Ginver: Generative Model Inversion Attacks Against Collaborative Inference

Yupeng Yin[1*] Xianglong Zhang[1*] Huanle Zhang[1], Feng Li[1], Yue Yu[2], Xiuzhen Cheng[1], Pengfei Hu[1†]
[1]Shandong University [2]National University of Defense Technology
{yinyp,zxlong22}@mail.sdu.edu.cn,{dtczhang,fli,xzcheng,phu}@sdu.edu.cn,yuyue@nudt.edu.cn

ABSTRACT

Deep Learning (DL) has been widely adopted in almost all domains, from threat recognition to medical diagnosis. Albeit its supreme model accuracy, DL imposes a heavy burden on devices as it incurs overwhelming system overhead to execute DL models, especially on Internet-of-Things (IoT) and edge devices. Collaborative inference is a promising approach to supporting DL models, by which the data owner (the victim) runs the first layers of the model on her local device and then a cloud provider (the adversary) runs the remaining layers of the model. Compared to offloading the entire model to the cloud, the collaborative inference approach is more data privacy-preserving as the owner's model input is not exposed to outsiders. However, we show in this paper that the adversary can restore the victim's model input by exploiting the output of the victim's local model. Our attack is dubbed Ginver[1]: Generative model inversion attacks against collaborative inference. Once trained, Ginver can infer the victim's unseen model inputs without remaking the inversion attack model and thus has the generative capability. We extensively evaluate Ginver under different settings (e.g., white-box and black-box of the victim's local model) and applications (e.g., CIFAR10 and FaceScrub datasets). The experimental results show that Ginver recovers high-quality images from the victims.

CCS CONCEPTS

• **Security and privacy** → *Privacy protections*; • **Networks** → **Network privacy and anonymity**.

KEYWORDS

model inversion attack, collaborative inference, whitebox/blackbox

ACM Reference Format:
Yupeng Yin, Xianglong Zhang, Huanle Zhang, Feng Li, Yue Yu, Xiuzhen Cheng, Pengfei Hu. 2023. Ginver: Generative Model Inversion Attacks Against Collaborative Inference . In *Proceedings of the ACM Web Conference 2023 (WWW '23), May 1–5, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3543507.3583306

*These authors contributed equally to this work.
†Corresponding author.
[1]Our code is available at https://github.com/Ay1798/Ginver/tree/main

WWW '23, May 1–5, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583306

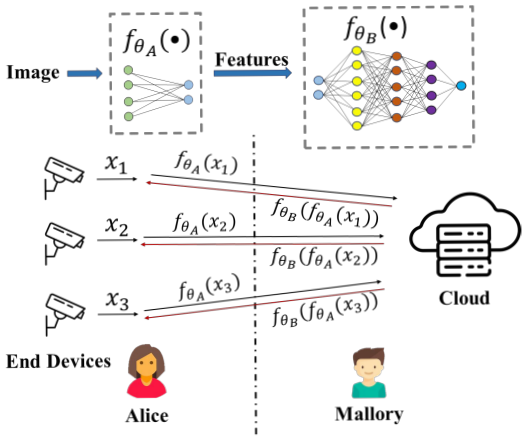


Figure 1: Illustration of collaborative inference. The full model f_θ is divided into f_{θ_A} and f_{θ_B} which runs on the local and the cloud, respectively.

1 INTRODUCTION

The last decade has witnessed the rapid development of Deep Neural Networks (DNN) in many fields, such as image classification [22, 23, 33], object detection [25, 32], and natural language processing [27, 34]. However, DNNs have become increasingly heavy regarding computation overhead and memory footprint. As a result, running DNN models on a local machine is challenging, especially on resource-constraint devices, such as massive Internet-of-Things (IoT) nodes.

To cope with the overwhelming running overheads of DNN models, collaborative model inference pipelines have been proposed [7, 8, 10, 18, 20, 28, 31], in which different devices execute different parts of a DNN model. Figure 1 illustrates a typical collaborative inference system. For example, Alice wants to run a DNN model (denoted by f_θ) on an input (denoted by x), i.e., $f_\theta(x)$, but her device cannot support it. By leveraging the collaborative inference approach, Alice only needs to run the first layers of the model (denoted by f_{θ_A}) and sends the results of $f_{\theta_A}(x)$ to a cloud provider Mallory. Then, Mallory runs the remaining model (denoted by f_{θ_B}) and returns the final inference result to Alice. Obviously, the collaborative inference approach can facilitate the DNN deployment by offloading most system overhead to the cloud.

The collaborative inference approach seems to preserve data privacy as Alice's model input x does not leave her device. However, several work have shown that Alice's model input can be inferred when Mallory is an adversary, i.e., model inversion attack against collaborative inference [13, 14]. Given the intermediate features,

i.e., the output of $f_{\theta_A}(\cdot)$, existing approaches rely on traditional optimization methods to perform the inversion attack, which results in poor image quality especially for a large neural network [24]. Furthermore, with existing approaches, Mallory needs to repeat the optimization process for each input x. It takes minutes for the existing approaches to recover an single image in our experiment. Undoubtedly, existing approaches are time-consuming and laboring, which greatly constrains their applicability in practice.

In this paper, we propose Ginver, a generative model inversion attack against collaborative inference. Ginver learns an inverse model $\mathcal{G}$ with respect to f_{θ_A}, which satisfies $\mathcal{G}\left(f_{\theta_A}(x)\right) = x$. Then, Ginver can directly apply the learnt model $\mathcal{G}$ to the outputs of $f_{\theta_A}(\cdot)$ and recover the model inputs. In addition to the white-box setting that f_{θ_A} is available to Mallory, we also consider the more challenging black-box setting that the model structure and parameters of f_{θ_A} are unknown to Mallory. In the black-box setting, existing work assumes that Mallory holds an auxiliary dataset in the same data distribution as Alice's [13, 14]. This assumption is impractical in general because the reason why Mallory needs to steal Alice's data is that Alice's data are sensitive and protected. In comparison, Ginver does not require any auxiliary dataset to work. Instead, Ginver leverages Natural Evolutionary Strategies (NES) to approximate the reconstruction loss, which turns out to be highly effective to recover the input. To the best of our knowledge, Ginver is the first work that can perform the model inversion attack in the black-box setting without an auxiliary dataset.

Compared to existing work, Ginver is more practical because with Ginver, Mallory (1) does not require knowledge about the data distribution of the victims, (2) efficiently launch the attack within a second without the re-optimazation process, (3) is robust against common defense strategies, and (4) more accurately steals unseen input. We conduct extensive experiments to verify the effectiveness of Ginver and compare Ginver with existing work in model inversion attacks. Specifically, we study the performance of Ginver on CIFAR10 and FaceScrub datasets under the white-box and black-box settings. The experimental results show that Ginver achieves the state-of-the-art performance in the model inversion attack against collaborative inference.

