequently, passive clients can detect anomalies by analyzing these differences in gradients. Unlike SplitGuard, Gradient Scrutinizer does not necessitate changes to labels. It operates by collecting and analyzing received batch gradients with identical and differing labels during each training round. The key finding is that during honest training, the gradient distances for embeddings with the different label should be greater than those for embeddings with same labels. By comparing these gradient distances, Gradient Scrutinizer can effectively detect deviations from honest training patterns, indicating potential attacks.

Differences between defenses and detections: While defenses are designed to significantly mitigate the privacy leakage by manipulating the embeddings, i.e., the embeddings are modified to preserve its representation ability as well as the privacy, the detection methods will not change the learning process of embeddings and only observe the variations of exchanged information to detect an attack. Specifically, current detections for data reconstruction attack analyze the difference of gradients between the honest and malicious training to detect the attack. Once there is an attack, the victims can stop the training process to prevent further privacy leakage.

III. Problem Description

A. VFL protocol

In a VFL system, N passive clients collaborate with an active client to train and utilize a VFL model. Each passive client, denoted as $n \in [N]$, operates a bottom model $f_n(\cdot)$. The active client manages a top model $f_0(\cdot)$ and a bottom model $f_a(\cdot)$. The training data $D_{train} := (x_j)_{j=1}^{M}$ is vertically partitioned among $N+1$ clients. For each sample $j \in [M]$, the active client holds both partial features $x_{j,a}$ and the label y_j; while each passive client n, $n \in [N]$, possesses a disjoint portion of the features $x_{j,n}$.

In the training phase, for a selected batch $\mathcal{B}$, each passive client n computes the corresponding embeddings $h_{j,n} = f_n(x_{j,n}), j \in \mathcal{B}$, and sends them to the active client. The active client processes its partial features to produce embeddings $h_{j,a} = f_a(x_{j,a}), j \in \mathcal{B}$. Upon concatenating all batch embeddings, $h_j = [h_{j,a}, h_{j,1}, \ldots, h_{j,N}]$, the active client completes forward propagation through the top model. Loss is computed as $L = \frac{1}{|\mathcal{B}|}\sum_{j\in\mathcal{B}} l(f_0(h_j), y_j)$, for some loss function (e.g., cross entropy) $l(a,b)$ between prediction a and label b. Gradients on the embeddings are sent back to each client for updating their respective bottom models. This process iterates until all models converge.

In the inference phase, when the prediction on a test sample is requested, each client computes an embedding using its bottom model and sends it to the active client. The active client aggregates these embeddings and uses the top model to predict the label. This approach ensures that data privacy is maintained by sharing only the embeddings rather than the raw features, and also enhances computational efficiency by leveraging the split learning mechanism.

B. Threat model

Adversary's capability and knowledge. In our threat model, the active client is treated as a malicious adversary capable of deviating from the standard VFL training protocol. Specifically, this adversary can alter the training task and transmits malicious gradients to other clients, e.g., alter the classification task to a reconstruction task. As the active client, the adversary controls a top model $f_0(\cdot)$, and accesses partial data features $x_{j,a}, j \in [M]$, along with a bottom model $f_a(\cdot)$ (see Figure 1).

Additionally, the adversary possesses an auxiliary dataset, $D_{aux} := (x_i, y_i)_{i=1}^{M_{aux}}$, which shares a similar distribution with the training data but remains distinct from it, i.e., $D_{aux} \cap D_{train} = \emptyset$. Each sample in the auxiliary dataset contains the complete set of features and the corresponding label. This assumption is congruent with previous attacks against SL [12], [27], [50] and presents a more realistic capability compared to those models that assume direct querying of passive clients' bottom models. In practice, adversaries often have access to subsets or samples of target data, which are used for validation or other purposes [32]. In some cases, training data may also include a mix of public and private datasets [34], [49], which makes this assumption more practical.

Goal and metric. The primary objective of the adversary is to reconstruct target clients' data features, denoted as $X_t = (x_{j,n})_{n\in\mathcal{T}_N, j\in\mathcal{T}_X}$, where $\mathcal{T}_N$ is the set of target clients with size $|\mathcal{T}_N| \leq N$, and $\mathcal{T}_X$ is the set of target samples. We denote the reconstructed features as $\tilde{X}_t$, and use the mean squared error (MSE) between X_t and $\tilde{X}_t$ over target samples as the metric to measure the effectiveness of the attack.

IV. Challenges

In this section, we point out the main challenges to achieve effective and stealthy data reconstruction attack on VFL. We also review the current attacks' limitations.

Challenge 1: distributed features and limited model access. VFL involves multiple clients each holding a unique subset of data features and a bottom model. This dispersed and heterogeneous environment raises challenges for the adversary,

to use its own partial features to effectively reconstruct target features from various sources. Additionally, given the distributed nature of bottom models across multiple clients, each guarding its data privacy, any attempt at querying or data injection would trigger an alarm for potential privacy breach and thus, is typically prohibited. This inaccessibility of passive clients' models complicates the adversary's strategy to manipulate them for data leakage.

Challenge 2: powerful detection strategies. Malicious data reconstruction attacks are prone to detection by sophisticated mechanisms such as SplitGuard and Gradient Scrutinizer. One major reason is that current malicious attacks, e.g. AGN and FSHA, alter the honest training task, i.e., replace the honest training loss function with malicious loss function, and thus affecting the transmitted gradients. While such detections are highly sensitive to deviations in training behavior, as they typically monitor gradient flows for anomalies that could indicate malicious activities. To make the attack more stealthy, a natural solution is to integrate the honest training task (e.g., classification) into the malicious task. To evaluate the efficacy of this solution, we employed SplitGuard as a detection mechanism to test against modified malicious attack strategies AGN and FSHA on MNIST dataset, where the training loss is the summation of classifcation loss and the malicious loss. As depicted in Figure 2, the detection scores for both attacks, even with the modified loss, remained below the threshold of 0.9, yielding successful detection. This outcome suggests that simply combining loss functions is insufficient to circumvent current detections.

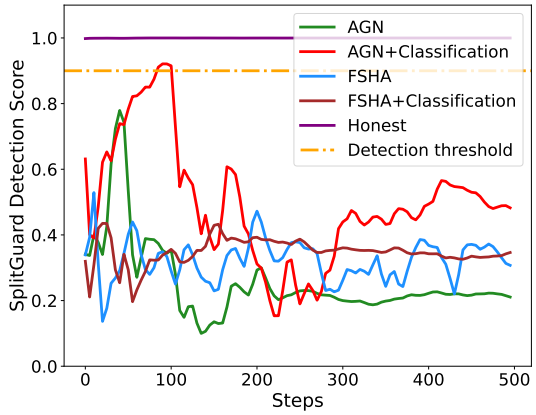


Fig. 2: Comparison of Splitguard scores for two malicious attacks, and their variants with modified loss functions.

Existing HBC data reconstruction attacks on VFL operates under strong, often unrealistic assumptions about adversary's capabilities, such as the ability to query the target model or inject data directly into target clients, which are impractical in real-world scenarios. Malicious attacks like FSHA and AGN completely ignore the label information in their designs, which is the root cause of their detectability, and incorporating the labels into attack designs can potentially further improve the reconstruction performance.

The above challenges steer us toward a compelling research question: *"Under a practical adversary model, can we develop a malicious data reconstruction attack in VFL that successfully circumvents current detection mechanisms while achieving superior reconstruction performance?"* We affirmatively answer this question by developing a novel attack strategy URVFL, which addresses all highlighted challenges.

V. Methodology of URVFL

In this section, we introduce our URVFL strategy. As illustrated in Figure 3, URVFL consists of the following three steps.

1) **Pretraining:** Before initiating VFL training, the adversary sets up three local models consisting of an encoder, a decoder, and a Discriminator with Auxiliary Classifier (DAC). The pretraining step involves the concurrent training of these models. This step is critical for minimizing reconstruction loss and effectively preparing the models for the subsequent steps of the attack.
2) **Malicious gradient generation:** With the onset of VFL training, the adversary freezes the encoder to preserve its learned embeddings and replaces the conventional top model with the DAC. The DAC is instrumental in transferring the embedding distribution from the encoder into the target model and integrating label information through classification processes. This integration ensures that the malicious training is indistinguishable from honest training, enhancing the stealthiness of the attack. Utilizing the malicious loss computed through DAC, the adversary meticulously crafts malicious gradients. These gradients are then dispatched to the target clients to manipulate their models, mimicking the embedding distribution of the encoder.
3) **Data reconstruction:** In the data reconstruction phase, the adversary leverages the decoder to reconstruct private features of target clients, from embeddings uploaded by passive clients and adversary's local embedding.

