Uncovering Risks of Data-Free Feature Vector Inversion Attacks Against Vector Databases

Shengyang Qin, Xinyu Lei, Nankun Mu, Hongyu Huang, Tian Xie, Xiao Zhang

***Abstract*—The vector database stores data as high-dimensional feature vectors. Some recently proposed attack techniques enable an adversary to launch feature vector inversion (FVI) attacks against vector databases. In FVI attacks, an adversary trains an FVI attack network to reconstruct the original private data from their feature vectors based on the assumption that an auxiliary dataset is available to the adversary. However, such a data-available assumption is too strong, making such FVI attacks unrealistic in many real-world scenarios. In this paper, we make the first systematic study on FVI attacks against vector databases in the data-free setting. To tackle the issue of no training data, we develop an output-to-input data generation technique that helps to generate synthetic fake samples for the FVI attack network training. In addition, to ensure the high quality of generated fake samples, we develop the accelerable complete bipartite graph (CBG) search strategy and the downstream-classifier-aided generator training strategy. Furthermore, we empirically identify multiple factors that influence the attack performance. Intriguingly, as the key insight of this work, we find that the proposed FVI attack technique in the data-free setting can be directly employed to boost the attack performance of FVI attacks in the auxiliary-dataset-available setting. Finally, we propose and study defenses against the proposed attacks.**

***Index Terms*—Vector database, feature vector inversion attack, data-free.**

I. Introduction

THE amount of unstructured data (*e.g.*, images and graphs) is increasing rapidly nowadays. To manage unstructured data, a common practice is to use machine learning (ML) techniques to extract a feature vector (*i.e.*, a list of real-valued numbers) from an unstructured data. Then, the unstructured data object can be represented by its feature vector that is mathematically and computationally convenient to be used for various downstream ML tasks. Feature vectors can also be called embeddings in natural language processing (NLP) ML tasks or templates in biometrics ML tasks. In recent years, deep learning (DL)-based algorithms have shown their exceptional ability in feature extraction since the extracted feature vectors usually achieve state-of-the-art performance when used in downstream ML tasks. To facilitate feature extraction, many deep neural network (DNN)-based feature (or embedding) extractors have been developed, pre-trained, and published online. For example, the pre-trained ResNet-50 [20] and MobileNet [24] are published to extract image feature vectors for different ML tasks. After being extracted by DNN-based feature extractors, feature vectors can be stored in many different forms of containers such as a database-like file (*e.g.*, Microsoft Excel), a relational database, or a dedicated vector database (*e.g.,* Pinecone [4], Milvus [2], Weaviate [5]). To avoid confusion, this paper uses the term "vector database" to specially refer to any databases (including database-like files, relational databases, and dedicated vector databases) that store feature vectors.

In vector databases, the original data (before feature extraction) is usually highly privacy-sensitive. For instance, the original data could be personal biometric data, personal health data, commercial valuable data, financial data, etc. If such data is leaked, it can be abused by an adversary. For example, if personal biometric data like face images are compromised, an adversary could use the victim's face image to bypass online authentication systems. Additionally, if personal health data is exposed, it could reveal sensitive information about an individual's health status, leading to a breach of personal privacy. Moreover, the original data could be the intellectual property (IP) of the data owner, and its leakage could result in IP infringements. Hence, it is crucial to ensure the confidentiality of the original data from potential adversaries when using vector databases. However, in this paper, we show that it is possible for an adversary to reconstruct the original data from their feature vectors. Studying such FVI attacks (*i.e.*, data reconstruction attacks) and possible defenses are the central themes of this work.

Limitations of Prior FVI Attacks. The attack techniques developed in prior studies (*e.g.*, [16], [34], [41]) can be used to launch FVI attacks against vector databases. The major limitation of prior FVI attacks in our attack scenario is that their attack success depends on a strong assumption: the adversary can access an auxiliary dataset that is a subset of the original data or a dataset with a similar distribution (*i.e.*, assuming a data-available threat model). The auxiliary dataset enables the adversary to create massive (data, vector) pairs, which can be used to train an FVI attack network to launch the attack. Nevertheless, in practice, an adversary might not always be able to acquire a high-quality auxiliary dataset, especially when the original dataset is highly confidential and kept secret from the public.

To eliminate the strong data-available assumption, this paper aims to study the data-free FVI attack. Compared with the data-available threat model, a successful FVI attack in the

Shengyang Qin, Nankun Mu, and Hongyu Huang are with the College of Computer Science, Chongqing University, Chongqing 400044, China (email: shengyangqin@stu.cqu.edu.cn; nankun.mu@cqu.edu.cn; hyhuang@cqu.edu.cn) (Corresponding Author: Nankun Mu)

Xinyu Lei is with the Department of Computer Science, Michigan Technological University, Houghton, MI 49931 USA (e-mail: xinyulei@mtu.edu).

Tian Xie is with the Department of Computer Science, Utah State University, Logan, UT 84322 USA (e-mail: tian.xie@usu.edu).

Xiao Zhang is with the College of Engineering and Computer Science, University of Michigan-Dearborn, Dearborn, MI 48124 USA (e-mail: zhanxiao@umich.edu).

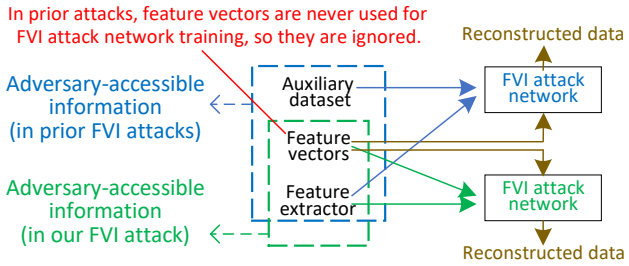


Fig. 1: Comparision between prior attacks and our studied attack.

data-free threat model would pose more severe threats to vector databases because fewer resources are needed by the adversary. Figure 1 compares prior data-available FVI attacks and our data-free FVI attack. In prior FVI attacks, feature vectors are never used for FVI attack network training, so they are ignored. In contrast, our FVI attack leverages feature vectors (that the adversary aims to inverse) to train the FVI attack network, making the data-free FVI attack possible. More specifically, our research is driven by answering the following four research questions (RQ).

- RQ1: Is it feasible for an adversary to launch a successful feature vector inversion attack when the auxiliary dataset is absent? If yes, how to design the attack?
- RQ2: What factors influence the designed feature vector inversion attack performance?
- RQ3: What are the further security implications of the proposed attack technique?
- RQ4: How to design solutions to mitigate the attacks?

Technical Challenges and Proposed Solutions. To answer RQ1, we aim to design a successful data-free FVI attack. There are mainly three technical challenges to be addressed.

First, it is challenging to train the FVI attack network if the dataset used for training is no longer available. To handle this challenge, we develop an **output-to-input data generation technique**. In such a technique, a generative adversarial network variant (GANV) is trained so that its generator can produce synthetic fake training samples. Specifically, the fake sample generator in GANV is trained to generate fake samples such that their extracted feature vectors' distribution is hard to be distinguished from that of adversary-accessible feature vectors. Since the usage of the feature extractor's outputs (*i.e.*, feature vectors) enables the generation of the feature extractor's inputs (*i.e.*, fake samples) in a backward manner, we call the above technique "output-to-input data generation technique".

Second, it is challenging to develop a high-performance size inference algorithm to infer the ground-truth size of the synthetic fake images. In the strict data-free threat model, the original image size information is agnostic to the adversary. It is infeasible to train a high-quality fake sample generator if the synthetic image's size information is unknown. To tackle this challenge, we propose the **accelerable complete bipartite graph (CBG) search strategy**. In the accelerable CBG search strategy, the adversary first trains many different-sized (generator, discriminator) pairs. Then, all generators and all discriminators are mutually connected to form a CBG. The key idea in the accelerable CBG research strategy is: fake samples generated by the generator initialized with the size (that is closest to the ground truth) cheat most of the discriminators in the CBG. Therefore, the adversary can measure the cheating success rate to infer the ground-truth size gradually via a large-step-to-small-step search. In addition, we develop a search space pruning strategy to accelerate the ground-truth size search process. The pruning strategy is developed based on the following critical observation: for some feature extractor types, the input size scope can be deduced based on the output size and network architecture in a backward manner.

Third, in case the label information of feature vectors is available to the adversary, how to leverage the label information to design stronger FVI attacks poses a non-trivial technical challenge. To handle this challenge, we develop the **downstream-classifier-aided generator training strategy**. In this strategy, we first train a downstream classifer using the feature vectors and their labels. Then, we employ two additional losses (*i.e.*, one-hot loss and information entropy loss) defined with the help of the downstream classifier to guide the fake sample generator training. As such, the hard label information conveyed in the downstream classifier can be properly leveraged. It is demonstrated by our experimental results that the downstream-classifier-aided generator training strategy indeed helps to improve FVI attack performance under most circumstances.

Attack Performance Factors. To answer RQ2, we conduct intensive experiments and precisely identify the factors that influence the attack performance. It generally holds that the more background knowledge and capabilities the attacker owns, the higher the attack performance. More experiential details can be found in Section IV-C.

Broader Impact. To answer RQ3, we discover that the proposed output-to-input data generation technique not only leads to FVI attacks in the data-free setting but also impacts prior data-available FVI attacks. Specifically, it can be used to augment the dataset used for training the attack network in the prior dataset-available FVI attacks, thereby improving their attack performance. Therefore, the developed FVI attack technique in the data-free setting helps FVI attacks in the data-available setting.

Defenses. To defend against the proposed FVI attacks, we propose and investigate three types of feature vector perturbation-based defenses including vector rounding, vector thresholding, and vector noising. It should be noted that there is always a tradeoff between security and utility. Through intensive experiments, the effective defenses (that strike the best tradeoffs between data security and data utility) under different settings are accurately recognized.

Contributions. The main contributions of this work are summarized below.

- As far as we know, this paper is the first systematic study of FVI attacks against vector databases in the data-free setting. By characterizing the background knowledge of the adversary across 3 dimensions, we systematically study 8 types of data-free FVI attacks.

- We develop multiple techniques/strategies (including output-to-input data generation technique, accelerable CBG search strategy, and downstream-classifier-aided generator training strategy) to realize the data-free FVI with high attack performance.
- Through intensive experiments of 8 attacks on 4 benchmark datasets, we empirically identify multiple factors that influence the attack performance, thereby deepening our holistic understanding of the vulnerabilities of vector databases against such FVI attacks.
- We are the first to reveal that the proposed output-to-input data generation technique can be directly employed to boost the attack performance of FVI attacks in the traditional data-available setting.
- We uncover that the proposed feature vector perturbation defenses (with proper parameter choices) can mitigate the proposed FVI attacks and the three impacted ML attacks. Our study sheds light on how to design better defenses.

II. Problem Statement

A. Problem Settings

In this paper, we focus on the vector database which stores feature vectors extracted from image data. The used feature extractor is the pre-trained DNN-based. The considered downstream ML task is the most frequently used image classification task.

B. Threat Model

Following the same threat model as prior auxiliary-dataset-available FVI attacks [16], [34], [41], the adversary's goal is to train an FVI attack network to launch the FVI attack from each adversary-accessible feature vector stored in the vector database. It is desirable that the reconstructed data should be close to its original data as much as possible. To have a systematic study of the attack, we characterize the background knowledge of the adversary across the following three dimensions.