This paper is organized as follows. First, we present related work in Section 2 and then provide preliminaries in Section 3. Afterwards, we clarify Ginver in Section 4 which includes both the white-box attack and the black-box attack. Experiment results and analysis of Ginver performance are given in Section 5. Finally, we conclude this paper in Section 6.

2 RELATED WORK

This section provides related work about general model inversion attacks and the emerging model inversion attacks against collaborative inference.

2.1 General Model Inversion Attack

Both model inversion attack and membership inference attack aim to steal data based on the model information, but they differ in essence. Membership inference attacks against neural networks have been extensively investigated [2, 4, 6, 15, 17, 29]. An adversary can exploit them to confirm whether a piece of data belongs to the training dataset. Different from membership inference attacks, a model inversion attack [9, 19, 30, 36] targets the reconstruction of the model input based on the model output. Many researchers have proposed inspiring work in this topic. For example, Matt et al. [9] gradually optimize the quality of the reconstructed data by narrowing the gap in model output confidence between reconstructed data and input data; Tan et al. [30] formulate the embedding distance in Euclidean space to extract the embedding of the victim's face by querying face verification system scores. Based on the extracted embedding, they design an embedding-GAN to recover face images; To reduce the noise in the reconstructed image and improve the efficiency of the attack, Khosravy et al. [19] reduce the attack search space into face feature vectors instead of general image space; Yang et al. [36] propose a model inversion in an adversarial setting via background knowledge alignment, which introduces a neural network-based generator to reconstruct the input image; and Chen et al. [5] design an inversion-specific GAN that can distill useful knowledge from a public dataset to perform a model inversion attack.

2.2 Model Inversion Attack Against Collaborative Inference System

With the popularity of IoT devices, collaborative inference has attracted more and more attention. However, the intermediate output in the process of collaborative inference becomes a breeding ground for privacy leakage. He et al. [13, 14] propose a model inversion attack against collaborative inference under white-box and black-box settings, where the malicious or curious cloud server can reconstruct the user input from the received intermediate result. However, they must repeat the optimization process to reconstruct each user input. Hence it is not scalable for performing a large-scale inversion attack on massive user inputs. More importantly, they rely on an auxiliary dataset to reconstruct the user input in the black-box setting. Mahendran et al. [24] propose reconstructing the original image with the image representation output of the middle layer, which could be extended to the collaborative inference framework. Our work belongs to this kind of attack but differs from existing work in that our work does not use an auxiliary dataset and is generative to unseen inputs.

3 PRELIMINARY

In this section, we introduce collaborative inference systems and the threat modeling of the model inversion attack against inference systems.

3.1 Collaborative Inference System

Recent years have witnessed the rapid proliferation of DL-enabled IoT devices, e.g., wearable devices and personal voice assistants. The inference of a DL model typically requires heavy computation and storage overhead, which is prohibitively expensive for resource-limited IoT devices. To address this problem, collaborative inference systems are proposed, which have the following advantages.

- **Flexibility.** Deploying advanced DNN models on end devices is challenging, whereas simple models usually do not have enough model accuracy. Hence, DL applications on IoT devices are hindered. The emergence of collaborative

inference systems makes it possible to apply advanced DNN models for inference on IoT devices by deploying the lightweight part of a DNN model on IoT devices and offloading the heavy part of the model to the cloud. Collaborative inference systems are flexible because various model division methods and offloading schemes can be used to accommodate different system scenarios.

- **Privacy.** In collaborative inference, IoT devices only need to transmit the intermediate features from a middle layer of a DNN model rather than the raw data to the cloud to perform the inference. Since the cloud does not have direct access to raw data, the data privacy vulnerability is significantly relieved. In this paper, however, we demonstrate that the model inversion attack can still cause large-scale privacy leakage in collaborative inference under real-world settings.

3.2 Threat Modeling

We adopt the most common two-party collaborative inference system to illustrate our attack. Our approach can be easily extended to multi-party systems as well. The two parties are the data owner and the cloud provider.

- **The data owner (Alice, Bob)**[2] **:** The data owners (e.g., end devices) are victims of privacy leakage. They apply DL inference for better application intelligence. However, due to their limited computation and storage capability, they need the assistance of an external edge or cloud server to achieve collaborative inference. During the collaborative inference, they only execute the computation of the first part of the neural network (i.e., f_{θ_A}).
- **The cloud provider (Mallory):** The cloud provider is the adversary, located at the edge or cloud (depending on the inference task). He has sufficient computational resources to deploy the second part of the model, i.e., f_{θ_B}. Once Mallory receives the intermediate features from Alice, Mallory proceeds with the rest inference and replies to Alice with the inference results. On the other hand, however, Mallory tries to restore Alice's model input corresponding to the intermediate features from Alice. As the same with existing work [13, 14], Mallory is assumed to be able to query Alice's local model f_{θ_A}. That is, Mallory can call f_{θ_A} on any input and receives the result even if the structure and parameters of f_{θ_A} are unknown.

Mallory aims to take advantage of these intermediate features from Alice to build a function mapping $\mathcal{G}$ with respect to f_{θ_A}, which approximates $\mathcal{G}(f_{\theta_A}(x))$ to x. Existing approaches [13, 14] have the following limitations:

- **laborious.** For each input, Mallory needs to re-optimize a new loss function, which is laborious and time-consuming. Therefore, existing approaches cannot perform the model inversion attack against collaborative inference in a large-scale manner.
- **Impractical.** Existing model inversion attacks rely on auxiliary datasets, of which the data distribution is similar to that of Alice's. However, this is impractical in many privacy-sensitive scenarios, e.g., medical imaging and facial recognition. In addition, it is highly likely that Alice does not reserve the inputs due to the privacy concern.

[2]Note that Alice and Bob do not refer to a specific data owner, but generally refer to each data owner in the training and exploiting phases, respectively.