A. Setup

We first denote a target model $f_p(\cdot) := (f_n(\cdot))_{n\in[N]}$ as the collection of the bottom models of all passive clients. In the setup phase, the adversary prepares the following three networks in replacement of the original top model:

1) Encoder ($f_e(\cdot)$): The encoder $f_e(\cdot)$ is designed to mimic the functionality of the target model $f_p(\cdot)$. The dimensions of the encoder are carefully aligned with those of $f_p(\cdot)$, such that its input dimension equals to the total number of features on the passive clients, and its output dimension equals to the sum of output dimensions of all passive clients' the bottom models.
2) Decoder ($f_d(\cdot)$): The decoder is tailored to reconstruct the target clients' data features. It accepts concatenated embeddings as input and outputs data features of the target clients.
3) Discriminator with Auxiliary Classifier (DAC) ($D(\cdot)$) : The DAC is engineered to classify and discriminate between real

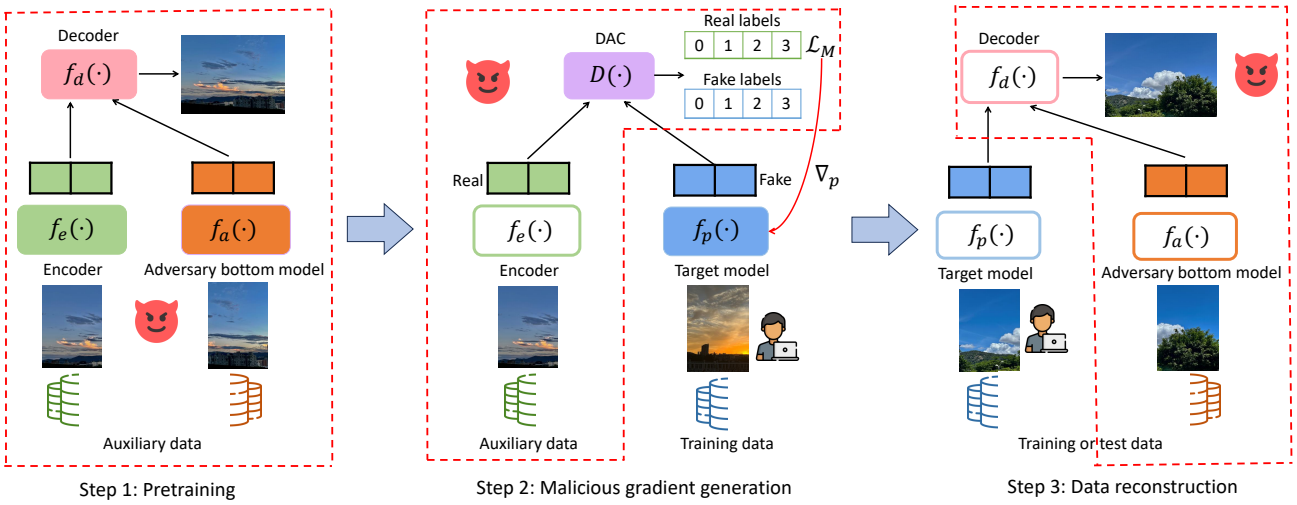


Fig. 3: Workflow of URVFL. The opaque rectangle indicates that the model is being trained, while the transparent rectangle represents that the model is frozen and only utilized in forward propagation.

labels y^+ and fake labels y^-, aiding in the validation of the embeddings' authenticity and malicious gradient generation.

Additionally, the adversary maintains a bottom model $f_a(\cdot)$. This model processes the partial data features available to the adversary, capturing their correlations with the target features.

B. Pretraining

Prior to commencing VFL training, the adversary constructs an encoder-decoder structure consisting of the adversary's bottom model $f_a(\cdot)$, the encoder $f_e(\cdot)$, and the decoder $f_d(\cdot)$. This setup is pretrained on the auxiliary dataset D_{aux}. The objective of this pretraining phase is threefold: to train the decoder to reconstruct target features, to enable the adversary's bottom model to learn the relationship between target and adversary's partial features, and to allow the encoder to capture the representation of passive clients' features for effective reconstruction.

In each iteration of training, the adversary selects a batch of auxiliary data, denoted as $\mathcal{B}_{aux}$. This batch is partitioned into two subsets: the adversary's features $x_{i,a}$ and the passive features $x_{i,p}, i \in \mathcal{B}_{aux}$, which are fed into $f_a(\cdot)$ and $f_e(\cdot)$, respectively. The resulting embeddings, $h_{i,a} = f_a(x_{i,a})$ and $h_{i,p} = f_e(x_{i,p}), i \in \mathcal{B}_{aux}$ are concatenated to form $h_i = (h_{i,a} \| h_{i,p})$. These concatenated embeddings are then forwarded to the decoder, which produces synthetic target features $\tilde{x}_{i,t} = f_d(h_i)$. The primary objective of the adversary is to minimize the reconstruction loss, which is computed as follows:

$$\mathcal{L}_R = \frac{1}{|\mathcal{B}_{aux}|} \sum_{i \in \mathcal{B}_{aux}} MSE(\tilde{x}_{i,t}, x_{i,t}). \tag{1}$$

The pretraining process is detailed in Algorithm 1, where the function Gradient(L, θ) computes the gradients of the loss L with respect to parameters θ; and the function Model_update$(f(\cdot), \nabla_\theta)$ represents the operation of updating the model $f(\cdot)$ using gradients ∇_θ.

Algorithm 1: Pretraining procedure.

Data: Auxiliary dataset $D_{aux} = \{x_{i,a}, x_{i,p}\}_{i=1}^{M_{aux}}$.
Initialization: Encoder $f_e(\cdot)$, decoder $f_d(\cdot)$, adversary's bottom model $f_a(\cdot)$.
while $f_e(\cdot)$, $f_d(\cdot)$, *and* $f_a(\cdot)$ *not converge* **do**
  The adversary select a batch $\mathcal{B}_{aux}$ from D_{aux};
  Compute $h_{i,a} = f_a(x_{i,a}), i \in \mathcal{B}_{aux}$, and $h_{i,p} = f_e(x_{i,p}), i \in \mathcal{B}_{aux}$;
  Compute $\tilde{x}_{i,t} = f_d([h_{i,a} \| h_{i,p}]), i \in \mathcal{B}_{aux}$;
  $\mathcal{L}_R \leftarrow \frac{1}{|\mathcal{B}_{aux}|} \sum_{i \in \mathcal{B}_{aux}} MSE(\tilde{x}_{i,t}, x_{i,t})$;
  $\nabla_e \leftarrow$ Gradient$(\mathcal{L}_R, f_e(\cdot))$;
  $\nabla_a \leftarrow$ Gradient$(\mathcal{L}_R, f_a(\cdot))$;
  $\nabla_d \leftarrow$ Gradient$(\mathcal{L}_R, f_d(\cdot))$;
  $f_e(\cdot) \leftarrow$ Model_update$(f_e(\cdot), \nabla_e)$;
  $f_a(\cdot) \leftarrow$ Model_update$(f_a(\cdot), \nabla_a)$;
  $f_d(\cdot) \leftarrow$ Model_update$(f_d(\cdot), \nabla_d)$;
end

C. Malicious gradient generation

After the pretraining, the adversary intends to transfer the embeddings distribution from the encoder $f_e(\cdot)$ to the target model $f_p(\cdot)$. Since the adversary lacks direct access to modify $f_p(\cdot)$, they resort to transmitting malicious gradients to the passive clients, which are designed to guide $f_p(\cdot)$ to mimic the behavior of $f_e(\cdot)$. In this sequel, for ease of exposition, we describe all operations with $f_p(\cdot)$, which means that the same operation is simultaneously performed on the bottom models of all passive clients.

An intuitive solution involves employing a discriminator to distinguish between embeddings from the encoder (labeled as real) and those produced by the target model (labeled as fake). The target model is then trained to deceive the discriminator into recognizing its embeddings as real, thus aligning its distribution with that of the encoder. This strategy focuses

on minimizing the Jensen–Shannon divergence (JS) [11] between the distributions of the embeddings from the auxiliary and training data, i.e., $\mathrm{JS}(P_{H_{aux}} \| P_{H_{train}})$, which facilitates the transfer of embeddings distribution.

However, a primary concern is that the discriminator approach completely ignores the label information available at the adversary, which may lead to 1) sub-optimal reconstruction performance; and 2) detection by methods that can tell the difference between gradients computed using and without using labels. To address these issues, we employ DAC, $D(\cdot)$, defined in (2), which not only differentiates between real and fake embeddings but also distinguishes the corresponding labels.

$$\min_{D} \mathbb{E}_{h,y\sim P_{H_{aux},Y}} CE(y^+, D(h)) + \mathbb{E}_{h,y\sim P_{H_{train},Y}} CE(y^-, D(h)). \tag{2}$$

Here, $P_{H_{aux},Y}$ and $P_{H_{train},Y}$ represent the joint distributions of embeddings and labels from the auxiliary and training datasets, respectively. The labels y^+ and y^- denote real and fake label y, respectively, and $\mathrm{CE}(\cdot)$ represents the cross-entropy loss.

To align the target model's feature embeddings more closely with the encoder's embedding, the adversary modifies the honest training task into an adversarial task:

$$\min_{h} \mathbb{E}_{h,y\sim P_{H_{train},Y}} CE(y^+, D(h)). \tag{3}$$

Incorporating DAC with this adversarial task minimizes the Kullback–Leibler (KL) divergence between $P_{H_{aux},Y}$ and $P_{H_{train},Y}$, i.e., $\mathrm{KL}(P_{H_{aux},Y} \| P_{H_{train},Y})$. This integration ensures that the adversarial task remains indistinguishable from the honest training task. The correlation of generated gradients with labels not only reduces the detectability, but also enhances the effectiveness of the embedding distribution transfer and thus the reconstruction performance.

Comparison between discriminator and DAC: We carry out empirical studies on the distribution transfer capabilities of discriminator and the DAC, via experiments on the Credit dataset with 2 labels, and the MNIST dataset with 10 labels. We use t-distributed stochastic neighbor embedding (t-SNE) to visualize the embedding distributions, and quantify the alignment of the target model's embeddings with those of the encoder by measuring the average cosine distance between them.

The results, illustrated in Figure 4 and 5, indicate a superior performance of the DAC in distribution transfer over discriminator. The t-SNE visualizations reveal that the embeddings' distribution of the target model more closely resembles that of the encoder when the DAC is employed. Furthermore, a smaller average cosine distance between the embeddings of the target model and the encoder is observed when using the DAC.

