Exploiting Type I Adversarial Examples to Hide Data Information: A New Privacy-Preserving Approach

Song Gao, Xiaoxuan Wang, Bingbing Song, Renyang Liu, Shaowen Yao, Wei Zhou, *Member, IEEE*, and Shui Yu, *Fellow, IEEE*

***Abstract*—Deep neural networks (DNNs) are sensitive to adversarial examples which are generated by corrupting benign examples with imperceptible perturbations, or have significant changes but can still achieve original prediction results. The latter case is termed as the Type I adversarial example which, however, has limited attention in the literature. In this paper, we introduce two methods, termed HRG and GAG, to generate Type I adversarial examples and attempt to apply them to the privacy-preserving Machine Learning as a Service (MLaaS). Existing methods for the privacy-preserving MLaaS are mostly based on cryptographic techniques, which often incur additional communication and computation overhead, while using Type I adversarial examples to hide users' privacy data is a brand-new exploration. Specifically, HRG utilizes the high-level representations of DNNs to guide generators, and GAG leverages the generative adversarial network to transform original images. Our solution does not involve any model modifications and allows DNNs to run directly on transformed data, thus arousing no additional communication and computation overhead. Extensive experiments on MNIST, CIFAR-10, and ImageNet show that HRG can perfectly hide images into noise and achieve similar accuracy as the original accuracy, and GAG can generate natural images that are completely different from the original images with a small loss of accuracy.**

***Index Terms*—Type I adversarial examples, deep neural networks, privacy-preserving MLaaS.**

Nomenclature

x	Original (unmodified, benign, clean) input example.
y	The ground truth label of x in the classification task, $y = 1, 2, \ldots, C$, where C is the total number of categories.
r	Adversarial perturbation.
x'	Adversarial example.
$\|\| \cdot \|\|_p$	l_p norm.
$f(\cdot)$	A well-trained DNN model.
θ	Parameters of f.
$H(\cdot)$	A well-trained DNN model without the final Softmax layer.
$G(\cdot)$	A generator. G_{enc} is the encoder of G that projects x into the latent representation z, and G_{dec} is the decoder of G that reconstructs x based on z. G_ω means a generator with parameters ω.
$D_\phi(\cdot)$	A discriminator with parameters ϕ.
$Clip_\varepsilon(\cdot)$	Per-pixel clipping. The clipped results will be in ε-neighborhood of 0.
$KL[P\|\|Q]$	Kullback-Leibler divergence. P is a true distribution and Q is a approximate distribution.
ξ	Random Gaussian noise.

Manuscript received 28 March 2023; revised 20 August 2023 and 22 October 2023; accepted 12 December 2023. Date of publication 4 March 2024; date of current version 27 May 2024. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62101480 and Grant 62162067, in part by the Yunnan Province expert workstations under Grant 202305AF150078, and in part by the Research and Application of Object Detection based on Artificial Intelligence. *(Corresponding author: Wei Zhou.)*

Song Gao, Bingbing Song, Renyang Liu, Shaowen Yao, and Wei Zhou are with the Engineering Research Center of Cyberspace and the National Pilot School of Software, Yunnan University, Kunming 650504, China (e-mail: gaos@ynu.edu.cn; songbingbing@mail.ynu.edu.cn; liurenyang@mail.ynu.edu.cn; yaosw@ynu.edu.cn; zwei@ynu.edu.cn).

Xiaoxuan Wang is with the School of Information Science and Technology, Yunnan Normal University, Kunming 650504, China (e-mail: wangxiaoxuan1037@163.com).

Shui Yu is with the School of Computer Science, University of Technology Sydney, Sydney NSW 2007, Australia (e-mail: shui.yu@uts.edu.au).

Recommended for acceptance by Prof. J. Catalão.

Digital Object Identifier 10.1109/TETCI.2024.3367812

I. Introduction

DEEP neural networks (DNNs) have achieved remarkable performance and empowered a wide range of applications in various domains [1], [2]. However, DNNs have been demonstrated to be fragile to adversarial examples, that is, adding crafted subtle perturbations to benign examples can easily fool a well-trained DNN [3], [4]. Adversarial examples pose a significant threat to DNNs in security-critical applications. For instance, adversarial traffic signs could mislead autonomous driving systems and threaten the safety of traffic participants [5]. Many studies have been proposed to design and resist adversarial examples [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], which, however, mainly focus on the Type II adversarial example. There is another type of adversarial example, the Type I adversarial example, which has received limited attention. Type I adversarial examples have noticeable transformations but can still achieve original predictions from the target model. Type I adversarial examples also threaten DNNs in that attackers could transform illegal images or videos into Type I adversarial versions to avoid detection by some deep learning detectors.