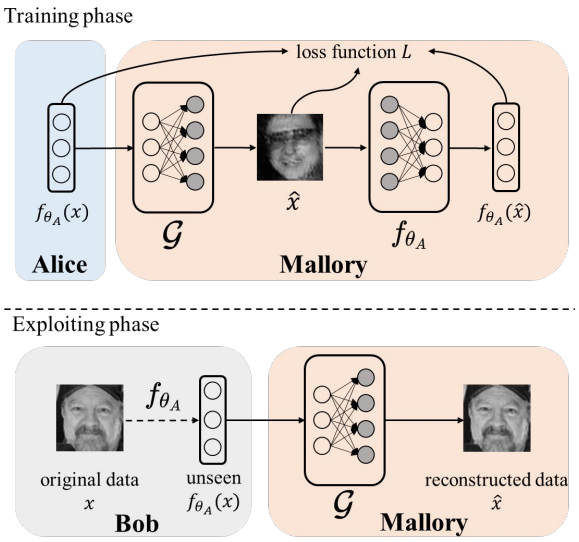


Figure 2: The overview of our proposed attack. In the training phase, Ginver trains the inverse model $\mathcal{G}$ with intermediate features from Alice. After training, Ginver can use $\mathcal{G}$ to quickly reconstruct the original data of Bob from unseen intermediate features.

4 OUR PROPOSED ATTACK

Figure 2 illustrates the attack procedure in Ginver, which includes the training and exploiting phases of the inversion model. Specifically, in the training phase, Mallory collects intermediate features uploaded by Alice and trains model $\mathcal{G}$ with the objective $\mathcal{G}(f_{\theta_A}(x)) = x$; in the exploiting phase, Mallory applies $\mathcal{G}$ to the unseen intermediate features from Bob, which reconstructs Bob's input. In this section, we elaborate on the inversion model training phase since the exploiting phase is straightforward.

4.1 Inversion Model Training Workflow

A middle layer of a neural network retains plentiful semantic characteristics of the input image. Inspired by this, we design a transposed convolutional neural network to recover the user input by upsampling received intermediate features. The objective function of optimizing the model $\mathcal{G}$ can be formulated as follows:

$$\min_{\mathcal{G}} \quad \|\mathcal{G}(f_{\theta_A}(x)) - x\|_2 \tag{1}$$

Equation 1 requires the original input x to calculate the distance. However, Mallory does not have direct access to the original input x; therefore, he cannot directly measure the difference between x and $\hat{x}$ (i.e., $\mathcal{G}(f_{\theta_A}(x))$). To address this problem, we replace the objective function $\|\mathcal{G}(f_{\theta_A}(x)) - x\|_2$ with $\|f_{\theta_A}(\hat{x}) - f_{\theta_A}(x)\|_2$. For simplicity, we use $\mathbb{DIS}(x, \hat{x})$ to represent $\|f_{\theta_A}(\hat{x}) - f_{\theta_A}(x)\|_2$ in the rest of this paper. Therefore, we need to minimize the objective

function:

$$\min_{\mathcal{G}} \quad \mathbb{DIS}(x, \hat{x}) \tag{2}$$

In forward propagation, weak salient features are abandoned due to nonlinear calculation, and thus it is difficult for $\mathbb{DIS}$ to accurately capture the gap between x and $\hat{x}$ when f_{θ_A} is the output of the model's deep layer. Consequently, an inversion model $\mathcal{G}$ trained on Equation 2 fails to produce high-quality vivid images. To improve the quality of the inversion image, we use the Total Variation [24] as the regularization term, which penalizes un-smooth images. Equation 3 gives the definition of Total Variation:

$$TV(x) = \sum_{i,j} \left((x_{i,j+1} - x_{i,j})^2 + (x_{i+1,j} - x_{i,j})^2 \right)^{\frac{\beta}{2}} \tag{3}$$

where $x_{i,j}$ represents the pixel value at position (i, j), and β can be used to adjust the quality of the recovered image. When β is large, the image is blurred but most of the noise can be removed; when β is small, the recovered image has more sharp boundary but also more noise. Based on [24], We set $\beta = 2$, which is effective in our experiments.

In summary, Ginver applies the following loss function in training the inversion model $\mathcal{G}$:

$$L = \mathbb{DIS}(x, \hat{x}) + \lambda \times TV(\hat{x}) \tag{4}$$

where the hyperparameter λ is to balance these two loss terms. The value of λ depends on the intermediate features obtained by the attacker. When the intermediate features come from a layer in the first few layers of Alice's model, a small λ is appropriate; otherwise, a large λ is preferred for a deep layer.

The inversion model $\mathcal{G}$ can be trained by taking the gradients of the loss function, i.e.,

$$\frac{\partial L}{\partial \theta_g} = \frac{\partial \mathbb{DIS}(x, \hat{x})}{\partial \hat{x}} \times \frac{\partial \hat{x}}{\partial \theta_g} + \lambda \times \frac{\partial TV(\hat{x})}{\partial \theta_g} \tag{5}$$

Since the second term $\frac{\partial TV(\hat{x})}{\partial \theta_g}$ is known (Equation 3), the major difference of our solutions between the white-box setting and the black-box setting is how to calculate $\frac{\partial \mathbb{DIS}(x,\hat{x})}{\partial \hat{x}}$.

4.2 White-Box Setting

In the white-box setting, f_{θ_A} is available to Mallory and thus we can directly compute the loss function L. Algorithm 1 in Appendix A sketches the training procedure of the model $\mathcal{G}$. We start by randomly selecting some intermediate feature samples as the training datasets, and use the method proposed in [11] to initialize the parameters of model $\mathcal{G}$, which will make our model more robust during the training process. After that, in each iteration, we take the selected intermediate feature sample as the input of $\mathcal{G}$ to obtain a set of recovered images. Then, we relay the recovered images to f_{θ_A}, which again generates the corresponding intermediate features. We update $\mathcal{G}$ by comparing the original intermediate features with the newly generated intermediate features. At the end of the iteration, we obtain the model $\mathcal{G}$ associated with f_{θ_A}.

4.3 Black-Box Setting

In the black-box setting, Alice's local model f_{θ_A} is unknown to Mallory. Therefore, it is impossible to optimize function L directly. To tackle this problem, we leverage the Natural Evolution Strategies (NES) technique [35] to approximate $\frac{\partial \mathbb{DIS}(x,\hat{x})}{\partial \hat{x}}$.

NES is a heuristic algorithm for performing real-valued "black-box" function optimization. It can optimize the objective function even without any knowledge about the specific parameters and structure of the objective function. Given this property, NES has been widely used in optimization problems [1, 3, 16]. NES does not directly maximize the objective function, but maximizes the expected value of the objective function under the search distribution (usually Gaussian distribution) to estimate the gradient.