Malicious gradient generation using DAC: During each training round, both the adversary and other clients select a batch of training data, denoted as $\mathcal{B}_{train}$. Upon receiving the embeddings $h_{j,p}, j \in \mathcal{B}_{train}$ from passive clients, the adversary computes the malicious loss defined by:

$$\mathcal{L}_M = \frac{1}{|\mathcal{B}_{train}|} \sum_{j\in\mathcal{B}_{train}} CE\left(y_j^+, D(h_{j,p})\right). \tag{4}$$

The adversary then calculates the gradient $\frac{\partial L_M}{\partial h_{j,p}}$ and transmits it back to the passive clients to guide their bottom (target) models' learning processes. On the other hand, the adversary selects a batch from the auxiliary data $\mathcal{B}_{aux}$ and computes the embeddings $h_{i,p} = f_e(x_{i,p}), i \in \mathcal{B}_{aux}$. These embeddings are used to compute the DAC loss:

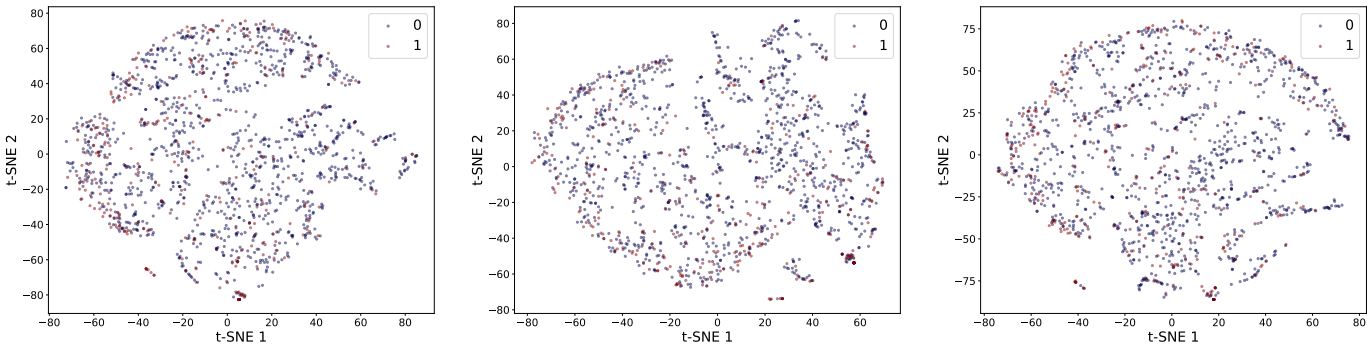
$$\begin{aligned}\mathcal{L}_D = &\frac{1}{|\mathcal{B}_{train}|} \sum_{j\in\mathcal{B}_{train}} CE(y_j^-, D(h_{j,p})) \\ &+ \frac{1}{|\mathcal{B}_{aux}|} \sum_{i\in\mathcal{B}_{aux}} CE(y_i^+, D(h_{i,p})).\end{aligned} \tag{5}$$

Minimizing $\mathcal{L}_D$ helps the DAC to better differentiate between real and fake embeddings associated with their labels, thereby refining its ability to guide the learning process of the target model while staying undetected. The detailed steps of malicious gradient generation are given in Algorithm 2.

Algorithm 2: Malicious gradient generation.

Data: Auxiliary dataset $D_{aux} = \{x_{i,a}, x_{i,p}\}_{i=1}^{M_{aux}}$ and training dataset $D_{train} = \{x_{j,a}, x_{j,p}\}_{j=1}^{M}$.
Initialization: Encoder $f_e(\cdot)$, DAC $D(\cdot)$, and target model $f_p(\cdot)$.
Adversary Procedure:
while *VFL training* **do**
  All clients agree a batch data $\mathcal{B}_{train}$ from D_{train};
  Receive and record passive clients' embeddings $h_{j,p}, j \in \mathcal{B}_{train}$;
  Compute the loss $\mathcal{L}_M = \frac{1}{|\mathcal{B}_{train}|} \sum_{j\in\mathcal{B}_{train}} CE(y_j^+, D(h_{j,p}))$;
  $\nabla_p \leftarrow$ Gradient$(\mathcal{L}_M, h_{j,p}, j \in \mathcal{B}_{train})$;
  Send ∇_p to the passive clients;
  The adversary select a batch of data $\mathcal{B}_{aux}$ from D_{aux};
  Compute $h_{i,p} = f_e(x_{i,p}), i \in \mathcal{B}_{aux}$;
  Compute the DAC loss $\mathcal{L}_D$ in (5);
  $\nabla_D \leftarrow$ Gradient$(\mathcal{L}_D, D(\cdot))$;
  $D(\cdot) \leftarrow$ Model_update$(D(\cdot), \nabla_D)$;
end
Passive Clients Procedure:
while *VFL training* **do**
  All clients agree a batch data $\mathcal{B}_{train}$ from D_{train};
  Compute embeddings $h_{j,p} = f_p(x_{j,p}), j \in \mathcal{B}_{train}$ and send embeddings to the active client;
  Receive gradient ∇_p;
  $f_p(\cdot) \leftarrow$ Model_update$(f_p(\cdot), \nabla_p)$;
end

An alternative synchronous training strategy: URVFL_sync. We also consider a variant of URVFL, named URVFL_sync, that replaces the pretraining with synchronous training of all models when generating malicious gradients. Specifically, during each training round, the adversary selects a batch of auxiliary data, $\mathcal{B}_{aux}$. The models $f_a(\cdot), f_d(\cdot)$ and $f_e(\cdot)$ are trained on $\mathcal{B}_{aux}$ to minimize the reconstruction loss $\mathcal{L}_R$ in (1). Upon receiving the embeddings $h_{j,p}, j \in \mathcal{B}_{train}$ from

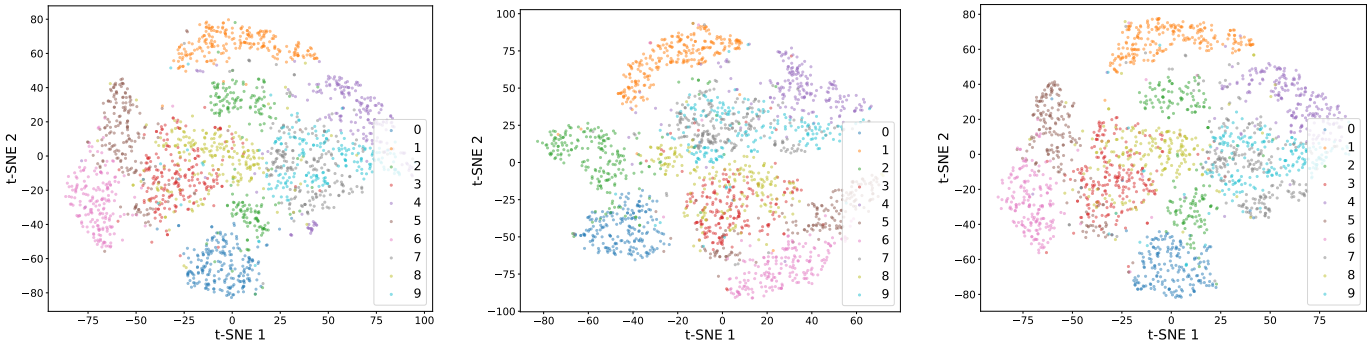


(a) Encoder's embedding distribution.

(b) Target model's embedding distribution using discriminator. Average cosine distance is 0.6121.

(c) Target model's embedding distribution using DAC. Average cosine distance is 0.1373.

Fig. 4: t-SNE visualization on Credit dataset with 2 classes.



(a) Encoder's embedding distribution.

(b) Target model's embedding distribution using discriminator. Average cosine distance is 0.0412.

(c) Target model's embedding distribution using DAC. Average cosine distance is 0.0283.

Fig. 5: t-SNE visualization on MNIST dataset with 10 classes.

the passive clients, the adversary calculates the gradients $\frac{\partial L_M}{\partial h_{j,p}}$ by minimizing the malicious loss $\mathcal{L}_M$ in (4), and transmits these gradients back to the passive clients. Following this, the adversary computes the embedding $h_{i,p}, i \in \mathcal{B}_{aux}$ using the updated encoder to calculate the DAC loss $\mathcal{L}_D$ in (5), and finally updates $D(\cdot)$. This process is repeated until all models reach convergence. The detailed steps of URVFL_sync, as outlined in Algorithm 3, are shown in Appendix A. Subsequent sections will demonstrate that URVFL_sync exhibits better reconstruction performance on certain datasets.

D. Data reconstruction

Having maliciously guided the training of the target model $f_p(\cdot)$ to mimic the encoder $f_e(\cdot)$, the adversary leverages the trained decoder $f_d(\cdot)$, alongside its bottom model $f_a(\cdot)$ to perform data reconstruction. For a target sample x (from training or test set) with adversary's partial features x_a and passive clients' features x_p, the adversary can reconstruct the private target features $\tilde{x}_t = f_d(f_a(x_a) \| f_p(x_p))$.

VI. Experiments

A. Experiment setup

Datasets and data processing: We conduct experiments across five representative datasets for VFL tasks, including two tabular datasets (i.e., Credit [48] and RT_IOT2022 [31]) and three image datasets (i.e., MNIST [4], CIFAR-10 [20], and Tiny imagenet [21]). Each dataset is partitioned into auxiliary, training, and test sets. For MNIST and CIFAR-10, we use the standard test sets provided. For the Credit dataset, we randomly allocate 30% of the samples to the test set, and for RT_IOT2022 and Tiny imagenet, 20% of the samples are selected as the test set. The auxiliary set, chosen randomly from the original training set, comprises 10% of the training set size in all main experiments, ensuring $|D_{aux}| : |D_{train}| = 1 : 10$ and $D_{aux} \cap D_{train} = \emptyset$. For image dataset, we normalize the data features to the range [-1,1]. For tabular dataset, most features are continuous or integer. We encode the categorical feature in Credit dataset using sklearn [28] and normalize all features with StandardScaler. Data features are evenly split between passive (target) and active clients, with each holding 50% of the features.

TABLE I: Results of data reconstruction attacks on tabular datasets. Attack methods followed by SG or GS represent implementations under SG or GS detections. URVFL and URVFL_sync achieve the same performance with and without detections. Some entries are missing due to 1) some attacks do not use encoders for data reconstruction, and hence do not have embedding MSE and cosine distance; and 2) the GS defense does not work with Credit dataset.