1) Knowledge About the Feature Extractor. Our study aims to cover two commonly studied settings in the prior studies: the white-box and the black-box setting. In the white-box setting, the adversary knows the target feature extractor's model details. In the black-box setting, the adversary does not know the target feature extractor's model details, but the adversary can query the feature extractor on inputs and receive the extracted feature vectors as outputs.

Real-World Examples. For the white-box setting, some online platforms (*e.g.*, GitHub) publicly share the pre-trained feature extractor. Consequently, the adversary can obtain the white-box feature extractor through online downloading. For the black-box setting, Encoder-as-a-Service (Eaas) [1], [3] provides the public with a cloud API to convert data into feature vectors. Therefore, the adversary can obtain the feature vector of the input data through invoking the cloud API.

2) Knowledge About the Size Information of Original Images. When an auxiliary dataset is available, the adversary typically views the image size of the auxiliary data as the target image size. In data-free settings, our study intends to address both cases: the adversary knows or does not know the size information of the original images.

Real-World Examples. The adversary may learn the size information of the original images from some side-channel information (*e.g.*, from the size of the accepted images when using the Eaas cloud API; from the image data owner who uses the vector database). In a more strict situation, the size information of the original images is agnostic to the adversary.

3) Knowledge About the Hard Label Information of Feature Vectors in the Database. The hard label is an integer that represents the class of each feature vector. Our study will take two cases into consideration: the adversary can or cannot access the hard label information of the feature vectors.

Real-World Examples. If the feature extractor is pre-trained by supervised learning, the label information may be stored along with the extracted feature vectors. In addition, if the extracted feature vectors are used for training downstream classifiers, the label information should be provided along with the feature vectors. In the above two cases, the adversary can access the label information stored along with feature vectors. In another scenario, if the feature extractor is pre-trained by unsupervised learning, the hard label may be absent.

TABLE I: Attack taxonomy. ✓ (×) means the adversary has (does not have) the knowledge.

Name	$\mathcal{MA}$	$\mathcal{SIZE}$	$\mathcal{LABEL}$	Name	$\mathcal{MA}$	$\mathcal{SIZE}$	$\mathcal{LABEL}$
Attack-1	$\mathcal{W}$	✓	×	Attack-5	$\mathcal{B}$	✓	×
Attack-2	$\mathcal{W}$	✓	✓	Attack-6	$\mathcal{B}$	✓	✓
Attack-3	$\mathcal{W}$	×	×	Attack-7	$\mathcal{B}$	×	×
Attack-4	$\mathcal{W}$	×	✓	Attack-8	$\mathcal{B}$	×	✓

Mathematically, we denote the knowledge about the feature extractor's model access as $\mathcal{MA}$. We have $\mathcal{MA} = \{\mathcal{W}, \mathcal{B}\}$, where $\mathcal{W}$ represents white-box access and $\mathcal{B}$ represents black-box access. The knowledge about the size information of original images is denoted as $\mathcal{SIZE}$ and knowledge about the hard label information is denoted as $\mathcal{LABEL}$. Therefore, we formulate the adversary's background knowledge $\mathcal{K}$ as the following triplet

$$\mathcal{K} = (\mathcal{MA}, \mathcal{SIZE}, \mathcal{LABEL}).$$

According to different background knowledge settings of the adversary, the data-free FVI attacks can be categorized into 8 types, as shown in Table I. In the table, attacks are slightly re-ordered to facilitate our attack methodology description in Section III.

III. Attack Methodologies

This section describes attack methodologies for 8 types of attacks, as shown in Table I.

A. Attack-1

In Attack-1, $\mathcal{K} = (\mathcal{W}, ✓, \times)$. Figure 2 shows the FVI pipeline which consists of three stages: fake sample generator training stage, FVI-Net training stage, and FVI-Net attack stage. Each stage is elaborated as follows.

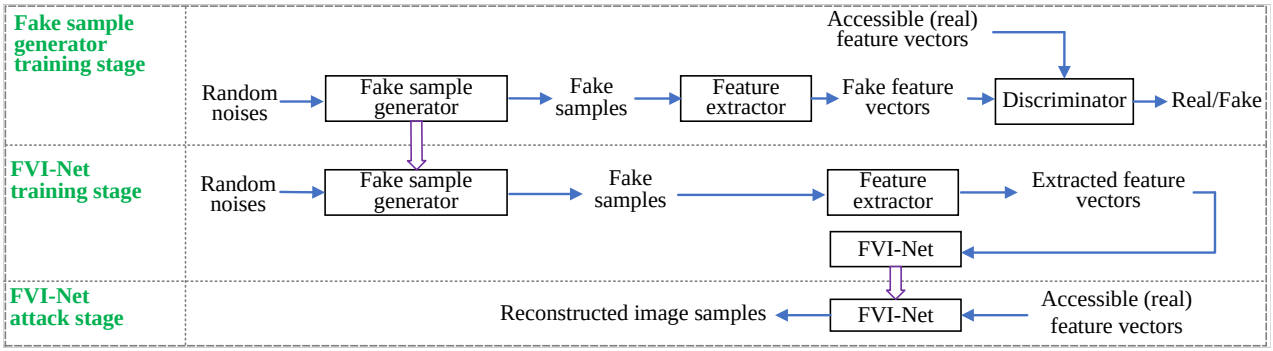


Fig. 2: The FVI attack pipeline (which consists of three stages) for Attack-1. At the first stage, a fake sample generator is trained in a data-free manner. At the second stage, the fake samples output by the well-trained generator are used for FVI-Net training. At the third stage, an adversary-accessible feature vector is fed into FVI-Net to output its reconstructed images. Note that the remaining attacks (*i.e.*, Attack-2 to Attack-8) follow the above three attack stages. In the remaining attacks, different strategies are developed to accommodate different background knowledge owned by the adversary.

1) **Fake Sample Generator Training Stage.** At this stage, we develop an output-to-input data generation technique. In such a technique, a generative adversarial network variant (GANV) is trained so that its generator can produce synthetic fake training samples used for the FVI attack network (named FVI-Net) training. FVI-Net is used to generate the reconstructed image given its feature vector. With the help of the fake sample generator in GANV, FVI-Net can be trained without reliance on an auxiliary database.

Mathematically, we denote the feature extractor and the vector database as F and $\mathcal{D}_v$, where $\mathcal{D}_v$ is extracted by F from an unknown original dataset $\mathcal{D}$. Given a random noise vector $\mathbf{z}$ as input, the generator G outputs a fake sample $\bar{\mathbf{x}}$. Then, the fake sample is forwarded into F that outputs fake feature vector $\bar{\mathbf{v}}$, where $\bar{\mathbf{v}} = F(\bar{\mathbf{x}})$. The fake feature vectors and the adversary-accessible real feature vectors in $\mathcal{D}_v$ can be used to train the discriminator D. Following [19], we express the training objective of GANV as

$$\min_G \max_D \mathcal{L}_{GANV}(G,D) = \mathbb{E}_{\mathbf{v}\sim P_{\mathcal{D}_v(\mathbf{v})}}[\log D(\mathbf{v})] + \mathbb{E}_{\mathbf{z}\sim N(0,1)}[\log(1-[D\circ F\circ G](\mathbf{z}))], \quad (1)$$

where $[D\circ F\circ G](\mathbf{z})$ denotes $D(F(G(\mathbf{z})))$. Through adversarial co-training between G and D, G can be trained to generate high-quality fake samples while D can be trained to evaluate whether a given sample is real or fake. In GANV co-training, the feature extractor F allows the gradients to pass through it, but its weights are fixed and never updated. For a well-trained GANV, feeding a sample to $D \circ F$, a probability can be calculated by D. If the output probability is greater than 0.5, then the input sample is evaluated to be real; otherwise, it is evaluated to be fake.

The GANV developed in Attack-1 is different from the classical GAN in two-fold. First, in GANV, a fixed neural network (*i.e.*, feature extractor) is inserted between the generator and the discriminator. Second, in the classical GAN, the discriminator is used to distinguish between two images. In contrast, in GANV, the discriminator is used to distinguish between two feature vectors. Note that the usage of the feature extractor's outputs (i.e., feature vectors) enables the generation of the feature extractor's inputs (i.e., fake samples) in a backward manner, so we call the above technique "output-to-input data generation technique" in this paper.

2) **FVI-Net Training Stage.** The goal of this stage is to train FVI-Net (denoted as R) that can generate the reconstructed image given its feature vector. To achieve the goal, we train FVI-Net using a cycle-consistence loss. Mathematically, given a fake sample $\bar{\mathbf{x}}$ generated from the well-trained generator G, the cycle-consistency loss is defined as

$$\mathcal{L}_{cyc} = d([R\circ F](\bar{\mathbf{x}}), \bar{\mathbf{x}}), \quad (2)$$

where $d(\cdot,\cdot)$ denotes the conventional ℓ_1 norm. By minimizing $\mathcal{L}_{cyc}$ in training, the well-trained FVI-Net can map a feature vector to its reconstructed image.

3) **Attack Stage.** At the attack stage, the well-trained FVI-Net (denoted as R^*) is used to reconstruct the original data from the corresponding feature vector stored in the vector database. Formally, for each $\mathbf{v} \in \mathcal{D}_v$, the reconstructed image $\mathbf{x}'$ can be obtained by $\mathbf{x}' = R^*(\mathbf{v})$.

B. Attack-2

In Attack-2, $\mathcal{K} = (\mathcal{W}, \checkmark, \checkmark)$. Compared with Attack-1, the adversary has extra knowledge of each feature vector's hard label. As Attack-1, Attack-2 also consists of three stages as shown in Figure 2.

1) **Fake Sample Generator Training Stage.** We aim to use the hard label information to train a better fake sample generator that improves the FVI attack performance. However, it is hard to directly make use of the hard label information to train a better fake sample generator. To address this issue, we develop the downstream-classifier-aided generator training strategy. In this strategy, the adversary uses the feature vectors and their hard labels to train a downstream classifier. With the help of the well-trained classifier, two extra loss functions are employed to guide the training of the fake sample generator. As a result, our strategy allows the adversary to leverage the hard label information to train a better fake sample generator.

To be specific, the adversary can use (vector, label) pairs to train a classifier C via standard supervised learning. After training, feeding a set of fake feature vectors $\{\bar{\mathbf{v}}_1, \bar{\mathbf{v}}_2, \ldots, \bar{\mathbf{v}}_n\}$ into the well-trained classifier C^*, the output logits $\{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n\}$ can be obtained by $\mathbf{t}_i = C^*(\bar{\mathbf{v}}_i)$.

The predicted labels $\{l_1, l_2, \ldots, l_n\}$ are then calculated by $l_i = \arg\max_j (\mathbf{t}_i)_j$. Inspired by [11], we introduce the one-hot loss and information entropy loss as follows.

• One-Hot Loss. An effective generator is expected to produce images that the classifier can assign to a specific category with a higher probability. Formally, the one-hot loss $\mathcal{L}_{oh}$ is given by

$$\mathcal{L}_{oh} = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_{ce}(\mathbf{t}_i, l_i), \tag{3}$$

where $\mathcal{L}_{ce}$ is the classical cross-entropy loss.