We present how NES can be used to estimate gradients and then train the model $\mathcal{G}$ in Algorithm 2 in Appendix A. First, we select n (must be even) random noises $\delta_1, \delta_2, ..., \delta_n$ as the search direction by symmetric sampling, which obey Gaussian distribution and satisfy $\delta_i = \delta_{n+1-i}$ for $i \in \{1, ..., n\}$. After that, we multiply this noise by an appropriate search variance σ and add them to the generated image $\hat{x}$, leading to n new images $\hat{x}_1, \hat{x}_2, ..., \hat{x}_n$ where $\hat{x}_i = \hat{x} + \sigma\delta_i$. Then, we compute the loss values $\mathbb{DIS}(x, \hat{x}_1), \mathbb{DIS}(x, \hat{x}_2), ..., \mathbb{DIS}(x, \hat{x}_n)$ by querying the local model f_{θ_A}. Finally, we use the following expression to estimate $\frac{\partial \mathbb{DIS}(x,\hat{x})}{\partial \hat{x}}$.

$$\frac{\partial \mathbb{DIS}(x, \hat{x})}{\partial \hat{x}} \approx \frac{1}{n\sigma} \sum_{i=1}^{n} \delta_i \mathbb{DIS}(x, \hat{x}_i) \tag{6}$$

Once the estimated value of $\frac{\partial \mathbb{DIS}(x,\hat{x})}{\partial \hat{x}}$ is obtained, we can substitute it into Equation 4 to calculate the gradients of the model $\mathcal{G}$.

5 EVALUATION

In this section, we evaluate our Ginver by extensive experiments. First, we explain our experiment setup in Section 5.1. We then investigate the performance of Ginver in the white-box setting and the black-box setting in Section 5.2 and Section 5.3, respectively. Moreover, we study the robustness of Ginver against common defense strategies in Section 5.4. Finally, we summarize our experimental results in Section 5.5.

5.1 Experimental Setup

5.1.1 Datasets and Model Structures. As illustrated in Table 1, we conduct our experiments on the following two datasets: CIFAR10 [21] and FaceScrub [26].

- **CIFAR10**: The CIFAR10 dataset includes 60, 000 RGB images which are divided into ten categories (e.g., airplane, bird etc.). The resolution of the images is 32×32. By referring to [13], we construct our target full model f_θ consisting of six convolutional layers and two fully connected layers. Its accuracy on the CIFAR10 dataset is 76.3%
- **FaceScrub**: The FaceScrub dataset consists of 106, 863 face images of 530 celebrities; there are about 200 images for each celebrity. These images are converted to 64×64 grayscale ones. Based on this dataset, we train a CNN with seven convolutional layers and two fully connected layers as the target model f_θ. The classification accuracy of the full model f_θ is 85.7% according to our empirical experiments.

We divide each dataset into three subsets: $Train_\theta$, $Train_\mathcal{G}$, and $Test$. $Train_\theta$ is used to train the entire model f_θ. $Train_\mathcal{G}$ is used to

Dataset	# of Images	Type	Resolution	Data Allocation	Target Model	Split Point	Dimension of intermediate features
CIFAR10	60,000	RGB	32×32	83% $train_\theta$ 12% $train_\mathcal{G}$ 5% $Test$	6 conv + 2 fc	- 1st activation layer (ReLU1) - 2nd activation layer (ReLU2) - 4th activation layer (ReLU4) - 6th activation layer (ReLU6)	- (x, 64, 32, 32) - (x, 64, 32, 32) - (x, 128, 16, 16) - (x, 128, 8, 8)
FaceScrub	106,863	Gray	64×64	80% $train_\theta$ 15% $train_\mathcal{G}$ 5% $Test$	7 conv + 2 fc	- 1st activation layer (ReLU1) - 2nd activation layer (ReLU2) - 4th activation layer (ReLU4) - 7th activation layer (ReLU7)	- (x, 128, 64, 64) - (x, 128, 32, 32) - (x, 256, 16, 16) - (x, 1024, 4, 4)

Table 1: Dataset and model information used in the experiments

train the inversion model $\mathcal{G}$, and $Test$ is used to test the validity of the inversion model $\mathcal{G}$. Note that for $Train_\mathcal{G}$, we only obtain the intermediate features from f_{θ_A} on the images to train $\mathcal{G}$. We randomly select 83%, 12% and 5% of all data samples for these three parts in CIFAR10 and 80%, 10% and 10% in FaceScrub.

As shown in Table 1, we adopt four different split point placement schemes to build Alice's local model f_{θ_A} from the entire model f_θ. Specifically, for each model, we place the split point at the 1st, the 2nd, the 4th and the last activation layer, respectively.

Baseline: In the white-box setting, we compare Ginver with the state-of-the-art (SOTA) work [13]. The baseline method is based on optimization for inversion attacks. As will be shown later, we compare Ginver with the baseline method in the white-box attack. Since no existing approach can work in black-box attacks without an auxiliary dataset, we only measure Ginver's performance when it comes to black-box attack.

5.1.2 Metrics. We use the following two metrics to quantify the attack effect, i.e., comparing the similarity/difference between the input image and the recovered image.

- **Structural Similarity Index (SSIM)**. It measures the consistency between two images in three aspects: luminance, contrast, and structure. The value range of SSIM is [0, 1]. A larger SSIM value indicates a higher similarity between two images.
- **Mean Squared Error (MSE)**. It is used to measure the pixel-wise difference between two images. A small MSE value means that two images have higher similarity at the pixel level.

5.1.3 Hyperparameter Setting.

- λ**:** λ is the smoothing coefficient in the loss function (see Equation (4)). We set the value of λ based on the results shown in [24]. In particular, when the split point is located in the shallow layer of a model, we set λ to a very small value (e.g., a positive real number closed to zero), while if the split point is located in deep layers, we increase λ proportionally up to 3.
- n **and** σ**:** These two parameters are the number of samples and search variance in NES, respectively. According to [16], we let to $n = 50$ and $\sigma = 0.001$.

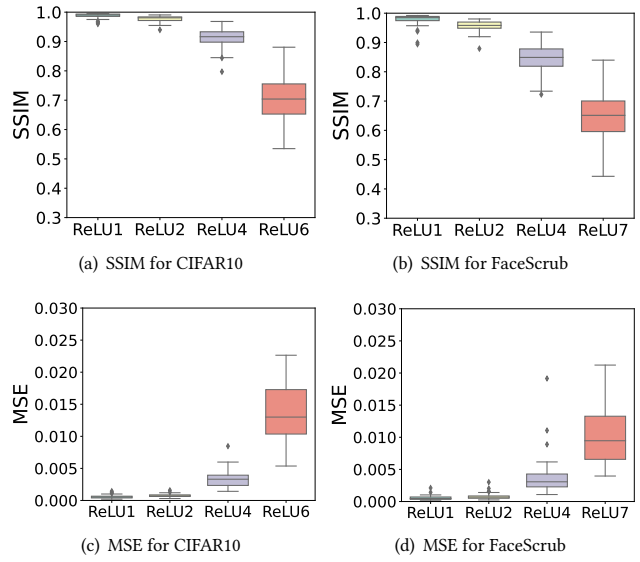
5.1.4 Hardware Equipment. We conduct our experiments on a server running Ubuntu 20.04 LTS. The server is equipped with Intel CPU i9-12900K, 64GB RAM, and Nvidia RTX 3090.