Method	Credit			RT_IOT2022		
	Recon MSE	Emb MSE	Emb Cos	Recon MSE	Emb MSE	Emb Cos
GRNA	1.1822 ± 0.0541	-	-	2.2542 ± 0.0325	-	-
GIA	0.9056 ± 0.0002	$\mathbf{0.2738} \pm 0.0008$	0.4681 ± 0.0215	1.8535 ± 0.0012	1.7276 ± 0.2356	0.4491 ± 0.0371
AGN	0.9656 ± 0.1196	-	-	2.2311 ± 0.1256	-	-
AGN-SG	1.4155 ± 0.1734	-	-	2.3967 ± 0.2628	-	-
AGN-GS	-	-	-	2.5670 ± 0.2909	-	-
PCAT	0.8612 ± 0.0984	0.5282 ± 0.0467	0.5486 ± 0.0197	2.2963 ± 0.4388	$\mathbf{1.5646} \pm 0.2488$	0.3742 ± 0.0195
SDAR	0.5327 ± 0.0374	0.5451 ± 0.1645	0.3103 ± 0.0931	1.6084 ± 0.1604	2.9362 ± 1.3356	0.2281 ± 0.1689
FSHA	0.5032 ± 0.0382	0.3906 ± 0.0622	0.2234 ± 0.0564	1.5773 ± 0.0507	2.2990 ± 0.6506	0.4859 ± 0.0984
FSHA-SG	0.7884 ± 0.1139	0.6056 ± 0.0704	0.5640 ± 0.0369	1.6895 ± 0.0419	2.7922 ± 1.0491	0.5909 ± 0.0745
FSHA-GS	-	-	-	1.7442 ± 0.0784	2.5768 ± 0.2169	0.5534 ± 0.1453
URVFL (SG/GS)	$\mathbf{0.4191} \pm 0.0541$	0.3078 ± 0.0778	$\mathbf{0.1658} \pm 0.0551$	1.3821 ± 0.0244	1.7894 ± 0.7435	$\mathbf{0.0549} \pm 0.0113$
URVFL_sync (SG/GS)	0.4722 ± 0.0228	0.3720 ± 0.0462	0.2277 ± 0.0441	$\mathbf{1.3277} \pm 0.0665$	2.3372 ± 0.5489	0.0938 ± 0.0340

Models: The adversary's encoder $f_e(\cdot)$ is configured identically to the passive clients' bottom model $f_p(\cdot)$ across all datasets. The structure of the adversary's bottom model $f_a(\cdot)$ mirrors that of $f_p(\cdot)$. For tabular data, we employ multi-layer perceptron (MLP) models for $f_e(\cdot), f_a(\cdot)$, $f_p(\cdot)$, the decoder $f_d(\cdot)$, and the DAC $D(\cdot)$. For image data, convolutional neural networks (CNNs) serve as the foundational architecture, with the CIFAR-10 and Tiny imagenet models featuring a residual block composed of three CNN layers and a skip connection. Detailed dataset information, model structures, and training parameters are provided in the Appendix B.

Metrics: We measure the reconstruction performance using average MSE between the reconstructed features and the original features, denoted as *Recon MSE*. Besides, to comprehensively measure the reconstruction performance on image datasets, we also leverage *PSNR* and *SSIM* as the metrics. To analyze the effectiveness of DAC compared to other embedding distribution transfer techniques (e.g., using discriminator or the top model), we measure both the average MSE and the cosine distance between the encoder's (or shadow model's) embeddings and the target model's embeddings, denoted as *Emb MSE* and *Emb Cos*, respectively. Before computing the cosine distance, all embeddings are normalized.

Environment: All experiments are conducted on a single machine equipped with four NVIDIA RTX 4090 GPUs. Each experiment is repeated five times, with the average results reported.

B. Baselines and detections

To evaluate the efficacy of our methods, we compare with three prominent data reconstruction strategies in VFL, as well as adaptations of attacks from SL. For HBC settings in VFL, we select GRNA and GIA, and for the malicious setting, we consider AGN. Additionally, we adapt three SL strategies for VFL: HBC attacks PCAT and SDAR, and the malicious attack FSHA.

GRNA: The adversary uses a generator model to create synthetic features. These synthetic features are then used to query the target model to update the generator, which minimizes the distance between the predictions made from synthetic features and those from real features.

GIA: This approach involves training a shadow model to emulate the target model's behavior. Then the adversary feeds random noise into the trained shadow model and optimizes noise to reduce the distance between the noise embeddings and real data embeddings.

AGN: AGN discards the top model and instead uses a generator to produce synthetic data from the adversary's features and aggregated embeddings. Concurrently, a discriminator works to differentiate between the generator's output and genuine data samples.

PCAT: PCAT uses the top model to guide a shadow model, which simulates the behavior of the target model. A decoder is trained on the shadow model to reconstruct the target features.

SDAR: SDAR builds on PCAT by adding two discriminators that evaluate both the embeddings from the shadow model and the reconstructed features.

FSHA: In FSHA, an embedding discriminator replaces the top model. Besides, a shadow model and a decoder are trained to reconstruct target features.

We also implement SOTA detection methods, SplitGuard (SG) and Gradient Scrutinizer (GS) on malicious attacks, to assess reconstruction performance under these defenses.

SplitGuard (SG): Analyzes gradients from fake and regular batches to compute a detection score after each round of fake batches. Following the original method, the target client calculates an average score from the last 10 records; if it falls below a threshold of 0.9, an attack is detected, and updates to the bottom model are halted.

Gradient Scrutinizer (GS): Evaluates the detection score of received gradients in each round to identify potential attacks. Due to instability in detection scores leading to frequent misjudgments, we adopt an average score strategy to improve reliability and set the threshold to 0.8 in our experiments. Similarly, when the average detection score falls below the detection threshold, an attack is detected and the bottom model update stop.

TABLE II: Results of data reconstruction on image datasets. Attack methods followed by SG or GS represent implementations under SG or GS detections. URVFL and URVFL_sync achieve the same performance with and without defenses. Some entries are missing since these attacks do not use encoders for data reconstruction, and hence, there is no embedding distance.

Method	MNIST			CIFAR-10			Tiny imagenet		
	Recon MSE	Emb MSE	Emb Cos	Recon MSE	Emb MSE	Emb Cos	Recon MSE	Emb MSE	Emb Cos
GRNA	0.5533 ± 0.0919	-	-	0.3287 ± 0.0061	-	-	0.3869 ± 0.0103	-	-
GIA	0.9125 ± 0.0056	0.0179 ± 0.0018	0.2772 ± 0.0431	0.2550 ± 0.0004	1.4011 ± 0.1849	0.3957 ± 0.0535	0.3234 ± 0.0009	1.8109 ± 0.1039	0.3936 ± 0.0068
AGN	0.4801 ± 0.0027	-	-	0.4413 ± 0.0385	-	-	0.3786 ± 0.0190	-	-
AGN-SG	0.5131 ± 0.0014	-	-	0.5154 ± 0.0776	-	-	1.2398 ± 0.0102	-	-
AGN-GS	0.7073 ± 0.0813	-	-	0.5901 ± 0.0620	-	-	1.9577 ± 0.2531	-	-
PCAT	0.2446 ± 0.1312	0.0282 ± 0.0016	0.3484 ± 0.0299	0.3843 ± 0.0123	2.4672 ± 0.3024	0.5544 ± 0.0023	0.3063 ± 0.0249	2.7853 ± 0.0521	0.5526 ± 0.0025
SDAR	0.0839 ± 0.0751	0.1638 ± 0.1850	0.3142 ± 0.1699	0.3067 ± 0.0618	0.5230 ± 0.6965	1.2196 ± 0.4831	0.1363 ± 0.0293	2.1344 ± 0.1174	0.4692 ± 0.0281
FSHA	0.0536 ± 0.0118	0.0168 ± 0.0100	0.0909 ± 0.0157	0.0317 ± 0.0065	0.4657 ± 0.1286	0.1610 ± 0.0285	0.1025 ± 0.0084	1.4897 ± 0.2210	0.3056 ± 0.0203
FSHA-SG	0.5158 ± 0.2511	0.0621 ± 0.0095	0.2956 ± 0.0691	0.1765 ± 0.0390	2.1634 ± 0.1310	0.4282 ± 0.0480	0.3110 ± 0.1202	2.8927 ± 0.3759	0.4904 ± 0.0519
FSHA-GS	0.2829 ± 0.1374	0.1446 ± 0.0260	0.3077 ± 0.0402	0.1312 ± 0.0150	1.5940 ± 0.0684	0.3133 ± 0.0230	0.4797 ± 0.2383	3.0416 ± 0.4313	0.5200 ± 0.0423
URVFL (SG/GS)	0.0132 ± 0.0027	0.0045 ± 0.0009	**0.0287** ± 0.0029	0.0302 ± 0.0011	0.5679 ± 0.0834	0.1303 ± 0.0131	**0.0699** ± 0.0055	1.4316 ± 0.6673	**0.2139** ± 0.0195
URVFL_sync (SG/GS)	**0.0127** ± 0.0011	**0.0040** ± 0.0006	0.0341 ± 0.0022	**0.0176** ± 0.0139	**0.2793** ± 0.1868	**0.0675** ± 0.0549	0.0704 ± 0.0011	**1.1441** ± 0.0058	0.2434 ± 0.0077

TABLE III: Results of data reconstruction measured by PSNR and SSIM on image datasets. Attack methods followed by SG or GS represent implementations under SG or GS detections. URVFL and URVFL_sync achieve the same performance with and without defenses. ↑ denotes that the larger the value, the better the performance.