• Information Entropy Loss. A good generator is supposed to generate a balanced set, which contains a roughly equal number of synthetic images for each class. To achieve the goal, we introduce information entropy loss as follows. Specifically, given a probability vector $\mathbf{p} = \{p_1, p_2, ..., p_k\}$, the information entropy of $\mathbf{p}$ is calculated as $\mathcal{H}_{info}(\mathbf{p}) = -\frac{1}{k}\sum_i p_i \log(p_i)$. Given a set of output logits $\{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n\}$, the information entropy loss $\mathcal{L}_{ie}$ of fake samples is given by

$$\mathcal{L}_{ie} = -\mathcal{H}_{info}(\mathrm{Norm}(\frac{1}{n}\sum_{i=1}^{n} \mathbf{t}_i)), \tag{4}$$

where $\mathrm{Norm}(\cdot)$ represents the softmax normalization.

To sum up, the final objective function is defined as

$$\mathcal{L}_{total} = \mathcal{L}_{GANV}(G, D) + \beta\mathcal{L}_{oh} + \gamma\mathcal{L}_{ie}, \tag{5}$$

where $\mathcal{L}_{GANV}(G, D)$, $\mathcal{L}_{oh}$, and $\mathcal{L}_{ie}$ can be found in Eq. (1), Eq. (3), and Eq. (4), respectively. The hyperparameters β and γ are adopted to balance different terms. The well-trained generator G can be obtained by solving the final objective function. Given random noises as input, G aims to output images that share a similar distribution of the original dataset.

2) **FVI-Net Training Stage and Attack Stage.** The two stages are consistent with Attack-1, so their descriptions are skipped.

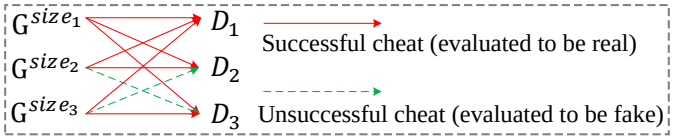


Fig. 3: A simplified example to illustrate our accelerable CBG search strategy. G^{size_i} and D_i are adversarially co-trained.

C. Attack-3

In Attack-3, $\mathcal{K} = (\mathcal{W}, \times, \times)$. Compared with Attack-1, the size information of the original image is not available to the adversary. Therefore, in Attack-3, we design a size inference algorithm to infer the size of the original image data. In the size inference algorithm, we propose the accelerable complete bipartite graph (CBG) search strategy, which is briefly introduced below.

First, we train multiple (generator, discriminator) pairs with different sizes. After training, all generators and all discriminators are mutually connected to form a CBG. The key idea in the accelerable CBG search strategy is based on the following observation: fake images generated by the generator initialized with the size (that is closest to the ground truth) cheat most of the discriminators in the CBG. Therefore, the adversary can measure the cheating success rate to infer the ground-truth size gradually via a large-step-to-small-step search. For brevity, the developed algorithm is called the **L**arge-**S**tep-to-**S**mall-**S**tep (LS^3) search algorithm. Note that in LS^3 search algorithm, we also design a pruning strategy to accelerate the search (see Section III-D).

Figure 3 shows a simplified numerical example to illustrate our accelerable CBG search strategy. As shown, a sample generated by G^{size_1} can cheat all discriminators, whereas a sample generated by G^{size_2} and G^{size_3} can only cheat two of the three discriminators. As a result, $size_1$ is inferred as the size that is closest to the ground truth. In our algorithm, we use multiple samples (instead of one) to increase the inference accuracy.

Specifically, we define the size of an image as a 2-tuple of pixels consisting of height and width (denoted as $h \times w$). For a generator initialized with size $h \times w$, we define a metric called average cheating success rate (ACSR). We denote the number of test fake samples and the number of discriminators as N_t and N_D, respectively. Let $N_s^{h\times w}$ be the number of average successful cheating samples initialized with size $h \times w$ among a set of discriminators. Let $G^{h\times w}$ denote the generator initialized with size $h \times w$. After training, the average number of successful cheating samples on a set of discriminators is denoted as $N_s^{h\times w}$, which is given by

$$N_s^{h\times w} = \frac{1}{N_D}\sum_{i}^{N_D}\sum_{j}^{N_t} \mathbb{I}([D_i \circ G^{h\times w}](\mathbf{z}_j) > 0.5), \tag{6}$$

where $\mathbb{I}(\cdot)$ is the indicator function that equals 1 if the following inequality holds, and 0 otherwise. Note that if the output probability of a discriminator is greater than 0.5, then the input sample is evaluated to be real; otherwise, it is evaluated to be fake. Accordingly, the ACSR of $G^{h\times w}$ is defined as

$$ACSR^{h\times w} = N_s^{h\times w}/N_t, \tag{7}$$

where $ACSR^{h\times w} \in [0, 1]$. After one round of search, a size close to the ground truth can be found. Then, the estimated size scope is narrowed down and a smaller step size is set. Next, the next round of search is performed to obtain a more accurate size. The above search process repeats until the step size is 1.

Algorithm 1 displays LS^3 algorithm. Given a feature extractor F, a vector dataset $\mathcal{D}_v$, an estimated scope $(h_{min}, h_{max}) \times (w_{min}, w_{max})$, and the step size r, this algorithm is expected to output the most likely ground-truth size. In practice, $(h_{min}, h_{max}) \times (w_{min}, w_{max})$ is set to be $(1, 2048) \times (1, 2048)$ since the input size of most image feature extractors is less than 2048×2048. If the ground-truth size is not in the above pre-set scope, a larger value for h_{max} and w_{max} can be set. Specifically, in one round of search (*i.e.*, from Line 4 to Line 18), the algorithm searches for the best $(h^{\star}, w^{\star})$ in a grid manner, where the grid step size is r. Then, from Line 19 to Line 23, the algorithm narrows down the search scope and reduces the step size. Next, the algorithm goes to the next round of search. The above procedures repeat until the step size is 1.

Algorithm 1: LS^3 search algorithm

A vector dataset $\mathcal{D}_v$, a feature extractor F, an estimated scope $(h_{min}, h_{max}) \times (w_{min}, w_{max})$, a step size r. Inferred height $h^{\bigstar}$, inferred width $w^{\bigstar}$.
while $r \geq 1$ **do**
  /* **Prepare stage** */
  Initialize $listG \leftarrow [], listD \leftarrow [], maxACSR \leftarrow 0$, $h \leftarrow h_{min}$;
  while $h \leq h_{max}$ **do**
    $w \leftarrow w_{min}$;
    while $w \leq w_{max}$ **do**
      Initialize $G^{h\times w}$ and $D^{h\times w}$;
      Train $G^{h\times w}$ and $D^{h\times w}$ by minimizing Eq. (1);
      Add $G^{h\times w}$ to $listG$;
      Add $D^{h\times w}$ to $listD$;
      Update $w \leftarrow w + r$;
    Update $h \leftarrow h + r$;
  /* **Inference stage** */
  for $i = 1$ to $ListG$.length **do**
    $G^{h\times w} = ListG[i]$;
    Calculate $ACSR^{h\times w}$ using Eq. (6)-(7);
    if $ACSR > maxACSR$ **then**
      Update $maxACSR \leftarrow ACSR^{h\times w}$;
      Update $h^{\bigstar} \leftarrow h$;
      Update $w^{\bigstar} \leftarrow w$;
  Update $h_{min} \leftarrow h^{\bigstar} - r$;
  Update $w_{min} \leftarrow w^{\bigstar} - r$;
  Update $h_{max} \leftarrow h^{\bigstar} + r$;
  Update $w_{max} \leftarrow w^{\bigstar} + r$;
  Update $r \leftarrow \lfloor r/2 \rfloor$;

With the help of LS^3 algorithm, the adversary can infer a size that is highly likely to be the ground truth. Once the size is accurately inferred, the background knowledge setting of Attack-3 is identical to Attack-1. Therefore, the detailed descriptions of the remaining attack pipelines of Attack-3 are omitted. Remarkably, as shown by our experiments in Section IV-B, the inferred size may have a small deviation from the ground truth. In this case, we have experimentally verified that the attack performance of Attack-2 is only slightly reduced (see Figure 7).

D. Attack-4

In Attack-4, $\mathcal{K} = (\mathcal{W}, \times, \checkmark)$. Compared with Attack-3, the adversary acquires the additional information of hard label. In Attack-4, the adversary can still invoke Algorithm 1 to infer the size of the original image. Once the size information is inferred, Attack-4 reduces to Attack-2. Thus, the detailed descriptions of the attack pipelines of Attack 4 are skipped.

Both Attack-3 and Attack-4 need to invoke Algorithm 1 to infer the size of the original images. We note that Algorithm 1 used CBG search strategy can be accelerated via the search space pruning strategy described below.

Search Space Pruning Strategy. For some types of feature extractors, a search space pruning strategy can be developed by using the information of the feature extractor's output size (before flatted into a vector) and its network architecture. The pruning strategy is developed based on the following critical observation: for some feature extractor types, the input size scope can be deduced based on the output size and network architecture. Hence, the input size search space can be narrowed down and Algorithm 1 can be accelerated.

We use the most commonly used convolutionary neural network (CNN)-based feature extractor as an example to illustrate our idea. Mathematically, we use $k_h \times k_w$ to denote the kernel size, use $strd$ to represent the stride, and use p to denote the number of rows or columns added to all four sides of the input. Accordingly, a feature processing layer (FPL) can be viewed as a function $\mathbf{y} = FPL(k_h, k_w, s, p, \mathbf{x})$, where FPL usually performs dimension reduction of $\mathbf{x}$. Let h_{in} and h_{out} represent the height of $\mathbf{x}$ and $\mathbf{y}$, respectively. Let w_{in} and w_{out} represent the width of $\mathbf{x}$ and $\mathbf{y}$, respectively. Their relationships are given by

$$h_{out} = \lfloor \frac{h_{in} - k_h + 2p}{strd} + 1 \rfloor, w_{out} = \lfloor \frac{w_{in} - k_w + 2p}{strd} + 1 \rfloor. \quad (8)$$

Accordingly, we have the following theorem.

Theorem 1. *Given an input* $\mathbf{x}$ *with size* $h_{in} \times w_{in}$ *and its FPL-output* $\mathbf{y}$ *with size* $h_{out} \times w_{out}$*, it holds that*

$$h_{in} \in [h_{out} \times strd + C_0 - strd, h_{out} \times strd + C_0 - 1],$$
$$w_{in} \in [h_{out} \times strd + C_1 - strd, h_{out} \times strd + C_1 - 1],$$

where $C_0 = k_h - 2p$ *and* $C_1 = k_w - 2p$.