5.2 Attacks in the White-Box Setting

(a) **CIFAR10**

(b) **FaceScrub**

Figure 3: Inversion results in white-box attack. In the first four rows, we illustrate the images recovered by Ginver with four different split point placement schemes. The results obtained by the baseline method where the split point is in the last activation layer are presented in the fifth row. We also give the original images in the last row.



(a) SSIM for CIFAR10 (b) SSIM for FaceScrub

(c) MSE for CIFAR10 (d) MSE for FaceScrub

Figure 4: SSIM and MSE metrics of Ginver in recovering images in the white-box setting for different split points. A higher SSIM and a lower MSE are better.

We first show the inversion results of both Ginver and the baseline method in white-box attack in Figure 3. Specifically, we deploy the split point at four different layers in Ginver and illustrate the corresponding recovered images in the first four rows, respectively. It is demonstrated that, when the split point is placed at deeper layers, there is only a very slight decrease in terms of image quality. Fortunately, even the split point is at the last activation layer, the images recovered by Ginver is still recognizable. Therefore, the performance of Ginver can be guaranteed, regardless of how the split point is placed. In contrast, when we apply the baseline algorithm to the target model split at the last layer, the obtained recovered images are rather obscure.

We also evaluate Ginver by showing the statistical quantities of the SSIM and MSE of the images outputted by Ginver in Figure 4. We observe that for both datasets, Ginver can accurately recover the images with high structural consistency (SSIM greater than 0.9) and little noise (MSE less than 0.003) when the split point is located in the first two activation layers. Even when the split point is placed at a deeper layer (e.g., ReLU4 layer in our case), the performance of Ginver is still guaranteed. For example, when the split point is at the last layer, the medium values of SSIM for the CIFAR10 dataset and the FaceScrub dataset are 0.703 and 0.651, respectively, while the ones of MSE for the two datasets are 0.013 and 0.009, respectively. In addition, we find that there are some abnormal points when the split point is placed at ReLu4 layer. The reason may be that some samples lost more salient features in the process of forward propagation, making them more difficult to be recovered.

The comparison between Ginver and the baseline method in terms of SSIM and MSE is reported in Figure 5. It is apparent that Ginver consistently performs better than the baseline method. In particular, Ginver outperforms baseline significantly when the intermediate features are from a deep layer (e.g., the last activation layer). For instance, the baseline results in an average of MSE 0.04 on FaceScrub dataset, while Ginver reduces the average MSE to only 0.01.

	CIFAR10		FaceScrub	
	Ginver	Baseline	Ginver	Baseline
ReLU1	0.46	77.18	1.02	106.54
ReLU2	0.50	131.76	1.14	402.78
ReLU4	0.52	394.16	1.36	452.64
ReLU6(7)	0.68	445.56	1.37	992.02

Table 2: Time overhead (in seconds) to recover 100 images.

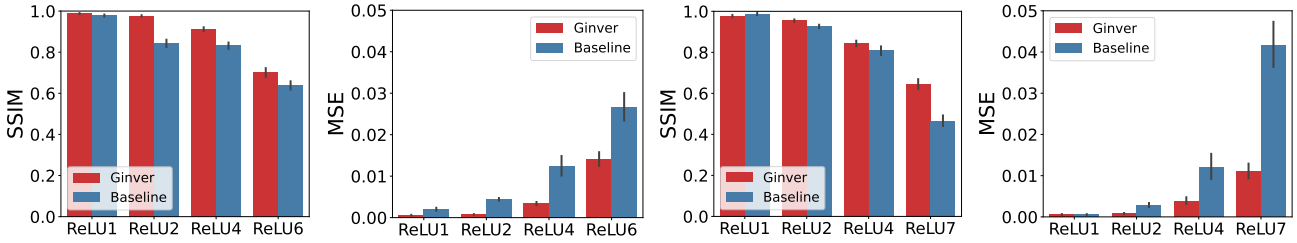
In addition to the better quality of recovered images, Ginver is more efficient for launching a large-scale attack. Table 2 tabulates how long it takes to recover 100 images in Ginver and the baseline, which shows that Ginver reduces the time overhead by more than a hundred times. For instance, Ginver only takes 0.68 seconds to recover 100 images from the last split point on the CIFAR10 dataset, while the baseline requires 445.56 seconds to steal the data. The reason why Ginver is efficient in recovering an image is that once the model $\mathcal{G}$ is well trained by Ginver, the attacker can apply it to the new unseen data directly. In contrast, the baseline must repeat the whole optimization process for each new input.

5.3 Attacks in the Black-Box Setting

Since [13] is not compatible the black-box attack without the auxiliary dataset, we hereby show the performance of Ginver only. The time cost for Ginver to recover the 100 images under the black-box setting are very close to the one in the white-box setting, as Ginver entails only a light-weight inference process under both of the different settings. Therefore, we hereby do not illustrate the details, especially considering the space is limited.

In Figure 6, we illustrate the performance of Ginver by locating the split point at different layers. We also give the original images as ground truth for the purpose of comparison. As expected, we have decreased quality of the recovered images if we split at deeper layers. Specifically, when we split the neural network at the first few layers (e.g., from ReLU1 to ReLU4 for both datasets), there is only a minor decrease in terms of the quality of the recovered images. When the split point is at the last layer (i.e., ReLU6 for CIFAR10 and ReLU7 for FaceScrub), the recovered images are obscure. Typically, however, the split point locates at the first few layers of a complex model in collaborative inference to relieve the local computation and storage cost for resource-constraint devices, hence Ginver is still effective in real-world scenarios.

Figure 7 illustrates the SSIM and MSE of Ginver for recovering images under the black-box setting. As expected, the performance of Ginver degrades as the split point going deeper. Fortunately, the performance still can be ensured until we split the model at the last layer, which consistent with what we have shown in Figure 6. In particular, even when the split point is located at ReLU4 layer, the SSIMs are higher than 0.8 and the MSEs are lower than 0.01. These results indicate that Ginver is of high efficacy to recover images with high fidelity. Also, it is demonstrated again that the performance of Ginver degrades drastically only when the split point is at the last layer, while such a split scheme is rarely adopted in practical collaborative inference systems.