Method	MNIST		CIFAR-10		Tiny imagenet	
	PSNR↑	SSIM ↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
GRNA	8.8317±0.0005	0.1467±0.0001	11.3939±0.0074	0.0019±0.0001	7.3986±0.5618	0.0045±0.0001
GIA	6.4411±0.0006	0.0029±0.0001	11.9625±0.0001	0.0018±0.0001	10.9342±0.0002	0.0010±0.0000
AGN	9.2787±0.2651	0.1400±0.0033	10.4527±0.1293	0.0150±0.0001	10.2442±0.0487	0.0150±0.0001
AGN-SG	8.9643±0.1067	0.0499±0.0015	9.0854±1.0272	0.0156±0.0001	3.4810±0.0000	0.0006±0.0256
AGN-GS	7.8738±0.6595	0.0746±0.0027	7.7755±0.9967	0.0142±0.0001	3.7093±0.4056	0.0034±0.0001
PCAT	14.0504±0.4956	0.4506±0.0177	11.9012±1.6192	0.0727±0.0053	11.3199±0.1573	0.0160±0.0001
SDAR	17.8807±0.6992	0.7093±0.0017	14.9388±2.3774	0.2525±0.0121	14.7862±0.9083	0.1318±0.0017
FSHA	22.7711±2.2248	0.8901±0.0006	19.7955±2.2895	0.5312±0.0298	15.9321±0.1248	0.2205±0.0003
FSHA-SG	10.4247±1.6129	0.1928±0.0176	13.2670±0.5318	0.1294±0.0006	12.6357±0.7394	0.0395±0.0002
FSHA-GS	11.1577±4.5247	0.2151±0.0137	15.1751±0.5590	0.3083±0.0053	12.4729±1.9752	0.0577±0.0001
URVFL(GS/SG)	25.1349±1.2354	**0.9385**±0.0002	21.0666±0.0165	0.5927±0.0002	**17.6107**±0.0992	0.3023±0.0001
URVFL_sync(GS/SG)	**25.2919**±0.1369	0.9324±0.0002	**26.2360**±0.0030	**0.8125**±0.0001	17.5381±0.0029	**0.3053**±0.0001

In our experimental results (Table I, II, and III), we append the GS or SG to the names of baseline methods to indicate their evaluation under detection mechanisms, SG or GS.

C. Data reconstruction results

Overview: As shown in Table I, II, and III, for all datasets, our proposed URVFL and URVFL_sync consistently outperform all other approaches in achieving the lowest reconstruction errors and embedding cosine distance. Our methods also achieve the best reconstruction performance measured by PSNR and SSIM compared with other baselines in image datasets. This underscores the effectiveness of our techniques and the advantage of embedding distribution transfer using DAC. For the Tiny imagenet, URVFL_sync achieves a reconstruction error of 0.0699, PSNR 17.5381, SSIM 0.3053, and cosine distance of 0.2139, significantly outperforming FSHA, which has a reconstruction error of 0.1025, PSNR 15.9321, SSIM 0.2205, and cosine distance of 0.3056.

Our methods demonstrate robustness against detection mechanisms, maintaining consistent performance and circumventing both detection methods malicious attacks, i.e., URVFL and URVFL_sync is not detected by SG and GS. In sharp contrast, the performance of AGN and FSHA become worse when faced with the detections, and even completely failed under some circumstances. Specifically, AGN's reconstruction error increases dramatically from 0.3786 to 1.9577, and FSHA's from 0.1025 to 0.4797 for the Tiny imagenet dataset. This demonstrates the stealthiness and adaptability of URVFL and URVFL_sync compared to other methods.

An observation from the results is that a smaller embedding MSE distance does not guarantee a more accurate reconstruction or a more similar embedding distribution, as evidenced by GIA in the Credit dataset and PCAT in the RT_IOT2022 dataset, which has the smallest embedding MSE distance but significantly higher cosine embedding distance and reconstruction error compared to our methods. While minimizing the embedding MSE distance helps reconstruction, it does not guarantee a more similar embedding distribution, as metrics like cosine distance better capture the similarity between distributions. Generally, the decoder is trained to reconstruct data from the embedding distribution. Therefore, even if our methods have a higher embedding MSE distance, as long as the target model generates a similar embedding distribution, the decoder can still work effectively.

Additionally, a smaller embedding cosine distance does

not necessarily correspond to a lower reconstruction error in URVFL and URVFL_sync. For example, for the MNIST and RT_IOT2022 dataset, while URVFL_sync achieves a lower reconstruction error than URVFL, it exhibits a slightly higher embedding cosine distance. This can be attributed to the synchronous training of all model components in URVFL_sync, which promotes dynamic learning and results in model diversity. Despite the higher average embedding distance, the decoder in URVFL_sync is more effective at reconstructing features, effectively compensating for the increased cosine distance.

We do not record AGN-GS and FSHA-GS in the Credit dataset. It is because GS doesn't work in the Credit dataset, which can also be observed in Figure 6j, caused by the limited data features (23 features) and imbalanced label distribution (77% label 0 and 23% label 1).

Visualization of the reconstruction: Table IV visualizes the reconstruction performance of the proposed URVFL, alongside the malicious attack baselines, AGN and FSHA, for the CIFAR-10 dataset. We also report the visualization result of Tiny imagenet in Table XI in Appendix C. The results clearly demonstrate that URVFL produces more accurate and distinct reconstructions compared to the baselines. While AGN and FSHA yield coarser results, URVFL maintains high-quality reconstruction even under SG detection. In contrast, both FSHA and AGN struggle to reconstruct informative pixels effectively when faced with detection mechanism SG.

SG and GS detection results: Figure 6 illustrates the detection scores for URVFL and URVFL_sync, which consistently exceed the detection thresholds, aligning indistinguishably with honest training scenarios. Although in the CIFAR-10 and Tiny imagenet dataset, URVFL's detection score is lower than the threshold at the beginning, its average score remains above the threshold, ensuring stealthiness against target clients. In contrast, malicious attack baselines, AGN and FSHA, consistently maintain scores below the detection thresholds of SG and GS in the MNIST and CIFAR-10 datasets, rendering them detectable.

In the RT_IOT2022 and Credit datasets under defense SG, AGN sporadically surpasses the threshold, yet its average detection score falls below the threshold, leading passive clients to detect an attack promptly. FSHA's SG scores remain below the threshold throughout, making it easily detectable. Under GS, FSHA exceeds the threshold at the beginning but is ultimately detected in the RT_IOT2022 dataset. In the Credit dataset, as previously noted, GS is ineffective due to imbalanced labels and limited features.

Embedding distances across attack iterations: Figure 7 visualizes the changes in embedding distance during the attack for our methods compared to FSHA on the CIFAR-10 dataset. The figure clearly demonstrates that our methods steadily reduce the embedding distance throughout the attack, resulting in lower overall embedding distances and effectively aligning the embedding distribution with the target model. In contrast, FSHA exhibits less stability, despite also showing a decreasing trend in embedding distance.

Furthermore, we observe that the SG score for a single iteration of our methods falls below the threshold within the first 10 iterations in Fig. 6, whereas FSHA remains below the threshold throughout the entire training process, leading to the detection of the attack. This difference may be attributed to the more drastic changes in embedding distance during our training process. As illustrated in Figure 7, our methods cause rapid changes in embedding distance initially, which then slow down, while FSHA maintains a more consistent fluctuations in embedding distance throughout the attack.

D. Ablation Study

This section explores the robustness of URVFL and URVFL_sync (abbreviated as sync) by evaluating their performance under various parameter settings. These settings include changes in the adversary's feature size, use of heterogeneous encoder models, variations in auxiliary dataset size, and differences in the number of target clients. Except for the variables specifically altered for this study, the model setups and implementation environment remain consistent with those described in the main experiments.

1) Effect of adversary's feature size: To explore how the size of the feature set held by the adversary affects the reconstruction performance of URVFL and sync, we conduct experiments targeting the reconstruction of 30% of the features while varying the proportion of features held by the adversary from 0% to 70%. At 0% feature possession, the scenario reduces to split learning where no features or bottom model are held by the adversary. These experiments are performed on the Credit and RT_IOT2022 (abbreviated as IOT) datasets, with the results summarized in Figure 8.

Consistent with our expectations, increasing the proportion of features held by the adversary leads to less reconstruction error. As illustrated in Figure 8, the reconstruction error in the Credit dataset using URVFL decreases from 0.8315 to 0.5189 as the feature possession increases from 0% to 70%. This trend is observed for both datasets and methods.

2) Effect of encoder's complexity: In our main experiments, the encoder mirrors the target model's structure. However, when the adversary lacks precise knowledge about the target model's configuration, discrepancies between the encoder and the target model's structures can affect the transfer of embedding distributions and, consequently, the overall reconstruction performance. To evaluate the impact of varying encoder complexities, we conduct experiments on the CIFAR-10 dataset, where the target model comprises **3 residual blocks**. We test the following encoder configurations to assess their effect on performance. **MLP Model**: Comprises 3 linear layers, with input and output dimensions matching those of the target model; **1-Res Model**: Utilizes a single residual block. **5-Res Model**: Extends to 5 residual blocks, increasing complexity. We test the reconstruction error and measure both the embeddings MSE and cosine distances for each encoder configuration.

Table V clearly illustrates that the adversary achieves the lowest reconstruction error and the smallest embedding cosine distance when using an encoder the same as the target model, as observed for both URVFL and sync. This optimal alignment

TABLE IV: Visualization of reconstructed features on the CIFAR-10 dataset with and without the detection SG. Except for the original image, the right side of each image is the reconstructed image. The last column is the average reconstruction error.

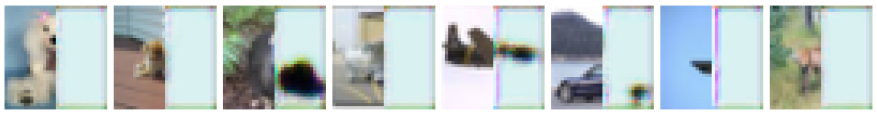
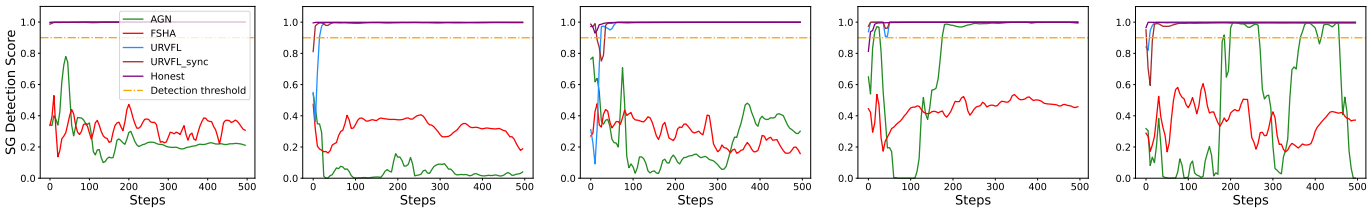
	Original		MSE
Without defense	AGN		0.3868
	FSHA		0.0479
	DMAVFL		0.0307
With defense	AGN		0.6536
	FSHA		0.5136
	DMAVFL		0.0307

TABLE V: Reconstruction error and average embeddings distance using different encoder's model.