Proof. According to Definition 1 and Eq. (8), we have

$$h_{out} = \lfloor \frac{h_{in} - k_h + 2p}{strd} + 1 \rfloor. \quad (9)$$

According to Eq. (9), it holds that

$$h_{out} \leq \frac{h_{in} - k_h + 2p}{strd} + 1 < h_{out} + 1. \quad (10)$$

Eq. (10) can be re-written as

$$(h_{out}-1)\times strd+k_h-2p \leq h_{in} < h_{out}\times strd+k_h-2p. \quad (11)$$

By setting $C_0 = k_h - 2p$ in Eq. (11), we have

$$h_{in} \in [(h_{out} - 1) \times strd + C_0, h_{out} \times strd + C_0). \quad (12)$$

Because $h_{in} \in \mathbb{Z}^+$, Eq. (12) is equivalent to

$$h_{in} \in [(h_{out} - 1) \times strd + C_0, h_{out} \times strd + C_0 - 1]. \quad (13)$$

Similar to the analysis for the height of input, the width of input follows

$$w_{in} \in [(w_{out} - 1) \times strd + C_1, w_{out} \times strd + C_1 - 1], \quad (14)$$

where $C_1 = k_w - 2p$. □

According to Theorem 1, given the output size in the last layer of the CNN-based feature extractor, the adversary can successively infer the range of the input size for the previous layer until the first layer. For example, a LeNet-5-based feature extractor contains 4 feature processing layers. They can be sketched as $\mathbf{y}_1 = FPL(5, 5, 1, 1, \mathbf{x})$, $\mathbf{y}_2 = FPL(2, 2, 2, 0, \mathbf{y}_1)$, $\mathbf{y}_3 = FPL(5, 5, 1, 1, \mathbf{y}_2)$, $\mathbf{y} = FPL(2, 2, 2, 0, \mathbf{y}_3)$, where $\mathbf{y}_i$ is the output of i-th layer. If $\mathbf{y}$ is 5×5, then it can be deduced that $h_{in} \in [32, 35]$ and $w_{in} \in [32, 35]$ according to Theorem 1.

E. Attack-5

In Attack-5, $\mathcal{K} = (\mathcal{B}, \checkmark, \times)$. The only difference between Attack-1 and Attack-5 is that the feature extractor is accessed in the white-box manner in Attack-1, whereas it is accessed in the black-box manner in Attack-5. The black-box access of the feature extractor interrupts the backpropagation channel,

so it is impossible to train the fake sample generator using the classical backpropagation algorithm [31], [33]. To address the issue, our key idea is to employ the zeroth-order method [12] to estimate the gradients backpropagated through the black-box feature extractor. Like Attack-1, Attack-5 also consists of three stages.

1) **Fake Sample Generator Training Stage.** The loss function used in Attack-5 is the same as the loss function used in Attack-1 (*i.e.*, $\mathcal{L}_{GANV}$ in Eq. (1)) since Attack-1 is the corresponding white-box version of Attack-5. In Attack-5, the zeroth-order method is employed to estimate gradients, making it possible to train the fake sample generator. We use F_b to represent the black-box feature extractor. To estimate the gradient of the black-box feature extractor F_b on an input $\bar{\mathbf{x}}$, the adversary can first evaluate the loss function values at two very close points $\bar{\mathbf{x}} + h\mathbf{e}$ and $\bar{\mathbf{x}} - h\mathbf{e}$, where h is a small step size constant and the vector $\mathbf{e}$ specifies the direction on which the gradient is estimated. Accordingly, the gradient of F_b at $\bar{\mathbf{x}}$ along the direction $\mathbf{e}$ can be estimated as $[F_b(\bar{\mathbf{x}} + h\mathbf{e}) - F_b(\bar{\mathbf{x}} - h\mathbf{e})]/2h$. It has been proved that the estimation error can be reduced if $h \to 0$ [12]. Hence, generator G can still be well-trained by estimating gradients passing through the black-box feature extractor.

2) **RNet Training Stage and Attack Stage.** Once the gradients are estimated, Attack-5 is reduced to Attack-1, Therefore, the two stages are the same as Attack-1 (see Section III-A).

F. Remaining Attacks

The remaining attacks include Attack-6, Attack-7, and Attack-8. Their white-box versions are Attack-2, Attack-3, Attack-4, respectively. Consequently, all of them can be designed following the same methodology as Attack-5: first using the same loss function as their corresponding white-box versions and then employing the zeroth-order method to estimate the gradients pass through the black-box feature extractor. Hence, the detailed descriptions of the remaining attacks are omitted.

IV. Experiments

A. Experiment Setup

Datasets. We extract vector datasets from image datasets, which are introduced as follows. The extracted vector datasets are used in our experiments. For simplicity, we still use the name of the image dataset for the corresponding vector dataset.

- MNIST [15]: This dataset contains 70,000 grayscale images categorized into 10 handwritten digit classes, each being 28×28 pixels.
- Fashion-MNIST [42]: This dataset contains 70,000 grayscale images categorized into 10 clothing classes with each image sized at 28x28 pixels.
- CIFAR10 [26]: This dataset contains 60,000 color images across 10 distinct classes. Each image is 32×32 pixels.
- CelebA [29]: This dataset contains more than 200,000 colored face images with 40 different binary attributes. Following [10], we create an 8-class classification task by concatenating the top three attributes.

Feature Extractors. For each dataset, we employ two commonly used DNN-based feature extractors. The used feature extractors for different datasets are summarized below.

- For MNIST and Fashion-MNIST, we use ResNet18 (default) [20] and LeNet-5 [27] as feature extractors. To suit the feature extractor architecture, the original images are resized to 32×32 pixels in the image pre-processing step.
- For CIFAR10, we use ResNet34 (default) [20] and VGG19 [36] as feature extractors.
- For CelebA, we use IresNet18 (default) [17] and Sphere20 [28] as the feature extractors. The default feature extractor is IresNet18.

In our experiments, unless explicitly stated, the feature extractor for each dataset uses the default one marked above. For all datasets, the feature extractors are pre-trained using supervised learning. For MNIST, Fashion-MNIST, and CIFAR10 datasets, we use the popular approach of fine-tuning pre-trained model [39], by which the feature extractors are first pre-trained on ImageNet as initialization for fine-tuning. For CelebA dataset, we adopt the algorithm in Arcface [14] to pre-train IresNet18 and use the algorithm in Cosface [40] to pre-train Sphere20.

Evaluation Metrics. We use the following metrics to assess the attack performance.

1) Average pixel difference (AvgPD): AvgPD is defined as the average pixel-wise distance between the original image and its reconstructed version. This metric is defined as $AvgPD(\mathbf{x}, \mathbf{x}') = \frac{1}{M}\sum_{m=1}^{M} |\mathbf{x}_m - \mathbf{x}'_m|$, where M is the total number of pixels in the image. Generally, the smaller AvgPD, the better the attack performance.
2) Mean square error (MSE): MSE measures the average of the squares of the errors. This metric is defined as $MSE(\mathbf{x}, \mathbf{x}') = \frac{1}{M}\sum_{m=1}^{M}(\mathbf{x}_m - \mathbf{x}'_m)^2$. Generally, the smaller MSE, the better the attack performance.
3) Learned perceptual image patch similarity (LPIPS): LPIPS quantifies image similarity by comparing deep feature representations from visual models trained with human-annotated similarity judgments [44]. The LPIPS metric has been demonstrated to be more consistent with the human visual system's perception of image similarity than the MSE distance. It is defined as $LPIPS(\mathbf{x}, \mathbf{x}') = \sum_j \frac{1}{W_j H_j}\sum_{h,w} ||w_j \odot (\mathbf{x}^j_{hw} - \mathbf{x}'^j_{hw})||_2^2$, where w_j contains weights for each of the features in layer j, $||\cdot||_2$ represents ℓ_2 distance, and $\odot$ denotes the multiplication of the feature vectors at each pixel by the feature weights. In addition, $\mathbf{x}^j_{hw}$ and $\mathbf{x}'^j_{hw}$ denote the normalized feature vectors (of $\mathbf{x}$ and $\mathbf{x}'$) at layer j with the pixel location at (h, w). Generally, the smaller LPIPS, the better the attack performance.

Implementation, Network Architectures, and Training Details. See Appendix A.

B. Experimental Results for 8 Attacks

Attack-1 & Attack-2.

Visualization Results. Figure 4 visualizes some instances of the ground-truth images and the corresponding reconstructed images under different attack settings. There are two observations. First, images reconstructed by Attack-1 and Attack-2

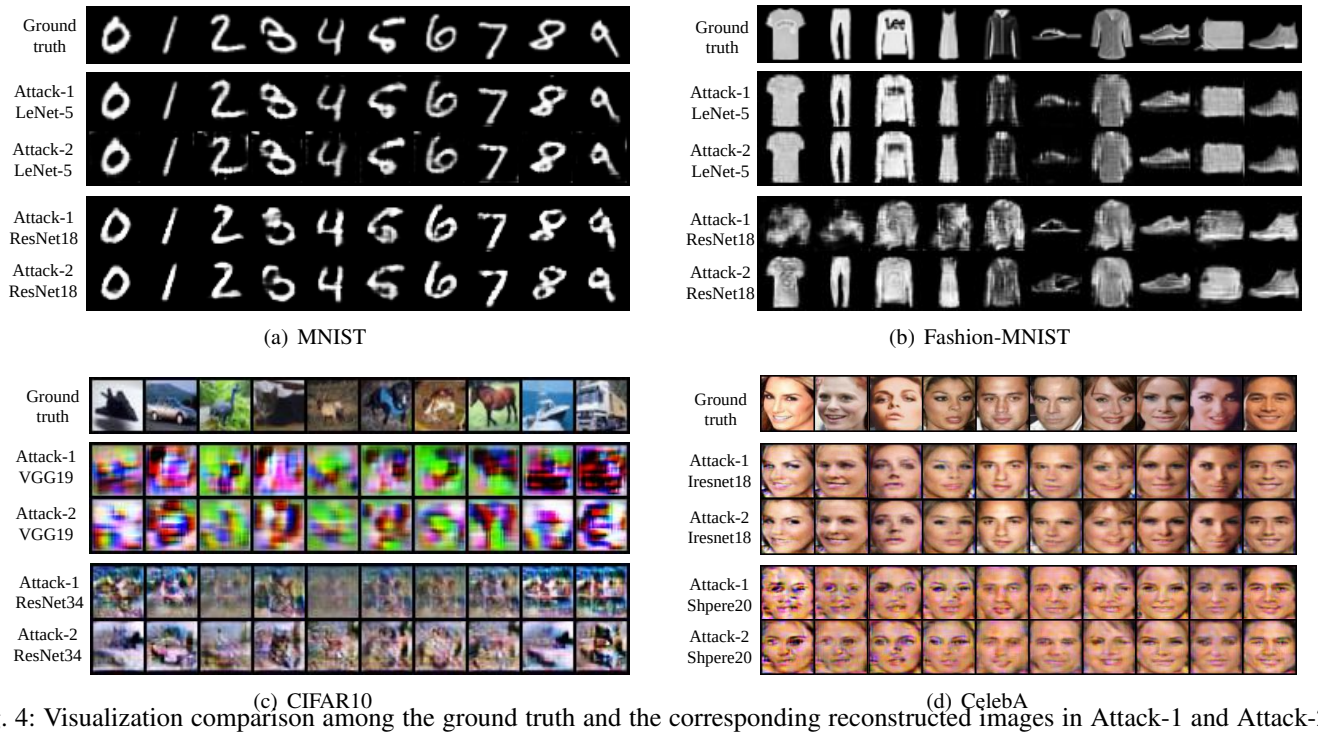


Fig. 4: Visualization comparison among the ground truth and the corresponding reconstructed images in Attack-1 and Attack-2. The ground-truth images are shown in the first row, whereas the reconstructed images are shown in rows 2-5.

TABLE II: Experimental results in terms of three evaluation metrics under Attack-1 and Attack-2. A-i is the abbreviation for Attack-i. The symbol $\downarrow$ means the smaller the value, the better the attack performance. The best (*i.e.*, lowest) quantitative results are highlighted in **Bold**.