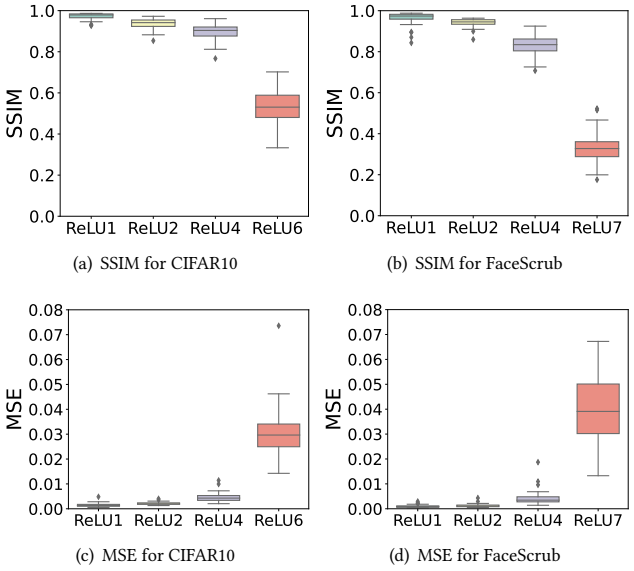
(a) SSIM for CIFAR10 (b) MSE for CIFAR10 (c) SSIM for FaceScrub (d) MSE for FaceScrub

Figure 5: Overall comparison between Ginver and the baseline for the CIFAR10 and FaceScrub datasets in the white-box setting.

(a) **CIFAR10**

(b) **FaceScrub**

Figure 6: Inversion results in black-box attack. The first four rows show the recovered images at different split points, while the last row shows the original images.



(a) SSIM for CIFAR10 (b) SSIM for FaceScrub (c) MSE for CIFAR10 (d) MSE for FaceScrub

Figure 7: SSIM and MSE metrics of Ginver in recovering images in the black-box setting for different split points.

Figure 8: Inversion result from the last activation layer of Resnet-50 in the white-box setting. The first row is the inversion image, and the second row is the original image.

5.4 Potential Defense Strategies

We conduct more experiments to understand two commonly used defense strategies against model inversion attacks.

5.4.1 Defense Strategy 1: A Deep Split Point. Previous results indicate deep split point can defend Ginver to a certain extent in the black-box setting. To investigate whether the white-box setting can benefit from a deep layer, we train a ResNet-50 model [12] based on the FaceScrub dataset. In this experiment, images are resized to 224×224 to fit the model input. We set the split point at the last activation layer, whose output dimension is $(x, 1024, 7, 7)$. As shown in Figure 8, although the recovered images are blurry, they are still recognizable. Hence, we can conclude that the split point in a deep layer cannot defend Ginver effectively under a white-box setting. To some extent, this experimental result also shows that Ginver can be applied to more complex models.

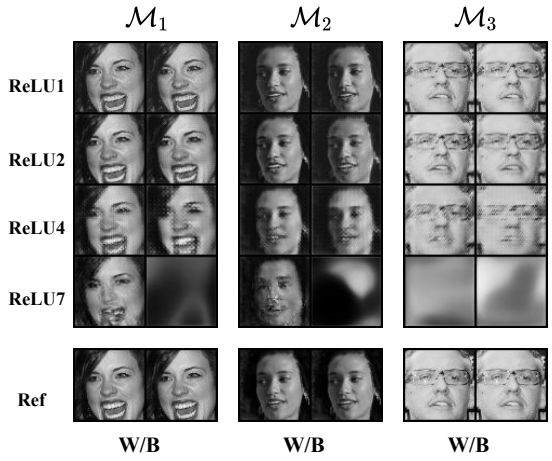


Figure 9: Inversion results with different number of channels. "W" and "B" refers to white-box and black-box, respectively.

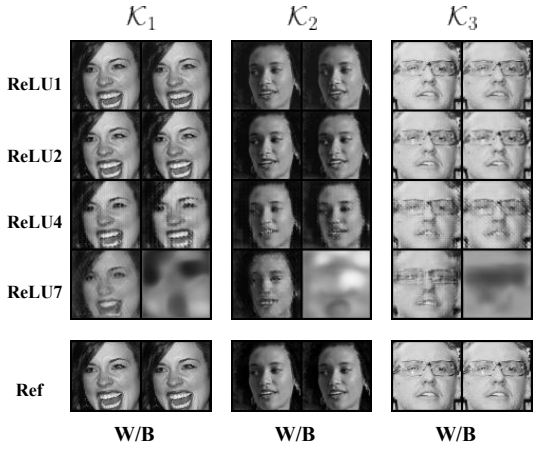


Figure 10: Inversion results with bottleneck structure. "W" and "B" refers to white-box and black-box, respectively.

5.4.2 Defense Strategy 2: Reduced Dimension of Intermediate Features. The reason why the collaborative inference system is vulnerable to Ginver is that the intermediate features contain a large amount of information. Reducing the dimension of intermediate features may mitigate the model inversion attacks to some extent. We conduct experiments under both white-box and black-box settings to verify this idea by reducing the number of channels in each convolutional layer in the full model f_θ. We retrain three classification models $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_3$ with FaceScrub dataset. The number of channels in each convolutional layer of them is $\frac{1}{2}$, $\frac{1}{4}$, and $\frac{1}{8}$ of the original model, respectively. The classification accuracy of these three models is 84.14%, 82.54%, and 78.4%. We adopt the same split scheme as Table 1 and evaluate the performance of Ginver against these new dimension reduced models under both white-box and black-box settings.

As shown in Figure 9, under the white-box setting, when the number of channels in the convolutional layer decreases to a certain level (i.e., $\mathcal{M}_3$), Ginver cannot recover a recognizable recovered image if the split point is placed at the ReLU7 layer. This implies that it is feasible to mitigate model inversion attacks by reducing the dimension of intermediate features in this case. However, this approach also results in considerable performance degradation of DL model. Hence such a defense scheme is impractical for accuracy-oriented tasks. Moreover, we also found that this method does not provide sufficient defense when the split points are in the first few layers, even in the black-box condition.

In addition, we also tested the impact of the bottleneck structure [12] on Ginver. This structure can reduce the dimension of the intermediate value to reduce the transmission overhead while maintaining the accuracy of the model, so it is widely used in collaborative inference system. We use this structure to reduce the dimension of the median value of each split point to $\frac{1}{2}(\mathcal{K}_1)$, $\frac{1}{4}(\mathcal{K}_2)$, and $\frac{1}{8}(\mathcal{K}_3)$ of the original, and the results are shown in Figure 10. Obviously, this structure cannot defend Ginver very well. Compared with the previous defense, it cannot prevent Ginver from recovering recognizable images from the intermediate values of deep split points under white-box conditions, so thus the benefit of bottleneck structure against Ginver is limited.