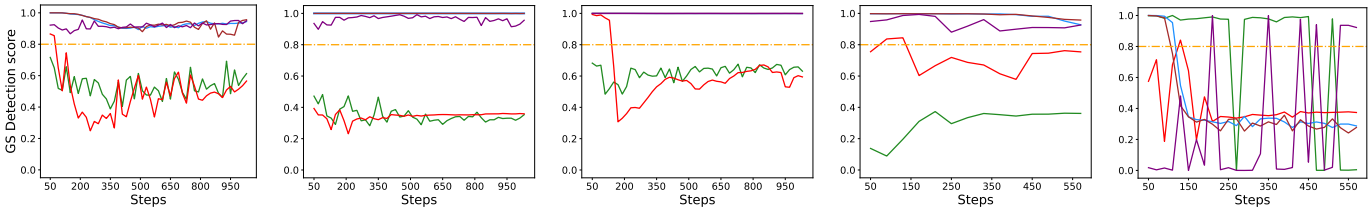
Encoder	Metric	URVFL	sync
MLP	Recon MSE	0.2545	0.2362
	Emb MSE dis	7.7767	3.9726
	Emb Cos dis	0.6337	0.6081
1-Res	Recon MSE	0.0561	0.0434
	Emb MSE dis	0.3175	0.3059
	Emb Cos dis	0.1632	0.1450
3-Res (Same)	Recon MSE	**0.0302**	**0.0176**
	Emb MSE dis	0.5679	0.2793
	Emb Cos dis	0.1303	0.0675
5-Res	Recon MSE	0.0313	0.0366
	Emb MSE dis	0.8771	0.7071
	Emb Cos dis	0.1081	0.1112

underscores the importance of model congruence in enhancing the effectiveness of embedding distribution transfer and data reconstruction. Conversely, when the encoder structure deviates from that of the target model, specifically when switching from a CNN to an MLP, the reconstruction performance significantly deteriorates, with reconstruction error increasing by an order of magnitude (from 0.0302 to 0.2545). This dramatic drop highlights the challenges of using less compatible models for data reconstruction attack. Moreover, the results indicate that a complex encoder, featuring a higher number of residual blocks, outperforms a simpler encoder configuration. The complex encoder not only enhances the decoder's ability to reconstruct the data accurately but also learns a more informative embedding. This suggests that increasing the encoder's complexity, when the target model is unknown, can lead to better performance in both embedding distribution transfer and data reconstruction.

3) Effect of auxiliary dataset: In the main experiments, we set the size of the adversary's auxiliary dataset, $|D_{aux}|$, relative to the training set size, $|D_{train}|$, at a ratio of $1:10$. To explore the impact of varying auxiliary dataset sizes, we adjust the ratio of $|D_{aux}|:|D_{train}|$ from $0:10$ to $1:1$ in the MNIST and CIFAR10 dataset. Specifically, at a ratio of 0:10, the auxiliary dataset of



(a) SG scores in MNIST. (b) SG scores in CIFAR-10. (c) SG scores in Tiny. (d) SG scores in IOT. (e) SG scores in Credit.



(f) GS scores in MNIST. (g) GS scores in CIFAR-10. (h) GS scores in Tiny. (i) GS scores in IOT. (j) GS scores in Credit.

Fig. 6: Detection scores of SG and GS on malicious attacks and the honest training across four datasets. We abbreviate Tiny imagenet dataset to Tiny and RT_IOT2022 dataset to IOT.

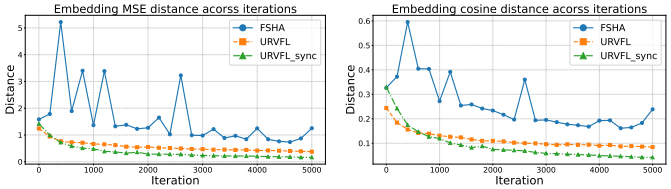


Fig. 7: Embedding MSE and cosine distances across attack iterations using FSHA, URVFL, and URVFL_sync.

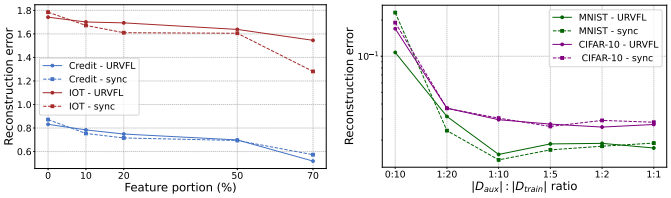


Fig. 8: Reconstruction error under various adversary's feature portions.

Fig. 9: Reconstruction error for different $|D_{aux}|$ relative to $|D_{train}|$.

MNIST and CIFAR-10 comprise the FashionMNIST [42] and the downsampled Tiny imagenet, respectively, introducing a distinct distribution from the training set. This setup aims to test the adaptability of the reconstruction process to changes in the auxiliary dataset's distribution.

From Figure 9 we can see that significant data leakage can occur, even in the absence of auxiliary data with the same distribution of the training set (i.e., with out-of-distribution auxiliary data). This observation is highlighted by a reconstruction error of 0.1076∼0.2311, when the adversary uses FashionMNIST data to reconstruct images from MNIST, and 0.1696∼0.1906 when the adversary uses Tiny imagenet data to reconstruct images from CIFAR-10. We also observe that for both URVFL and sync, the reconstruction error decreases quickly as $|D_{aux}|$ becomes relatively larger. However, after the ratio of $|D_{aux}| : |D_{train}|$ reaches 1:10, increasing auxiliary dataset size results in minor improvements in reconstruction error. This suggests that there is a threshold beyond which additional auxiliary data does not substantially improve the model's ability to reconstruct target features, pointing to a diminishing return on the utility of expanding the auxiliary set.

4) Effect of the number of target clients: Our proposed methods, URVFL and sync, are designed to handle scenarios involving multiple target clients. To assess the impact of varying the number of target clients on data reconstruction performance, we conduct experiments on the IOT dataset with the number of target clients ranging from 1 to 5. Consistent with the setup in the main experiments, 50% of the features are equally distributed among the target clients.

TABLE VI: Reconstruction errors for different numbers of target clients.

# of target clients	1	2	3	4	5
URVFL	**1.3821**	1.5801	1.5986	1.6718	1.9820
sync	**1.3277**	1.6447	1.7597	1.8660	2.2635

The results presented in Table VI indicate that an increase in the number of target clients, each sharing a smaller portion of the features, leads to diminished reconstruction performance. Specifically, the reconstruction error increases from 1.3821 with a single target client to 1.9820 when there are five target clients. This degradation in performance can be attributed to the distribution of features among more clients, each employing a separate bottom model to learn their respective features. This setup tends to cause the loss of crucial feature correlation information during the learning process. Moreover, as the encoder is trained on the complete set of features and is hence able to capture more feature correlations, the increased disparity between the encoder and the target models' understanding of the feature correlation further exacerbates the challenge of accurate data reconstruction.

VII. Evaluations on defenses

While URVFL is designed and demonstrated to penetrate detection-based methods of SplitGuard and Gradient Scrutinizer, we also evaluate its effectiveness against general defensive methods for privacy protection in VFL and SL. *Note that unlike SplitGuard and Gradient Scrutinizer, these methods essentially introduce perturbations to the communicated embeddings or gradients, severely degrading the performance of honest training.* Specifically, we consider three such defenses: Nopeek [37], obfuscation with random noise [15], and differential privacy (DP) [1], [5], [40].

Nopeek Defense. Nopeek is a method for minimizing distance correlation, widely implemented in both VFL and SL to prevent privacy leakage. It operates by minimizing the correlation between embeddings and the original features to reduce information leakage. During VFL training, passive clients incorporate the Nopeek loss into their training loss, defined as:

$$\mathcal{L}_N = \frac{1}{|\mathcal{B}_{train}|} \sum_{j \in \mathcal{B}_{train}} (\alpha \cdot DCOR(x_{j,n}, f_n(x_{j,n})) + (1-\alpha) \cdot TASK(y_j, f_n(x_{j,n})), \tag{6}$$

where *DCOR* represents the distance correlation metric, *TASK* denotes the VFL training loss, and α is a hyper-parameter balancing training and defense.

We challenge Nopeek by deploying URVFL and sync against it with varying α values from 0.1 to 1.0 on the MNIST dataset. The results in Table VII show that when $\alpha < 0.9$, Nopeek fails to thwart our attacks. As α approaches 0.9 and beyond, Nopeek begins to impede the attack effectiveness. Notably, even at $\alpha = 0.95$, URVFL still achieves a small reconstruction error of 0.0280. Setting $\alpha = 1.0$ effectively defends the attack. However, this completely replaces the original VFL task with minimizing the *DCOR* loss, which results in an accuracy of merely 69.03%.

TABLE VII: Reconstruction error against Nopeek.

α	0	0.2	0.4	0.6	0.8	0.9	0.95	1.0
URVFL	0.0132	0.0151	0.0122	0.0138	0.0116	0.0171	0.0280	0.4759
sync	0.0127	0.0143	0.0122	0.0113	0.0135	0.0156	0.0403	0.4462
Honest Acc(%)	99.11	97.82	98.28	98.04	97.90	95.03	94.72	69.03

Obfuscation with random noise. Perturbation methods, such as adding noise to transmitted embeddings, are commonly employed in ML to safeguard privacy. Following the approach outlined in [15], we introduce noise with standard deviations σ ranging from 0.2 to 0.8 into the embeddings sent by target clients. This defense is tested against URVFL and sync on the MNIST dataset, and the results are reported in Table VIII. The results suggest that higher noise levels more effectively impede the attacks. This defense is more effective for sync, as its DAC learns a dynamic distribution of the encoding embedding during training, and is more susceptible to fluctuations caused by noise. Nonetheless, our attacks manage to reconstruct informative features even under significant noise, for instance, URVFL achieving a reconstruction error of 0.0472 at $\sigma = 0.7$. It is also important to note that adding larger noise adversely affects the honest training accuracy.