Datasets	MNIST				Fashion-MNIST				CIFAR10				CelebA			
Feature extractors	LeNet5		ResNet18		LeNet5		ResNet18		VGG19		ReNet34		IresNet18		Sphere20	
Attack types	A-1	A-2	A-1	A-2	A-1	A-2	A-1	A-2	A-1	A-2	A-1	A-2	A-1	A-2	A-1	A-2
AvgPD $\downarrow$	**0.023**	0.030	**0.060**	0.061	**0.044**	0.045	0.174	**0.093**	0.390	**0.380**	**0.372**	**0.372**	0.113	**0.112**	0.173	**0.165**
MSE$\downarrow$	**0.004**	0.007	0.028	**0.027**	**0.007**	**0.007**	0.078	**0.028**	0.223	**0.211**	**0.181**	0.191	0.025	**0.024**	0.051	**0.047**
LPIPS $\downarrow$	**0.018**	0.031	0.055	**0.052**	**0.086**	0.090	0.203	**0.094**	0.406	**0.403**	0.373	**0.320**	**0.173**	0.182	0.283	**0.276**

are clearly recognizable using two different feature extractors on MNIST, Fashion-MNIST, and CelebA datasets. However, the attack performance is comparatively worse on CIFAR10 dataset. Second, under most circumstances, images reconstructed using Attack-2 are clearer than those using Attack-1 since the additionally available label information in Attack-2 contributes to the FVI.

Quantitative Results. Table II shows experimental results in terms of three evaluation metrics under Attack-1 and Attack-2. The results are averaged on 1,000 tests. There are three discoveries from Table II. First, both Attack-1 and Attack-2 have small AvgPD, MSE, and LPIPS on MNIST, Fashion-MNIST, and CelebA datasets. For instance, we have MSE $= 0.004$ on MNIST using LeNet-5, MSE $= 0.007$ on FashionMNIST using LeNet-5, and MSE $= 0.025$ on CelebA using IresNet18. In contrast, the metrics AvgPD, MSE, and LPIPS are relatively large on CIFAR10 dataset. Second, compared with using ResNet18 as the feature extractor, the attack performance is better if LeNet5 is used. Third, under most circumstances, Attack-2 achieves better attack performance than Attack-1. Compared with Attack-1, Attack-2 leads to AvgPD decrease by 6.75% on average, MSE decrease by 9.33% on average, and LPIPS decrease by 9.22% on average.

The attack performance is comparatively worse on CIFAR10 dataset. This is consistent with the traditional auxiliary-dataset-available FVI attacks on CIFAR10 dataset. The possible reason is that images in CIFAR10 have greater varieties. The limited number of adversary-accessible feature vectors cannot well cover such varieties, so the well-trained fake sample generator cannot generate sufficient diversified fake samples to train FVI-Net with high attack performance. It requires much more highly diversified feature vectors to train a good fake sample generator to enhance FVI-Net training. Hence, the attack performance on CIFAR10 is expected to be further improved if more diversified feature vectors are available. Although attack performance on CIFAR10 dataset is comparatively worse, some reconstructed images still leak critical information about their original images. For example, as shown in Figure 4(c) (in the last row and second column), a vague vehicle is still recognizable from the reconstructed image. For the other three datasets, attack performance is comparatively better, indicating that the proposed FVI attacks indeed lead to severe security concerns.

Attack-3 & Attack-4. In the two attacks, the adversary cannot

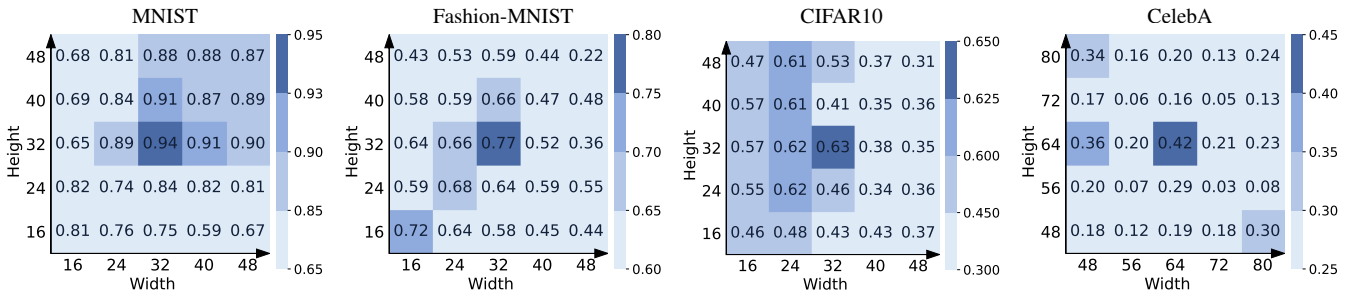


Fig. 5: ACSR maps of the fake sample generators initialized with different sizes plotted on MNIST, FMNIST, CIFAR10, and CelebA, respectively. The step size used in LS^3 algorithm is 8. Each entry in the map denotes an ACSR value. In each map, the central entry is the ground-truth size. The largest ACSR is obtained if the tested size matches the ground truth.

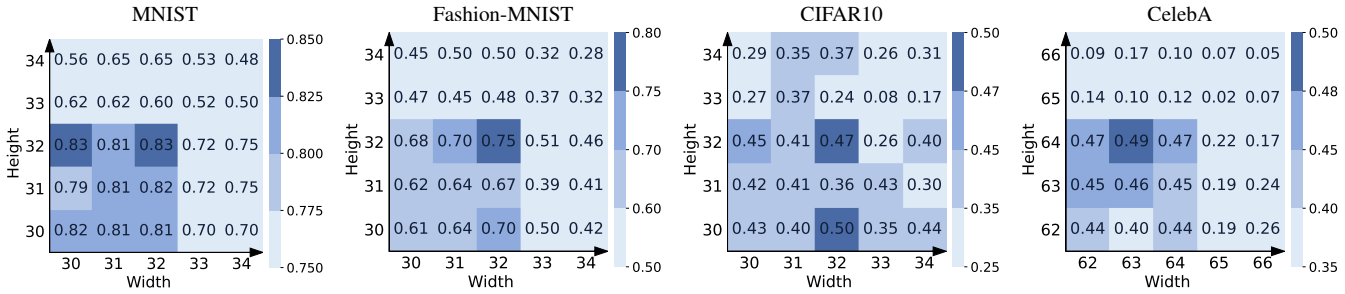


Fig. 6: ACSR maps of the fake sample generators initialized with different sizes plotted on MNIST, FMNIST, CIFAR10, and CelebA, respectively. The step size used in LS^3 algorithm is 1. In each map, the central entry is the ground-truth size. The largest ACSR is obtained if the tested size is close to ground truth.

acquire the size information of the original image, so both attacks need to invoke the size inference algorithm. If the size is inferred correctly, Attack-3 and Attack-4 are reduced to Attack-1 and Attack-2, respectively. Thus, we skip the attack performance evaluation and focus on evaluating the performance of the size inference algorithm instead.

Size Inference Algorithm Evaluation.

ACSR Map. When the fake sample generator is initialized with a certain size, an ACSR score can be computed according to Eq. (7). Therefore, by varying the size, an ACSR map (in the form of a 2D matrix) can be plotted. The ACSR map helps to understand the performance of the proposed LS^3 algorithm, so we plot the ACSR map under two cases as follows.

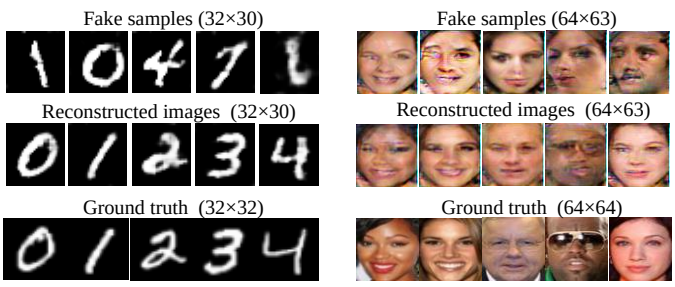
• Case 1 (step size $r = 8$): We aim to plot the ACSR map of the fake sample generators initialized with different sizes if the step size is large (*i.e.*, step size $r = 8$). Figure 5 plots ACSR maps of the fake sample generators initialized with different sizes for different datasets. For MNIST, Fashion-MMIST, and CIFAR10 datasets, we show the results for the size scope $[16, 48]$. For CelebA dataset, we show the results for the size scope $[48, 80]$. It can be observed that ACSR is generally decreasing with an increasing gap between the tested size and the ground-truth size. Additionally, the generator initialized with the ground-truth image size has the largest ACSR. Note that if we extend the tested size scope, identical observations can be obtained. Therefore, if the step size is relatively large (*i.e.*, $r = 8$), then it is not hard for LS^3 algorithm to find a size that is close to the ground truth and then properly shrink the scope for further size inference.

• Case 2 (step size $r = 1$): LS^3 algorithm searches the size until the step size $r = 1$. Therefore, we plot the ACSR map of the fake sample generators initialized with different sizes if $r = 1$. Figure 6 shows the plotted ACSR map, in which we have two findings. First, ACSR is generally decreasing with an increasing gap between the tested size and the ground-truth size. Second, the initialized size of the generator with the highest ACSR may have a small error compared to ground truth. Hence, LS^3 algorithm can infer a size that is close to ground truth.

LS^3 Algorithm Performance. Table III displays the comparison between the ground truth and the LS^3-inferred size. It is evident that LS^3 algorithm is capable of inferring the ground-truth size (or a size that is very close to the ground truth).

TABLE III: Comparison between the ground truth and LS^3-inferred size.

Datasets	MNIST	Fashion-MNIST	CIFAR10	CelebA
Ground truth	32×32	32×32	32×32	64×64
LS^3-inferred size	32×30	32×32	30×32	64×63



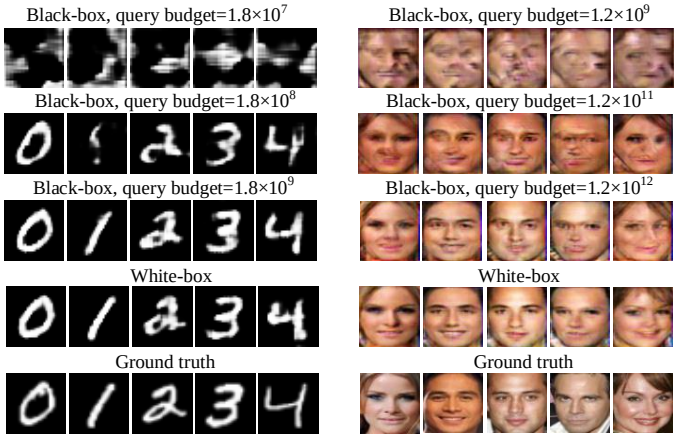
(a) MNIST (b) CelebA

Fig. 7: First row: the generated fake samples with inaccurate inferred size. Second row: reconstructed images with inaccurate inferred size. Third row: the corresponding ground-truth images.

TABLE IV: Quantitative results of different attacks and CelebA under white-box setting and black-box setting. A-i is the abbreviation for Attack-i. The quantitative results in the white-box setting serve as the lower bound of these metrics. The best (*i.e.*, lowest) quantitative results under the black-box setting are highlighted in **Bold**.