5.5 Experiment Observations

Through the extensive evaluation, we can draw the following conclusions about Ginver. (1) Ginver is high effective in both the white-box setting and the black-box setting, with relatively high SSIM and low MSE. (2) Ginver is extremely fast to recover an image because Ginver does not need to re-optimize itself for new input. It only takes Ginver about one second to recover 100 images in a small server. (3) The defense strategy of applying a deep split point is less effective for the white-box setting than the black-box setting. However, even for the black-box setting, this defense strategy is impractical as in collaborative inference system it is rare to deploy a heavy model on the local device. (4) The defense strategy of reducing the intermediate features works poorly against our attack as it significantly deteriorate the application model accuracy.

6 CONCLUSION

In this paper, we propose Ginver, a novel generative model inversion attack against the collaborative inference system. Specifically, we design an inversion model to reconstruct the user input based on the intermediate features. Compared with existing schemes, Ginver can perform a large-scale attack without a tedious re-optimization process. Furthermore, we propose an NES-based inversion model training algorithm without an auxiliary dataset for black-box scenarios. We conducted extensive experiments to evaluate the performance of Ginver in both white-box and black-box scenarios. The evaluation results demonstrate Ginver achieves better performance compared with the baseline [13]. Besides, Ginver is robust against typical defense strategies.

ACKNOWLEDGMENTS

This work is supported by the National Key Research and Development Program of China (No. 2021YFB3100400), National Natural Science Foundation of China (Grant No. 62202276, 62232010, 62072278), Shandong Science Fund for Excellent Young Scholars (No. 2022HWYQ-038) and Shandong Provincial Natural Science Foundation (No. ZR2022LZH010).

REFERENCES

[1] Alon Berliner, Guy Rotman, Yossi Adi, Roi Reichart, and Tamir Hazan. 2022. Learning Discrete Structured Variational Auto-Encoder using Natural Evolution Strategies. In The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022. OpenReview.net. https://openreview.net/forum?id=JJCjv4dAbyL

[2] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. 2022. Membership inference attacks from first principles. In 2022 IEEE Symposium on Security and Privacy (SP). IEEE, 1897–1914.

[3] Guangke Chen, Sen Chen, Lingling Fan, Xiaoning Du, Zhe Zhao, Fu Song, and Yang Liu. 2021. Who is Real Bob? Adversarial Attacks on Speaker Recognition Systems. In 42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021. IEEE, 694–711. https://doi.org/10.1109/SP40001.2021.00004

[4] Hanxiao Chen, Hongwei Li, Guishan Dong, Meng Hao, Guowen Xu, Xiaoming Huang, and Zhe Liu. 2022. Practical Membership Inference Attack Against Collaborative Inference in Industrial IoT. IEEE Trans. Ind. Informatics 18, 1 (2022), 477–487. https://doi.org/10.1109/TII.2020.3046648

[5] Si Chen, Mostafa Kahla, Ruoxi Jia, and Guo-Jun Qi. 2021. Knowledge-enriched distributional model inversion attacks. In Proceedings of the IEEE/CVF international conference on computer vision. 16178–16187.

[6] Christopher A Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. 2021. Label-only membership inference attacks. In International conference on machine learning. PMLR, 1964–1974.

[7] Amir Erfan Eshratifar, Mohammad Saeed Abrishami, and Massoud Pedram. 2019. JointDNN: An efficient training and inference engine for intelligent mobile cloud computing services. IEEE Transactions on Mobile Computing 20, 2 (2019), 565–576.

[8] Yihao Fang, Ziyi Jin, and Rong Zheng. 2019. TeamNet: A Collaborative Inference Framework on the Edge. In 39th IEEE International Conference on Distributed Computing Systems, ICDCS 2019, Dallas, TX, USA, July 7-10, 2019. IEEE, 1487–1496. https://doi.org/10.1109/ICDCS.2019.00148

[9] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In Proceedings of the 22nd ACM SIGSAC conference on computer and communications security. 1322–1333.

[10] Johann Hauswald, Thomas Manville, Qi Zheng, Ronald Dreslinski, Chaitali Chakrabarti, and Trevor Mudge. 2014. A hybrid approach to offloading mobile image classification. In 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 8375–8379.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In 2015 IEEE International Conference on Computer Vision (ICCV). 1026–1034. https://doi.org/10.1109/ICCV.2015.123

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. IEEE Computer Society, 770–778. https://doi.org/10.1109/CVPR.2016.90

[13] Z. He, T. Zhang, and R. B. Lee. 2019. Model inversion attacks against collaborative inference. In the 35th Annual Computer Security Applications Conference.

[14] Zecheng He, Tianwei Zhang, and Ruby B. Lee. 2021. Attacking and Protecting Data Privacy in Edge-Cloud Collaborative Inference Systems. IEEE Internet Things J. 8, 12 (2021), 9706–9716. https://doi.org/10.1109/JIOT.2020.3022358

[15] Hongsheng Hu, Zoran Salcic, Gillian Dobbie, Jinjun Chen, Lichao Sun, and Xuyun Zhang. 2022. Membership Inference via Backdooring. In Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022, Luc De Raedt (Ed.). ijcai.org, 3832–3838. https://doi.org/10.24963/ijcai.2022/532

[16] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. 2018. Black-box Adversarial Attacks with Limited Queries and Information. In Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research, Vol. 80), Jennifer G. Dy and Andreas Krause (Eds.). PMLR, 2142–2151. http://proceedings.mlr.press/v80/ilyas18a.html

[17] Jinyuan Jia, Ahmed Salem, Michael Backes, Yang Zhang, and Neil Zhenqiang Gong. 2019. Memguard: Defending against black-box membership inference attacks via adversarial examples. In Proceedings of the 2019 ACM SIGSAC conference on computer and communications security. 259–274.

[18] Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. 2017. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. ACM SIGARCH Computer Architecture News 45, 1 (2017), 615–629.

[19] Mahdi Khosravy, Kazuaki Nakamura, Yuki Hirose, Naoko Nitta, and Noboru Babaguchi. 2022. Model Inversion Attack by Integration of Deep Generative Models: Privacy-Sensitive Face Generation from a Face Recognition System. IEEE Transactions on Information Forensics and Security 17 (2022), 357–372.

[20] Jong Hwan Ko, Taesik Na, Mohammad Faisal Amir, and Saibal Mukhopadhyay. 2018. Edge-host partitioning of deep neural networks with feature space encoding for resource-constrained internet-of-things platforms. In 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS). IEEE, 1–6.

[21] A. Krizhevsky. 2012. Learning Multiple Layers of Features from Tiny Images. (2012).