TABLE VIII: Reconstruction error against obfuscation.

σ	0	0.1	0.3	0.5	0.7
URVFL	0.0132	0.0214	0.0252	0.0351	0.0472
sync	0.0127	0.0161	0.0301	0.0497	0.0752
Honest Acc(%)	99.11	98.34	97.65	94.84	94.00

Differential privacy (DP). DP provides a mathematical framework for quantifying privacy guarantees for ML models. In our attacks, as the gradients returned from the active client may be maliciously generated, we propose to apply ε-DP mechanism, which involves adding Laplacian noise to received gradients, to disrupt the malicious training. We test varying levels of privacy by adjusting the parameter ε from 0.1 to 10, where a smaller ε implies stronger privacy due to higher noise levels. The average reconstruction errors of URVFL and sync on MNIST are shown in Table IX, indicating that DP is most effective at an ε of 0.1, successfully impeding our proposed attacks. However, as ε increases, its defensive effectiveness decreases. At $\varepsilon = 10$, the noise assists the attack, potentially by making the model updates more robust, thereby facilitating the embedding distribution transfer. It's important to note that while DP helps to protect privacy, it also hurts the accuracy of honest training.

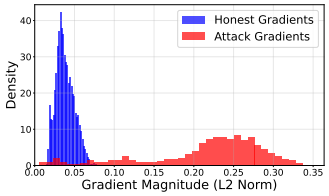
TABLE IX: Reconstruction error against DP. $\varepsilon = +\infty$ corresponds to the scenario without DP.

ε	0.1	0.5	1	2	5	10	$+\infty$
URVFL	0.3134	0.0773	0.0443	0.0210	0.0210	0.0122	0.0132
sync	0.3178	0.1253	0.0507	0.0468	0.0172	0.0110	0.0127
Honest Acc(%)	87.75	90.00	94.97	95.90	96.72	97.36	99.11

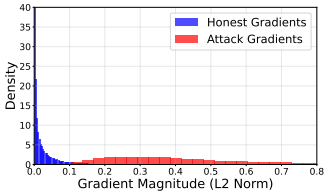
VIII. Potential detection of our attacks

In this section, we explore potential detection mechanisms tailored to identify our proposed attacks. Our attack integrates label information into the gradients, rendering simple observations of gradient differences across various labels ineffective for detection. Specifically, our approach not only classifies the corresponding labels of embeddings but also distinguishes between true and fake embeddings generated by either the local encoder or the target model. Consequently, we hypothesize that the increased number of labels and the incorporation of a GAN-like structure will influence the distribution of gradient norms. To investigate this, we analyze the distribution of gradients' L2 norms across both tabular and image datasets.

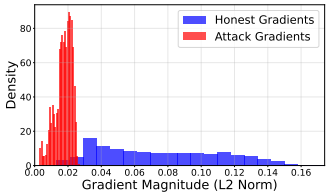
Fig.10 illustrates a clear distinction in the distribution of embedding gradients' L2 norms between honest training and our URVFL attack. In tabular datasets, the gradient norms under attack are sparsely distributed with a higher average value, whereas, in honest training, they concentrate around lower values. Conversely, for image datasets, where the pattern reverses, honest training results in sparsely distributed gradients with higher average norms, while the URVFL attack leads to concentrated gradients with lower average norms. This



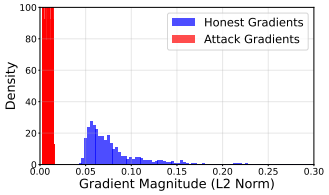
(a) Embedding gradients norm in Credit dataset.



(b) Embedding gradients norm in IOT dataset.



(c) Embedding gradients norm in CIFAR-10 dataset.



(d) Embedding gradients norm in Tiny imagenet dataset.

Fig. 10: Distribution of Embedding Gradients' L2 Norms in Honest Training versus Our URVFL Attack.

discrepancy is likely attributable to inherent differences in data characteristics and model architectures between image and tabular datasets.

Despite the varying gradient norm distributions between tabular and image datasets, a consistent and significant difference remains evident between honest training and the URVFL attack across both data types. Leveraging this insight, a passive client can enhance detection capabilities by adopting a two-step strategy. Initially, the passive client collects publicly available comprehensive training data and conducts local training for several hundred iterations to establish a baseline distribution of embedding gradients' L2 norms under honest training conditions. This baseline serves as a reference point for normal gradient behavior.

During subsequent training sessions, the passive client continuously records the distribution of embedding gradients. By comparing the observed gradient distribution against the pre-established honest training distribution, the client can identify significant deviations indicative of an attack. If the gradient norms exhibit patterns that deviate substantially from the baseline, such as increased sparsity and higher average norms in tabular datasets or the opposite pattern in image datasets, the passive client can infer the presence of an attack. Upon detecting such discrepancies, the passive client can stop updating the local bottom model, thereby preventing further compromise and safeguarding the integrity and privacy of the training process.

IX. Conclusion

We introduce URVFL, a novel undetectable malicious data reconstruction attack for vertical federated learning (VFL). URVFL innovatively incorporates training of a discriminator with auxiliary classifier (DAC) to integrate label information for malicious gradient generation, which not only makes the malicious training indistinguishable from honest training, but also substantially improves data reconstruction performance. Through visualization and quantification, we demonstrated the effectiveness of DAC compared to traditional discriminators, showcasing its superior capability in embedding transfer. Through extensive experiments, we empirically demonstrated that URVFL successfully evades SOTA detection methods and outperforms all existing data reconstruction attacks. While traditional defense methods based on embedding/model perturbations are shown to effectively defend URVFL, it is at the cost of serious degradation on the honest VFL task's performance. This calls for future research to develop effective defenses against malicious data reconstruction, with minimal impact on the honest task.

Acknowledgment

The work of Songze Li is in part supported by the National Nature Science Foundation of China (NSFC) Grant 62106057, and the Fundamental Research Funds for the Central Universities (Grant No. 2242024k30059). The work of Gaoning Pan is in part supported by the "Pioneer" and "Leading Goose" R&D Program of Zhejiang, China (Grant No. 2024C03288).

References

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 308–318.

[2] S. Abuadbba, K. Kim, M. Kim, C. Thapa, S. A. Camtepe, Y. Gao, H. Kim, and S. Nepal, "Can we use split learning on 1d cnn models for privacy preserving training?" in *ACM Asia Conference on Computer and Communications Security*, 2020, pp. 305–318.

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

[4] L. Deng, "The MNIST database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[5] C. Dwork, "Differential privacy," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2006, pp. 1–12.

[6] E. Erdogan, A. Küpçü, and A. E. Cicek, "Splitguard: Detecting and mitigating training-hijacking attacks in split learning," in *21st Workshop on Privacy in the Electronic Society*, 2022, pp. 125–137.

[7] E. Erdoğan, A. Küpçü, and A. E. Çiçek, "Unsplit: Data-oblivious model inversion, model stealing, and label inference attacks against split learning," in *Workshop on Privacy in the Electronic Society*, 2022, pp. 115–124.

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

[9] F. Fu, H. Xue, Y. Cheng, Y. Tao, and B. Cui, "Blindfl: Vertical federated machine learning without peeking into your data," in *International Conference on Management of Data*, 2022, pp. 1316–1330.

[10] J. Fu, X. Ma, B. B. Zhu, P. Hu, R. Zhao, Y. Jia, P. Xu, H. Jin, and D. Zhang, "Focusing on pinocchio's nose: A gradients scrutinizer to thwart split-learning hijacking attacks using intrinsic attributes," in *The Network and Distributed System Security Symposium*, 2023.

[11] B. Fuglede and F. Topsoe, "Jensen-shannon divergence and hilbert space embedding," in *International Symposium on Information Theory*. IEEE, 2004, p. 31.

[12] X. Gao and L. Zhang, "Pcat: Functionality and data stealing from split learning by pseudo-client attack," in *USENIX Security Symposium*, 2023, pp. 5271–5288.

[13] M. Gong, Y. Zhang, Y. Gao, A. Qin, Y. Wu, S. Wang, and Y. Zhang, "A multi-modal vertical federated learning framework based on homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, 2023.

[14] Z. He, T. Zhang, and R. B. Lee, "Model inversion attacks against collaborative inference," in *Annual Computer Security Applications Conference*, 2019, pp. 148–162.

[15] ——, "Attacking and protecting data privacy in edge–cloud collaborative inference systems," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9706–9716, 2020.

[16] L. Hou, Q. Cao, H. Shen, S. Pan, X. Li, and X. Cheng, "Conditional gans with auxiliary discriminative classifier," in *International Conference on Machine Learning*. PMLR, 2022, pp. 8888–8902.

[17] Y. Huang, W. Wang, X. Zhao, Y. Wang, X. Feng, H. He, and M. Yao, "Efmvfl: an efficient and flexible multi-party vertical federated learning without a third party," *ACM Transactions on Knowledge Discovery from Data*, vol. 18, no. 3, pp. 1–20, 2023.

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

[19] J. Kim, S. Shin, Y. Yu, J. Lee, and K. Lee, "Multiple classification with split learning," in *International Conference on Smart Media and Applications*, 2020, pp. 358–363.

[20] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[21] Y. Le and X. Yang, "Tiny imagenet visual recognition challenge," *CS 231N*, vol. 7, no. 7, p. 3, 2015.

[22] S. Li, D. Yao, and J. Liu, "Fedvs: Straggler-resilient and privacy-preserving vertical federated learning for split models," in *International Conference on Machine Learning*. PMLR, 2023, pp. 20 296–20 311.