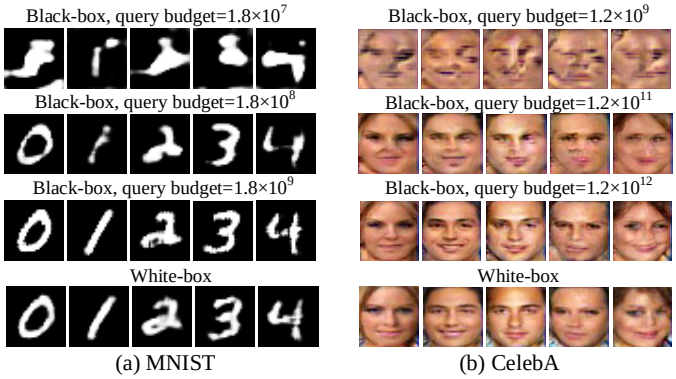
Dataset-Feature extractor	MNIST-ResNet18							CelebA-IresNet18						
$\mathcal{MA}$	$\mathcal{B}$		$\mathcal{B}$		$\mathcal{B}$		$\mathcal{W}$	$\mathcal{B}$		$\mathcal{B}$		$\mathcal{B}$		$\mathcal{W}$
Query budget	1.8×10^7		1.8×10^8		1.8×10^9		-	1.2×10^9		1.2×10^{10}		1.2×10^{11}		-
Attack types	A-5	A-6	A-5	A-6	A-5	A-6	-	A-5	A-6	A-5	A-6	A-5	A-6	-
AvgPD↓	0.316	0.162	0.079	0.079	**0.063**	0.065	0.060	0.164	0.170	0.152	0.138	0.127	**0.125**	0.112
MSE↓	0.212	0.117	0.043	0.044	**0.030**	0.033	0.027	0.043	0.046	0.037	0.034	0.030	**0.029**	0.024
LPIPS↓	0.288	0.169	0.086	0.089	**0.064**	0.097	0.052	0.298	0.338	0.228	0.214	**0.192**	0.197	0.173

Impact of Inferred Size Inaccuracy. As shown in Table III, the size inference algorithm may have a small error, so we conduct experiments to measure the impact of inferred size inaccuracy on attack performance. According to Table III, we set the inferred size as 32×30 on MNIST and set the inferred size as 64×63 on CelebA. Figure 7 shows experimental results. We find that a small error between the inferred size and ground-truth size only leads to subtle attack performance degradation.



(a) MNIST (b) CelebA

Fig. 8: Reconstructed images under the black-box setting (with different query budgets) in Attack-5 and the white-box setting (*i.e.*, Attack-1).



(a) MNIST (b) CelebA

Fig. 9: Reconstructed images under the black-box setting (with different query budgets) in Attack-6 and the white-box setting (*i.e.*, Attack-2).

Attack-5 & Attack-6. For simplicity, two representative datasets (*i.e.*, MNIST and CelebA) are used to evaluate the attack performance of Attack-5 and Attack-6.

Visulization Results. Figure 8 and Figure 9 visualize some instances of reconstructed images in Attack-5 and Attack-6 in different settings. The query budget is the maximum number of queries allowed to query the black-box feature extractor. For both Attack-5 and Attack-6, the white-box FVI attack performs better than their corresponding black-box versions. In addition, with an increasing query budget, the attack performance of Attack-5 gradually approaches that of Attack-1 (*i.e.*, the white-box version of Attack-5). The same law can be discovered between Attack-6 and Attack-2.

Quantitative Results. Table IV shows the quantitative results of Attack-5 and Attack-6 on MNIST and CelebA in terms of three evaluation metrics under different settings. The quantitative results of Attack-5 and Attack-6 are consistent with the visualization results.

Attack-7 & Attack-8. For Attack-7 and Attack-8, the adversary should first infer the original image size. As shown in experiments in Table III, the size inference algorithm can infer the ground truth with no or a small error. If the size is inferred, Attack-7 and Attack-8 are reduced to Attack-5 and Attack-6, respectively. Thus, the attack performance evaluation of Attack-7 and Attack-8 is skipped.

C. Impact of Different Factors

Impact of Number of Feature Vectors. We choose Attack-1 on MNIST and CelebA to study the impact of the number of feature vectors on the attack performance. The laws observed from Attack-1 are followed by other attacks. Figure 10 shows the AvgFD as the function of the number of available feature vectors in the vector database. Figure 11 shows the MSE and the LPIPS as the function of the number of available feature vectors in the vector database. It can be revealed from Figure 10 that the more feature vectors, the better the attack performance. Moreover, some private information (*e.g.*, digit or face characteristics) can be uncovered from the reconstructed images even if there are only 100 feature vectors on MNIST and 1,000 feature vectors on CelebA.

Impact of Training Strategies of Feature Extractor. Other than supervised training of the feature extractor, another popular strategy is self-supervised learning. Thus, we choose Attack-1 to study the impact of two distinct training strategies of feature extractors. The laws observed from Attack-1 are followed by other attacks. The supervised training strategy can be found in the experimental setup in Section IV-A. On

TABLE V: Quantitative results of the feature vector inversion attack on four datasets with two different feature extractor training strategies: supervised learning and self-supervised learning. The symbol ↓ means the smaller the value, the better the attack performance.

Dataset Feature extractor	MNIST ResNet18		Fashion-MNIST LeNet-5		CIFAR10 ResNet34	
Training Strategy	Supervised	Self-Supervised	Supervised	Self-Supervised	Supervised	Self-Supervised
AvgPD↓	0.060	0.152	0.044	0.045	0.372	0.427
MSE↓	0.028	0.089	0.007	0.008	0.181	0.277
LPIPS↓	0.055	0.187	0.086	0.087	0.373	0.333

TABLE VI: Comparison between the attack performance of Attack-1, Attack-2, and the data-available FVI attack (i.e., baseline attack) in [30].

Datasets	MNIST			Fashion-MNIST			CIFAR10			CelebA		
Attacks	Attack-1	Attack-2	Baseline [30]	Attack-1	Attack-2	Baseline [30]	Attack-1	Attack-2	Baseline [30]	Attack-1	Attack-2	Baseline [30]
AvgPD↓	0.060	0.061	0.058	0.174	0.093	0.081	0.372	0.372	0.269	0.113	0.112	0.101
MSE↓	0.028	0.027	0.024	0.078	0.028	0.022	0.181	0.191	0.140	0.025	0.024	0.020
LPIPS↓	0.055	0.052	0.051	0.203	0.094	0.130	0.373	0.320	0.399	0.173	0.182	0.253

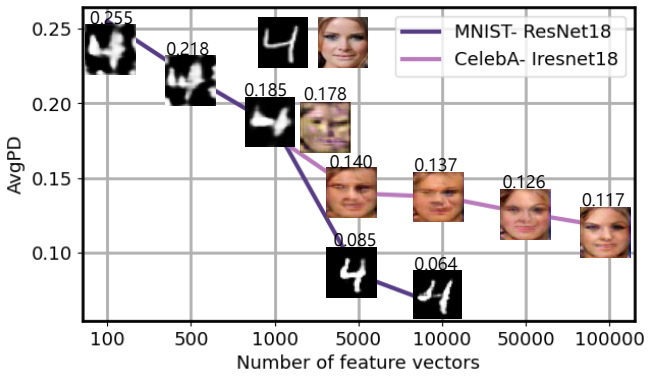


Fig. 10: AvgPD v.s. the number of feature vectors.

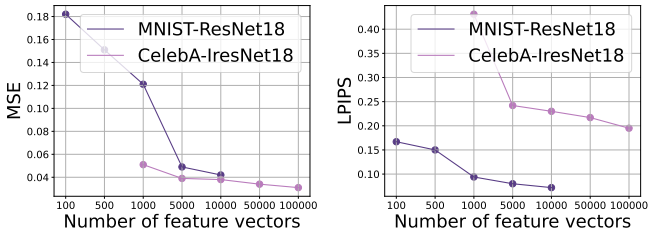


Fig. 11: MSE and LPIPS v.s. the number of feature vectors.

MNIST, Fashion-MNIST, and CIFAR10, we choose SimCLR [13] (a popular framework for self-supervised learning) as their training strategy adopted by training feature extractors.

The experimental results are shown in Table V. It can be observed that the feature extractors trained via supervised learning lead to better attack performance than the corresponding ones trained via self-supervised learning. For instance, we have MSE=0.060 on MNIST using ResNet18 trained via a supervised training strategy, whereas we have MSE=0.152 if a self-supervised strategy is used. The possible reason is that the feature extractor trained via supervised learning can extract feature vectors with higher quality. If the feature vectors have higher quality, then they can be fed to FVI-Net to output reconstructed images with higher quality, leading to better attack performance.

D. Comparsion with Data-Available FVI Attack

We compare our data-free FVI attacks with the prior auxiliary-data-available FVI attack [30]. In the experiments, we let the adversary have an auxiliary dataset, where its cardinality is 10% of the original image dataset. The auxiliary dataset is used to train the FVI attack network. Table VI shows the comparison between the attack performance of Attack-1, Attack-2, and the data-available FVI attack. It is found the attack performance of Attack-1 and Attack-2 is slightly worse than the data-available attack. This is because the data-available attack depends on the strong assumption that the adversary has part of the real training dataset.

V. Impacts on Prior Data-Available FVI Attacks

We note that the proposed output-to-input data generation technique can be employed in any prior auxiliary-dataset-available FVI attacks to boost their attack performance. In the proposed attacks, we develop the output-to-input data generation technique, which can be employed to augment the dataset used for training the attack network in the traditional auxiliary-dataset-available FVI attacks. As a result, the proposed output-to-input data generation technique can be used in any prior auxiliary-dataset-available FVI attacks (*e.g.*, [30], [34]) to boost their attack performance, leading to the state-of-the-art attack performance in the data-available threat model.

TABLE VII: Attack performance of the auxiliary-data-only FVI attack (Attack I) and the auxiliary-dataset-GANV-augmentation FVI attack (Attack II). The symbol ↓ means the smaller the value, the better the attack performance.

Datasets	MNIST		CelebA	
Attacks	Attack I	Attack II	Attack I	Attack II
AvgPD↓	0.058	$0.049\downarrow_{0.009}$	0.101	$0.100\downarrow_{0.001}$
MSE↓	0.024	$0.019\downarrow_{0.005}$	0.020	$0.020\downarrow_{0.000}$
LPIPS↓	0.051	$0.038\downarrow_{0.013}$	0.253	$0.172\downarrow_{0.069}$

We conduct the following experiments to confirm our claim. In our experiments, suppose that the adversary has a small auxiliary dataset with the same distribution as the original

dataset. The cardinality of the auxiliary dataset is 10% of the original image dataset. We use two attack strategies to train FVI-Net. 1) *Auxiliary-Dataset-Only FVI Attack.* Since the auxiliary dataset is available, the adversary can use images in the auxiliary dataset as outputs and their corresponding feature vectors as inputs to directly train FVI-Net. 2) *Auxiliary-Dataset-GANV-Augmentation FVI Attack*. Other than the auxiliary dataset, the output-to-input data generation technique can be additionally used to augment the original auxiliary dataset to train FVI-Net. Table VII exhibits the attack performance for the two attacks. As expected, the auxiliary-dataset-GANV-augmentation inversion attack achieves better attack performance than the auxiliary-dataset-only inversion attack. For example, it holds that LPIPS$= 0.253$ using the auxiliary-dataset-only inversion attack on CelebA. Using GANV-augmentation, we have LPIPS$= 0.172$, leading to a decrease of 0.069.