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States, Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger (Eds.). 1106–1114. https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

[23] Jiawei Ma, Hanchen Xie, Guangxing Han, Shih-Fu Chang, Aram Galstyan, and Wael Abd-Almageed. 2021. Partner-assisted learning for few-shot image classification. In Proceedings of the IEEE/CVF International Conference on Computer Vision. 10573–10582.

[24] Aravindh Mahendran and Andrea Vedaldi. 2015. Understanding deep image representations by inverting them. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015. IEEE Computer Society, 5188–5196. https://doi.org/10.1109/CVPR.2015.7299155

[25] Ishan Misra, Rohit Girdhar, and Armand Joulin. 2021. An end-to-end transformer model for 3d object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision. 2906–2917.

[26] H. W. Ng and S. Winkler. 2015. A data-driven approach to cleaning large face datasets. In IEEE International Conference on Image Processing.

[27] Ke Ning, Lingxi Xie, Jianzhuang Liu, Fei Wu, and Qi Tian. 2021. Interaction-integrated network for natural language moment localization. IEEE Transactions on Image Processing 30 (2021), 2538–2548.

[28] Daniele Jahier Pagliari, Roberta Chiaro, Enrico Macii, and Massimo Poncino. 2021. CRIME: Input-Dependent Collaborative Inference for Recurrent Neural Networks. IEEE Trans. Computers 70, 10 (2021), 1626–1639. https://doi.org/10.1109/TC.2020.3021199

[29] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In 2017 IEEE symposium on security and privacy (SP). IEEE, 3–18.

[30] Mingtian Tan, Zhe Zhou, and Zhou Li. 2021. The Many-faced God: Attacking Face Verification System with Embedding and Image Recovery. In Annual Computer Security Applications Conference. 17–30.

[31] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. 2017. Distributed deep neural networks over the cloud, the edge and end devices. In 2017 IEEE 37th international conference on distributed computing systems (ICDCS). IEEE, 328–339.

[32] Jianfeng Wang, Lin Song, Zeming Li, Hongbin Sun, Jian Sun, and Nanning Zheng. 2021. End-to-end object detection with fully convolutional network. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 15849–15858.

[33] Peng Wang, Kai Han, Xiu-Shen Wei, Lei Zhang, and Lei Wang. 2021. Contrastive learning based hybrid networks for long-tailed image classification. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 943–952.

[34] Xiao Wang, Xiujun Shu, Zhipeng Zhang, Bo Jiang, Yaowei Wang, Yonghong Tian, and Feng Wu. 2021. Towards more flexible and accurate object tracking with natural language: Algorithms and benchmark. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 13763–13773.

[35] Daan Wierstra, Tom Schaul, Jan Peters, and Juergen Schmidhuber. 2008. Natural Evolution Strategies. In 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence). 3381–3387. https://doi.org/10.1109/CEC.2008.4631255

[36] Ziqi Yang, Jiyi Zhang, Ee-Chien Chang, and Zhenkai Liang. 2019. Neural network inversion in adversarial setting via background knowledge alignment. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. 225–240.

A ALGORITHM

Algorithm 1 sketches the training procedure of the inversion model $\mathcal{G}$ under the white-box setting, while Algorithm 2 shows how NES can be used to estimate gradients and then train the inversion model $\mathcal{G}$ under the black-box setting.

Algorithm 1 Ginver training in the white-box setting

function TRAIN($f_{\theta_A}, \mathcal{F}, \lambda, \epsilon, T, k$)
  /*f_{θ_A} is the victim's model*/
  /*$\mathcal{F}$ is a set of intermediate features*/
  /*λ is the equilibrium coefficient in Equation 4*/
  /*ϵ is the learning rate*/
  /*T is the number of iterations*/
  /*k is BatchSize*/
  /*$\theta_{\mathcal{G}}$ represent the parameter of $\mathcal{G}$*/

  initialize $\mathcal{G}$, $t \leftarrow 0$
  while t < T **do**
    Randomly sample $f_{\theta_A}(x_1), f_{\theta_A}(x_2), \ldots, f_{\theta_A}(x_k)$ from $\mathcal{F}$
    $L = \frac{1}{k} \sum_{i=1}^{k} \| f_{\theta_A}\left(\mathcal{G}\left(f_{\theta_A}(x_i)\right)\right) - f_{\theta_A}(x_i) \|_2$
    $L = L + \frac{1}{k} \sum_{i=1}^{k} TV\left(\mathcal{G}\left(f_{\theta_A}(x_i)\right)\right)$
    $\theta_{\mathcal{G}} = \theta_{\mathcal{G}} - \epsilon * \frac{\partial L}{\partial \theta_{\mathcal{G}}}$
    $t = t + 1$
  end while
return $\mathcal{G}$
end function

Algorithm 2 Ginver training in the black-box setting

function TRAIN($\mathcal{F}, \lambda, \epsilon, T, k, n, \sigma$)
  /*$\mathcal{F}$ is a set of intermediate features*/
  /*λ is the equilibrium coefficient in Equation 4*/
  /*ϵ is the learning rate*/
  /*T is the number of iterations*/
  /*k is BatchSize*/
  /*n is the number of samples*/
  /*σ is search variance*/
  /*$\theta_{\mathcal{G}}$ represent the parameter of $\mathcal{G}$*/

  initialize $\mathcal{G}$, $t \leftarrow 0$
  while t < T **do**
    Randomly sample $f_{\theta_A}(x_1), f_{\theta_A}(x_2), \ldots, f_{\theta_A}(x_k)$ from $\mathcal{F}$ and call it $\mathcal{I}$
    $\hat{x} \leftarrow \mathcal{G}(\mathcal{I})$
    $grad = 0$
    for $i \in \{1, 2, ..., \lfloor \frac{n}{2} \rfloor\}$ **do**
      $\delta_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
      $\delta_{n+1-i} = -\delta_i$
      $\hat{x}_1 = \hat{x} + \sigma\delta_i$
      $\hat{x}_2 = \hat{x} + \sigma\delta_{n+1-i}$
      $grad = grad + \delta_i \mathbb{DIS}(x, \hat{x}_1)$
      $grad = grad + \delta_{n+1-i} \mathbb{DIS}(x, \hat{x}_2)$
    end for
    $grad = \frac{1}{n\sigma} * grad$
    $\frac{\partial L}{\partial \theta_{\mathcal{G}}} = grad * \frac{\partial \hat{x}}{\partial \theta_{\mathcal{G}}} + \lambda * \frac{\partial TV(\hat{x})}{\partial \theta_{\mathcal{G}}}$
    $\theta_{\mathcal{G}} = \theta_{\mathcal{G}} - \epsilon * \frac{\partial L}{\partial \theta_{\mathcal{G}}}$
    $t = t + 1$
  end while
return $\mathcal{G}$
end function