[23] Y. Liu, Y. Kang, T. Zou, Y. Pu, Y. He, X. Ye, Y. Ouyang, Y.-Q. Zhang, and Q. Yang, "Vertical federated learning," *arXiv preprint arXiv:2211.12814*, 2022.

[24] ——, "Vertical federated learning: Concepts, advances, and challenges," *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[25] X. Luo, Y. Wu, X. Xiao, and B. C. Ooi, "Feature inference attack on model predictions in vertical federated learning," in *International Conference on Data Engineering*. IEEE, 2021, pp. 181–192.

[26] F. Mireshghallah, M. Taram, A. Jalali, A. T. T. Elthakeb, D. Tullsen, and H. Esmaeilzadeh, "Not all features are equal: Discovering essential features for preserving prediction privacy," in *The Web Conference*, 2021, pp. 669–680.

[27] D. Pasquini, G. Ateniese, and M. Bernaschi, "Unleashing the tiger: Inference attacks on split learning," in *ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 2113–2129.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[29] M. G. Poirot, P. Vepakomma, K. Chang, J. Kalpathy-Cramer, R. Gupta, and R. Raskar, "Split learning for collaborative deep learning in healthcare," *arXiv preprint arXiv:1912.12115*, 2019.

[30] P. Qiu, X. Zhang, S. Ji, Y. Pu, and T. Wang, "All you need is hashing: Defending against data reconstruction attack in vertical federated learning," *arXiv preprint arXiv:2212.00325*, 2022.

[31] B. S. and R. Nagapadma, "RT-IoT2022 ," UCI Machine Learning Repository, 2024, DOI: https://doi.org/10.24432/C5P338.

[32] M. J. Sheller, B. Edwards, G. A. Reina, J. Martin, S. Pati, A. Kotrotsou, M. Milchenko, W. Xu, D. Marcus, R. R. Colen *et al.*, "Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data," *Scientific reports*, vol. 10, no. 1, p. 12598, 2020.

[33] J. Sun, Y. Yao, W. Gao, J. Xie, and C. Wang, "Defending against reconstruction attack in vertical federated learning," *arXiv preprint arXiv:2107.09898*, 2021.

[34] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE transactions on neural networks and learning systems*, vol. 34, no. 12, pp. 9587–9603, 2022.

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

[36] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," *arXiv preprint arXiv:1812.00564*, 2018.

[37] P. Vepakomma, A. Singh, O. Gupta, and R. Raskar, "Nopeek: Information leakage reduction to share activations in distributed deep learning," in *International Conference on Data Mining Workshops*. IEEE, 2020, pp. 933–942.

[38] M. N. Vu, J. Tre'R, R. Alharbi, and M. T. Thai, "Active data reconstruction attacks in vertical federated learning," in *IEEE International Conference on Big Data*, 2023, pp. 1374–1379.

[39] C. Wang, J. Liang, M. Huang, B. Bai, K. Bai, and H. Li, "Hybrid differentially private federated learning on vertically partitioned data," *arXiv preprint arXiv:2009.02763*, 2020.

[40] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.

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

[42] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[43] W. Xu, H. Fan, K. Li, and K. Yang, "Efficient batch homomorphic encryption for vertically federated xgboost," *arXiv preprint arXiv:2112.04261*, 2021.

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

[45] L. Yang, B. Tan, V. W. Zheng, K. Chen, and Q. Yang, "Federated recommendation systems," *Federated Learning: Privacy and Incentive*, pp. 225–239, 2020.

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

[47] R. Yang, J. Ma, J. Zhang, S. Kumari, S. Kumar, and J. J. Rodrigues, "Practical feature inference attack in vertical federated learning during prediction in artificial internet of things," *IEEE Internet of Things Journal*, 2023.

[48] I.-C. Yeh, "Default of Credit Card Clients," UCI Machine Learning Repository, 2016, DOI: https://doi.org/10.24432/C55S3H.

[49] Y. Zhao, Q. Liu, P. Liu, X. Liu, and K. He, "Medical federated model with mixture of personalized and shared components," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[50] X. Zhu, X. Luo, Y. Wu, Y. Jiang, X. Xiao, and B. C. Ooi, "Passive inference attacks on split learning via adversarial regularization," *arXiv preprint arXiv:2310.10483*, 2023.

APPENDIX A
URVFL_SYNC ALGORITHM

We detail the steps of URVFL_sync in Algorithm 3.

Algorithm 3: URVFL_sync.

Data: Auxiliary dataset $D_{aux} = \{x_{i,a}, x_{i,p}\}_{i=1}^{M_{aux}}$ and training dataset $D_{train} = \{x_{j,a}, x_{j,p}\}_{j=1}^{M}$.
Initialization: Encoder $f_e(\cdot)$, DAC $D(\cdot)$, adversary's bottom model $f_a(\cdot)$, and target model $f_p(\cdot)$.
Adversary Procedure:
while *VFL training* **do**
The adversary select a batch $\mathcal{B}_{aux}$ from D_{aux};
Compute $h_{i,a} = f_a(x_{i,a}), i \in \mathcal{B}_{aux}$, and $h_{i,p} = f_e(x_{i,p}), i \in \mathcal{B}_{aux}$;
Compute $\tilde{x}_{i,t} = f_d([h_{i,a} \| h_{i,p}]), i \in \mathcal{B}_{aux}$;
$\mathcal{L}_R \leftarrow \frac{1}{|\mathcal{B}_{aux}|} \sum_{i \in \mathcal{B}_{aux}} MSE(\tilde{x}_{i,t}, x_{i,t})$;
$\nabla_e \leftarrow$ Gradient$(\mathcal{L}_R, f_e(\cdot))$;
$\nabla_a \leftarrow$ Gradient$(\mathcal{L}_R, f_a(\cdot))$;
$\nabla_d \leftarrow$ Gradient$(\mathcal{L}_R, f_d(\cdot))$;
$f_e(\cdot) \leftarrow$ Model_update$(f_e(\cdot), \nabla_e)$;
$f_a(\cdot) \leftarrow$ Model_update$(f_a(\cdot), \nabla_a)$;
$f_d(\cdot) \leftarrow$ Model_update$(f_d(\cdot), \nabla_d)$;
Compute $h_{i,p} = f_e(x_{i,p}), i \in \mathcal{B}_{aux}$;
All clients agree a batch data $\mathcal{B}_{train}$ from D_{train};
Receive and record passive clients' embeddings $h_{j,p}, j \in \mathcal{B}_{train}$;
Compute the loss $\mathcal{L}_M = \frac{1}{|\mathcal{B}_{train}|} \sum_{j \in \mathcal{B}_{train}} CE(y_j^+, D(h_{j,p}))$;
$\nabla_p \leftarrow$Gradient$(\mathcal{L}_M, h_{j,p}, j \in \mathcal{B}_{train})$;
Send ∇_p to the passive clients;
Compute the DAC loss $\mathcal{L}_D$ in (5);
$\nabla_D \leftarrow$ Gradient$(\mathcal{L}_D, D(\cdot))$;
$D(\cdot) \leftarrow$ Model_update$(D(\cdot), \nabla_D)$;
end
Passive Clients Procedure:
while *VFL training* **do**
All clients agree a batch data $\mathcal{B}_{train}$ from D_{train};
Compute embeddings $h_{j,p} = f_p(x_{j,p}), j \in \mathcal{B}_{train}$ and send embeddings to the active client;
Receive gradient ∇_p;
$f_p(\cdot) \leftarrow$ Model_update$(f_p(\cdot), \nabla_p)$;
end

APPENDIX B
DATASETS AND MODEL DESCRIPTIONS

We illustrate all datasets and models information used in the experiments in the following Table X, where MLP is composed of a linear and an activation, e.g., ReLU layer, CNN layer is composed a convolution process and a linear layer, and Residual block is constructed by 3 CNN layers and a skip connection.

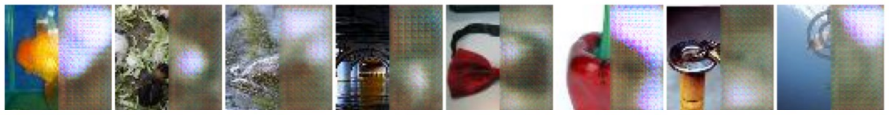
APPENDIX C
VISUALIZATION OF RECONSTRUCTION IN TINY IMAGENET

In addition to visualizing the reconstruction performance on CIFAR-10, we compare the reconstruction performance of AGN, FSHA, and URVFL on the Tiny ImageNet dataset in Table XI.

TABLE X: Dataset characteristics and models structure

Dataset	Credit	RT_IOT2022	MNIST	CIFAR-10	Tiny imagenet
Feature sizes of the adversary and the passive clients d_a, d_p	11, 12	41, 42	392, 392	512, 512	2048, 2048
The size of training dataset	19,091	89,920	54,546	45,455	72,728
The size of the auxiliary dataset	1,909	8,573	5,454	4,545	7,272
The size of the test dataset	9,000	24,624	10,000	10,000	20,000
$f_e(\cdot), f_a(\cdot)$, and $f_p(\cdot)$ structure	MLP layer×2	MLP layer×2	CNN layer×2	Residual block×3	Residual block×3
$f_d(\cdot)$ structure	MLP layer×3	MLP layer×3	CNN layer×3	CNN layer×3	CNN layer×4
$D(\cdot)$ structure	MLP layer×3	MLP layer×3	CNN layer×2	CNN layer ×3	CNN layer×3
Learning rate	1e-4	1e-4	1e-4	1e-3	1e-4
URVFL pretraing epochs	30	40	10	20	30

TABLE XI: Visualization of reconstructed features on the Tiny imagenet dataset with and without the detection SG. Except for the original image, the right side of each image is the reconstructed image. The last column is the average reconstruction error.

	Original		Recon MSE
Without detection SplitGuard	AGN		0.3896
	FSHA		0.1131
	URVFL		0.0660
With detection SplitGuard	AGN		1.0493
	FSHA		0.2501
	URVFL		0.0660