Remarkably, in auxiliary-dataset-available FVI attacks, the adversary can also augment data from the auxiliary dataset (that serves as the inputs of the feature extractor), so it is named input-to-input data augmentation technique. The output-to-input and input-to-input data augmentation techniques are orthogonal (not mutually exclusive), so they can be used together to increase the diversity of the training dataset to boost the FVI attack performance.

VI. Defenses

A. Defense Methodologies

To defend against targeted attacks (*i.e.*, Attack-1 to Attack-8), we propose the feature vector perturbation method, in which slight modifications are enforced on the feature vectors to mitigate the targeted attacks. The reasons why the feature vector perturbation method is effective are two-fold.

- **R1**: If the feature vectors are perturbed, then the distribution of the GANV-generated fake samples will drift away from the ground truth. Therefore, when these drifted fake samples are used to train FVI-Net, the attack performance of FVI-Net would be reduced.
- **R2**: Instead of using an original feature vector, if a perturbed feature vector serves as an input to FVI-Net, the output of FVI-Net (*i.e.*, the reconstructed image) will be less accurate, leading to attack performance degradation.

Specifically, we investigate three types of feature vector perturbation methods, which are introduced below.

- **Feature Vector Thresholding (FV-T).** For FV-T, in each feature vector, only the largest k percentage of values are maintained, and the others are set to zero. For instance, FV-T (Top 45%) means only maintaining the largest 45 percentage of feature vector entries.
- **Feature Vector Noising (FV-N).** For FV-N, the Gaussian noise with standard deviation set to a percentage of the current feature vector values' standard deviation is added to all entries in each feature vector. For example, FV-N ($2 \times$ STD) means adding 2 times of the current feature vector values' standard deviation to all entries in each feature vector.
- **Feature Vector Rounding (FV-R).** For FV-R, it rounds feature vectors to specified decimal points, thus decreasing the resolution of the feature vector values. For instance, FV-R (INT) denotes all entries in each feature vector are rounded to an integer.

Since modifications to the feature vectors may reduce their data utility for downstream tasks, there usually exists a tradeoff between attack resistance performance (*i.e.*, security) and data utility. Because of the security-utility tradeoff, a successful defense should largely reduce the attack performance while still keeping the data utility as much as possible. In the following, we conduct experiments to evaluate the performance of the three feature vector perturbation methods.

B. Defense Experiments

Experimental Setting. For simplicity, we choose the most representative attack (*i.e.*, Attack-1) to evaluate the performance of the proposed defenses.

Evaluation Metrics. To study the tradeoffs between security and utility, two types of metrics are measured. For security metrics, we re-use the metrics for measuring the attack performance, including AvgPD, MSE, and LPIPS (see Section IV-A). For the utility metric, we use the following three metircs. First, we use downstream classifier accuracy (DCA), which is the accuracy of the downstream classifier trained on the perturbed feature vectors. Second, we use Adjusted Rand Index (ARI) [25] to evaluate the utility of perturbed feature vectors if the downstream task is clustering for MNIST, Fashion-MNIST, and CIFAR-10. Third, as cosine similarity (CosS) metric is widely used in face recognition tasks, CosS is employed to assess the similarity between original feature vectors and perturbed feature vectors on the CelebA dataset. Both security and utility metrics will be compared with the baseline (no defense).

Experimental Results. Table VIII-XI summary the detailed experimental results of the three types of feature vector perturbation defenses against data-free inversion attacks on four datasets. As shown in Table VIII-XI, the three utility metrics exhibit similar security-utility tradeoffs across different datasets when using these defense methods. The defense performance of each defense is analyzed as follows.

- Feature Vector Thresholding (FV-T). From experimental results, it can be discovered that FV-T achieves a good performance in preserving the utility of the vector database while resisting the inversion attack on Fashion-MNIST using FV-T (Top 60%), CIFAR10 using FV-T (Top 60%), and CelebA using FV-T (Top 75%). On MNIST, FV-T (Top 45%) can preserve the utility of the vector database but its defense performance is not very satisfactory.
- Feature Vector Noising (FV-N). From experimental results, it is uncovered that FV-N ($1 \times$ STD) strikes a good security-utility tradeoff on MNIST and CIFAR10. As a numerical example, on MNIST dataset, FV-N ($1 \times$ STD) increases the MSE to 0.228 (up from 0.028 with no defense), while it only reduces the DCA by 0.01%, and reduces the ARI by 0.22%. On Fashion-MNIST, FV-N ($0.5 \times$ STD) shows a kind of worse security-utility tradeoff compared with FV-T (TOP 45%). On CelebA, FV-N defenses significantly decrease the attack performance.

However, the data utility after FV-N perturbation is also seriously damaged.

- Feature Vector Rounding (FV-R). On MNIST, Fashion-MNIST, and CIFAR10, the measure metrics (including AvgPD, MSE, and LPIPS) are relatively small with FV-R defense strategies.
  For CelebA dataset, FV-R (INT) significantly decreases the performance of inversion attacks. However, the data utility using FV-R (INT) is seriously damaged. For example, FV-R (INT) defense increases the MSE to 0.248, but it also reduces the DCA by about 9% and reduces the CosS by 0.14. In conclusion, FV-R is not an ideal defense.

TABLE VIII: Defense performance against the targeted attack (*i.e.*, Attack-1). The defenses are evaluated by the feature extractor ResNet18 trained on MNIST. We highlight defenses with good security-utility tradeoffs in **bold**.

Defense Method	AvgPD↑	MSE↑	LPIPS↑	DCA↑	ARI↑
No Defense	0.060	0.028	0.055	99.58%	96.67%
FV-T (TOP 45%)	0.119	0.077	0.174	99.57%	97.22%
FV-T (TOP 60%)	0.102	0.062	0.119	99.57%	97.12%
FV-T (TOP 75%)	0.092	0.051	0.088	99.57%	97.03%
FV-N (2 × STD)	**0.460**	**0.228**	**0.550**	**99.52%**	**96.17%**
FV-N (1 × STD)	**0.467**	**0.241**	**0.650**	**99.57%**	**96.45%**
FV-N (0.5 × STD)	0.080	0.042	0.074	99.58%	96.59%
FV-R (INT)	0.134	0.089	0.156	99.54%	98.07%
FV-R (0.1)	0.071	0.036	0.067	99.57%	97.01%
FV-R (0.01)	0.060	0.027	0.056	99.58%	97.00%

TABLE IX: Defense performance against the targeted attack (*i.e.*, Attack-1). The defenses are evaluated by the feature extractor LeNet-5 trained on Fashion-MNIST. We highlight defenses with good security-utility tradeoffs in **bold**.

Defense Method	AvgPD↑	MSE↑	LPIPS↑	DCA↑	ARI↑
No Defense	0.044	0.007	0.086	89.17%	39.07%
FV-T (TOP 45%)	**0.284**	**0.126**	**0.424**	**87.03%**	**44.40%**
FV-T (TOP 60%)	**0.237**	**0.094**	**0.387**	**87.90%**	**44.79%**
FV-T (TOP 75%)	0.141	0.044	0.356	88.58%	39.17%
FV-N (2 × STD)	0.183	0.042	0.352	72.10%	14.16%
FV-N (1 × STD)	0.176	0.043	0.419	80.54%	39.50%
FV-N (0.5 × STD)	0.158	0.039	0.479	84.72%	38.65%
FV-R (INT)	0.063	0.015	0.142	88.12%	44.42%
FV-R (0.1)	0.047	0.008	0.092	88.85%	39.07%
FV-R (0.01)	0.047	0.008	0.090	88.80%	39.08%

TABLE X: Defense performance against the targeted attack (*i.e.*, Attack-1). The defenses are evaluated by the feature extractor ResNet34 trained on CIFAR10. We highlight defenses with good security-utility tradeoffs in **bold**.

Defense Method	AvgPD↑	MSE↑	LPIPS↑	DCA↑	ARI↑
No Defense	0.372	0.181	0.373	95.38%	87.97%
FV-T (TOP 45%)	0.381	0.192	0.384	95.30%	87.97%
FV-T (TOP 60%)	**0.393**	**0.218**	**0.333**	**95.34%**	**87.97%**
FV-T (TOP 75%)	0.386	0.211	0.342	95.37%	87.97%
FV-N (2 × STD)	0.386	0.200	0.429	94.25%	85.84%
FV-N (1 × STD)	**0.392**	**0.207**	**0.365**	**95.10%**	**86.43%**
FV-N (0.5 × STD)	0.373	0.169	0.524	95.34%	87.97%
FV-R (INT)	0.380	0.196	0.366	95.26%	79.59%
FV-R (0.1)	0.377	0.195	0.333	95.36%	87.99%
FV-R (0.01)	0.371	0.196	0.338	95.37%	87.97%

Explainable Defense Effectiveness. We conduct experiments to explain why the feature vector perturbation method is effective. We use t-SNE [22], [37] to visualize distributions of the GANV-generated fake samples with different defense strategies, without defense, and the original images (*i.e.*, ground truth) on four datasets, as shown in Figure 12. We discover that the distribution of GANV-generated fake data is close to the ground truth if there is no defense. However, if defenses are employed, the distributions of the GANV-generated fake samples will drift away from the ground truth, thereby confirming the correctness of **R1** (see Section VI-A). Additionally, the larger the discrepancy between the GANV-generated fake samples, the better the defense performance. For instance, as shown in Figure 12, the distribution of GANV-generated fake samples with FV-N (2 × STD) defense is far from the ground truth among four datasets. Consequently, FV-N (2 × STD) defense achieves good resistance to the targeted attacks as shown in Table VIII-XI.

TABLE XI: Defense performance against the targeted attack (*i.e.*, Attack-1). The defenses are evaluated by the feature extractor IresNet18 trained on CelebA. We highlight defenses with good security-utility tradeoffs in **bold**.

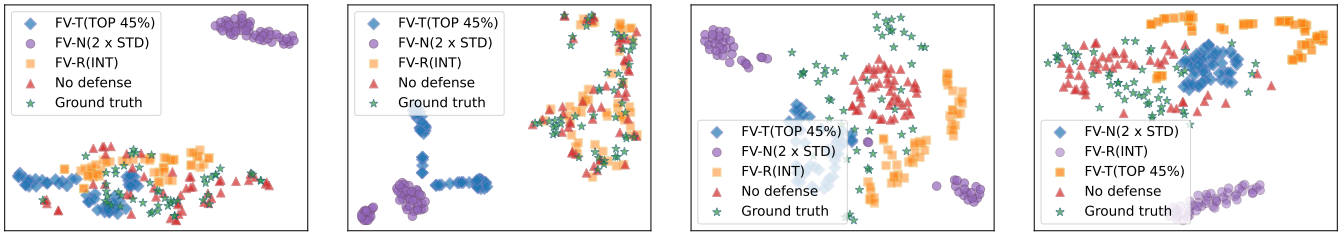
Defense Method	AvgPD↑	MSE↑	LPIPS↑	DCA↑	CosS↑
No Defense	0.113	0.025	0.173	73.54%	1.000
FV-T (TOP 45%)	0.189	0.060	0.440	70.97%	0.844
FV-T (TOP 60%)	**0.179**	**0.052**	**0.407**	**71.69%**	**0.874**
FV-T (TOP 75%)	**0.176**	**0.051**	**0.371**	**72.09%**	**0.899**
FV-N (2 × STD)	0.265	0.110	0.747	30.64%	0.320
FV-N (1 × STD)	0.248	0.097	0.752	36.09%	0.518
FV-N (0.5 × STD)	0.238	0.091	0.747	45.84%	0.736
FV-R (INT)	0.248	0.100	0.617	64.57%	0.860
FV-R (0.1)	0.144	0.037	0.242	72.60%	0.999
FV-R (0.01)	0.134	0.032	0.221	73.56%	1.000

VII. Related Work

In this section, we review the related work on generative neural networks and data reconstruction attacks.

Generative Adversarial Networks. Generative adversarial network (GAN) is first proposed in [19]. Classic GAN usually comprises a generator that produces an image and a discriminator that determines whether a generated image is real or fake. Remarkably, the GANV used in our attacks is different from the classical GAN in two-fold. First, a fixed neural network (*i.e.*, feature extractor) is inserted between the generator and the discrimination in GANV. Second, in the classical GAN, the discriminator is used to distinguish between two images. In contrast, in our GANV, the discriminator is used to distinguish between two extracted feature vectors.

Data Reconstruction Attacks. Data reconstruction attacks aim to reconstruct the original data from various information leaked to the adversary. According to the types of leaked information, such attacks can be divided into four types: gradient-based attacks [7], [18], [23], [43], [49], model-based attacks [8], [9], [21], [47], [48], feature vector-based attacks [16], [30], [34], [35], and other-information-based attacks [6], [32], [38], [45], [46]. This paper belongs to the feature vector-based attacks. Unlike previous feature vector-based attacks that require the availability of an auxiliary dataset, this work mainly focuses on the data-free setting, in which the adversary has no auxiliary dataset.



(a) MNIST (b) Fashion-MNIST (c) CIFAR10 (d) CelebA

Fig. 12: Applying t-SNE to visualize distributions of the GANV-generated fake samples with different defense strategies, without defense, and the original images (*i.e.*, ground truth) on four datasets.

VIII. Conclusion

In this work, we have conducted the first systematic study of FVI attacks against vector databases in the data-free setting. Our key innovation is to develop multiple techniques/strategies (including output-to-input data generation technique, accelerable CBG search strategy, and downstream-classifier-aided generator training strategy) to realize the data-free FVI attacks. In addition, we empirically identify multiple factors that influence the attack performance. Based on our study on 8 types of FVI attacks, it holds that the more background knowledge and capabilities the attacker owns, the higher the attack performance. It should be highlighted that the proposed FVI attack technique in the data-free setting can be directly employed to boost the attack performance of prior FVI attacks in the data-available setting, thereby leading to further impacts on the FVI attack research.

One limitation of this paper is that we only study FVI attacks on image data. However, the proposed data-free FVI attack can be applied to other data modalities, such as text and audio. For text and audio, the FVI attack performance varies depending on multiple factors, including the vector database size, the information entropy contained in the data, etc. For graph data, it is difficult to launch successful data-free FVI attacks because uniform graph embeddings can be produced from heterogeneous graph data (with different numbers of nodes and edges). We do not study these attacks in this paper due to the lack of space. In our future work, we will study FVI attacks on other modalities.

References

[1] Clarifai general image embedding model. https://clarifai.com/clarifai/main/models/general-image-embedding.
[2] Milvus. https://milvus.io/.
[3] Openai api. https://openai.com/blog/openai-api.
[4] Pinecone announces $28m series a for purpose-built database aimed at data scientists. https://tcrn.ch/3GGiVtB.
[5] Weaviate. https://github.com/semi-technologies/weaviate.
[6] Borja Balle, Giovanni Cherubin, and Jamie Hayes. Reconstructing training data with informed adversaries. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1138–1156, 2022.
[7] Franziska Boenisch, Adam Dziedzic, Roei Schuster, Ali Shahin Shamsabadi, Ilia Shumailov, and Nicolas Papernot. When the curious abandon honesty: Federated learning is not private. In *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*, pages 175–199, 2023.
[8] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX security symposium (USENIX security)*, pages 267–284, 2019.
[9] Nicolas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramer, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. In *32nd USENIX Security Symposium (USENIX Security)*, pages 5253–5270, 2023.
[10] David Cash, Joseph Jaeger, Stanislaw Jarecki, Charanjit Jutla, Hugo Krawczyk, Marcel-Cătălin Roşu, and Michael Steiner. Dynamic searchable encryption in very-large databases: Data structures and implementation. *Cryptology ePrint Archive*, 2014.
[11] Hanting Chen, Yunhe Wang, Chang Xu, Zhaohui Yang, Chuanjian Liu, Boxin Shi, Chunjing Xu, Chao Xu, and Qi Tian. Data-free learning of student networks. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)*, pages 3514–3522, 2019.
[12] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the ACM workshop on artificial intelligence and security (AISec)*, pages 15–26, 2017.
[13] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning (ICLR)*, pages 1597–1607, 2020.
[14] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 4690–4699, 2019.
[15] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine (SPM)*, 29(6):141–142, 2012.
[16] Chi Nhan Duong, Thanh-Dat Truong, Khoa Luu, Kha Gia Quach, Hung Bui, and Kaushik Roy. Vec2face: Unveil human faces from their blackbox features in face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (ICCV)*, 2020.
[17] Ionut Cosmin Duta, Li Liu, Fan Zhu, and Ling Shao. Improved residual networks for image and video recognition. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 9415–9422, 2021.
[18] Chong Fu, Xuhong Zhang, Shouling Ji, Jinyin Chen, Jingzheng Wu, Shanqing Guo, Jun Zhou, Alex X Liu, and Ting Wang. Label inference attacks against vertical federated learning. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1397–1414, 2022.
[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems (NeurIPS)*, 2014.
[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 770–778, 2016.
[21] Xinlei He, Jinyuan Jia, Michael Backes, Neil Zhenqiang Gong, and Yang Zhang. Stealing links from graph neural networks. In *30th USENIX Security Symposium (USENIX Security)*, pages 2669–2686, 2021.
[22] Geoffrey E Hinton and Sam Roweis. Stochastic neighbor embedding. *Advances in neural information processing systems (NeurIPS)*, 2002.
[23] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: information leakage from collaborative deep

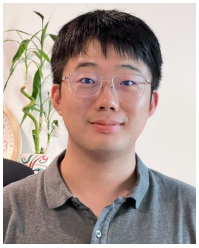
learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security (ACM CCS)*, pages 603–618, 2017.
[24] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.
[25] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2:193–218, 1985.
[26] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.
[27] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, pages 2278–2324, 1998.
[28] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 6738–6746, 2017.
[29] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.
[30] Guangcan Mai, Kai Cao, Pong C Yuen, and Anil K Jain. On the reconstruction of face images from deep face templates. *IEEE transactions on pattern analysis and machine intelligence (TPAMI)*, 41(5):1188–1202, 2018.
[31] Frank Rosenblatt et al. *Principles of neurodynamics: Perceptrons and the theory of brain mechanisms*. 1962.
[32] Ahmed Salem, Apratim Bhattacharya, Michael Backes, Mario Fritz, and Yang Zhang. {Updates-Leak}: Data set inference and reconstruction attacks in online learning. In *29th USENIX security symposium (USENIX Security)*, pages 1291–1308, 2020.
[33] Juergen Schmidhuber. Annotated history of modern ai and deep learning. *arXiv preprint arXiv:2212.11279*, 2022.
[34] Hatef Otroshi Shahreza, Vedrana Krivokuća Hahn, and Sébastien Marcel. Vulnerability of state-of-the-art face recognition models to template inversion attack. *IEEE Transactions on Information Forensics and Security (TIFS)*, 2024.
[35] Hatef Otroshi Shahreza and Sébastien Marcel. Template inversion attack against face recognition systems using 3d face reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19662–19672, 2023.
[36] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
[37] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research (JMLR)*, 2008.
[38] Edward Vendrow and Joshua Vendrow. Realistic face reconstruction from deep embeddings. In *NeurIPS 2021 Workshop Privacy in Machine Learning (NeurIPS)*, 2021.
[39] Grega Vrbančič and Vili Podgorelec. Transfer learning with adaptive fine-tuning. *IEEE Access*, pages 196197–196211, 2020.
[40] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 5265–5274, 2018.
[41] Emily Wenger, Francesca Falzon, Josephine Passananti, Haitao Zheng, and Ben Y Zhao. Assessing privacy risks from feature vector reconstruction attacks. *arXiv preprint arXiv:2202.05760*, 2022.
[42] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.
[43] Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M Alvarez, Jan Kautz, and Pavlo Molchanov. See through gradients: Image batch recovery via gradinversion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 16337–16346, 2021.
[44] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 586–595, 2018.
[45] Wanrong Zhang, Shruti Tople, and Olga Ohrimenko. Leakage of dataset properties in {Multi-Party} machine learning. In *30th USENIX security symposium (USENIX Security 21)*, pages 2687–2704, 2021.
[46] Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song. The secret revealer: Generative model-inversion attacks against deep neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 253–261, 2020.
[47] Zhikun Zhang, Min Chen, Michael Backes, Yun Shen, and Yang Zhang. Inference attacks against graph neural networks. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 4543–4560, 2022.
[48] Junhao Zhou, Yufei Chen, Chao Shen, and Yang Zhang. Property inference attacks against gans. *NDSS 2022*, 2022.
[49] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems (NeurIPS)*, 2019.

Shengyang Qin received the master's degree with the Department of Computer Science and Technology, Chongqing University, Chongqing, China, in 2025. She is currently an solution engineer of the China Telecom Corporation Limited Chongqing B ranch. Her current research focuses on machine learning and cybersecurity.

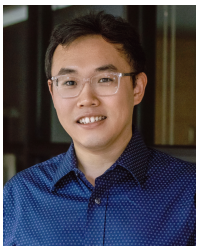
Xinyu Lei (Member, IEEE) received the Ph.D. degree with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, USA, in 2021. He is currently an Assistant Professor with the Department of Computer Science, Michigan Technological University, Houghton, MI, USA. He worked as a Research Assistant with the Texas A&M University at Qatar, Doha, Qatar, in 2013. His current research focuses on machine learning and cybersecurity.

Nankun Mu received the B.S. degree from the School of Big Data Software Engineering, Chongqing University, Chongqing, China, in 2011, and the Ph.D. degree from the College of Computer Science, Chongqing University, Chongqing, China, in 2015. He is currently an associate professor of the College of Computer Science, Chongqing University, Chongqing, China. His research interest includes AI security, data trading, privacy protection and intelligent optimization.

Hongyu Huang (Member, lEEE) received the Ph.D. degree with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2009. He is currently an Associate Professor with the College of Computer Science, Chongqing University, Chongqing, China. His current research focuses on cybersecurity, deep learning, and AI4Science.



Tian Xie is an Assistant Professor in the Department of Computer Science at Utah State University. His research focuses on advancing security and privacy in mobile networks, IoT systems, and cyber-physical infrastructures. His work has been recognized with prestigious awards such as the ACM MobiCom Best Community Paper Runner-Up, IEEE CNS Best Paper, AT&T Security Award, and Google Security Reward.



Xiao Zhang received his Ph.D. degree in Computer Science at Michigan State University, USA, in 2023. He is currently an Assistant Professor with the Department of Computer and Information Science, University of Michigan-Dearborn, Dearborn, MI, usa. His research interests are in the areas of nextgeneration wireless networking, AIoT, HCI, mobile computing and digital twins.