Adversarial examples are not always bad, some researches [15], [16] show that they can be used to study the weaknesses of DNNs, and can augment training datasets to improve the generalization and robustness of DNNs. In this paper, we explore a new application for Type I adversarial examples, applying them to the privacy-preserving Machine Learning as

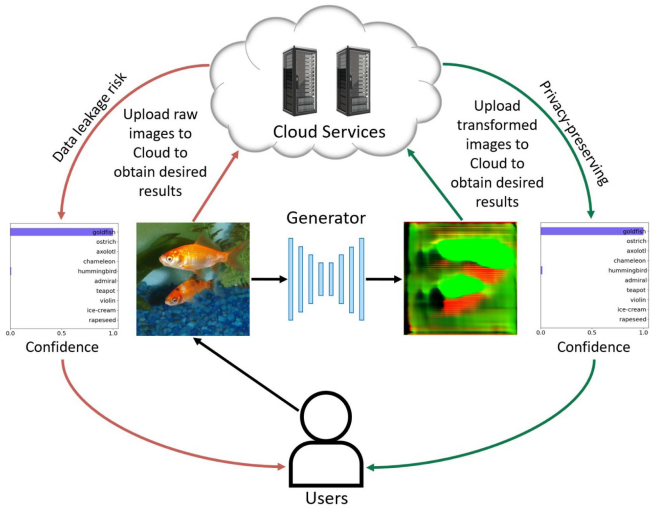


Fig. 1. Illustration of applying the Type I adversarial example to the privacy-preserving MLaaS. Left: Users send their original images to the Cloud to achieve desired results, which exposes the potential risk of data leakage. Right: Users send the Type I adversarial version of their original data to the Cloud without revealing data information.

a Service (MLaaS). MLaaS aims to provide machine learning services on the Cloud, and users can take advantage of cloud services without having to worry about data gathering, model training, and service maintenance. Nevertheless, using cloud services generally requires sharing data with service providers, which is often privacy sensitive and may create security concerns. Existing solutions to this issue mainly adopt cryptographic techniques, such as Secure Multi-Party Computation (SMC) [19], [20], [21], Trusted Computing Base (TCB) [22], [23], and Homomorphic Encryption (HE) [24], [25], [26]. Using Type I adversarial examples to prevent DNNs from directly accessing raw data is a whole new way in which users can achieve desired results by sending the Type I adversarial version of their private data to the Cloud without revealing data information (see Fig. 1).

To achieve our purpose, we propose two strategies, high-level representation guiding (HRG) and generative adversarial guiding (GAG), to generate Type I adversarial examples. HRG sets the loss function as the difference between high-level representations of DNNs induced by raw and transformed examples to guide the training of generators. HRG is very simple and efficient, but it has no control over the generated Type I adversarial examples, usually hiding data information into noise. To obtain natural transformed examples, GAG first takes advantage of the auxiliary classifier generative adversarial network (ACGAN) [27] to transform examples from one class to another class, and then adopts a strategy similar to HRG to produce subtle perturbations to the generated examples to obtain original predictions. Our solution is particularly simple and does not require any operations other than transforming private data with generators. Therefore, there are no modifications to DNNs, nor is there additional computation and communication overhead in the inference process.

In summary, our contributions are as follows:

- We explore a new way for the privacy-preserving MLaaS that employs the Type I adversarial example to conceal the original information of input data. Our solution is efficient and does not incur additional computation and communication overhead in the inference phase. Users simply send the Type I adversarial version of their private data to the Cloud to get desired results, avoiding the risk of data leakage.
- We propose two approaches to generate Type I adversarial examples, one exploits the high-level representation guiding to guide generators to hide raw data into noise, and the other leverages the generative adversarial network to generate natural examples.
- Extensive experiments on three real datasets verify that our approaches can effectively hide data information while achieving reasonable classification accuracy. Meanwhile, due to our methods do not increase the extra calculation of DNNs, they can be applied to large-scale images.

The rest of this work is organized as follows: The related work is shown in Section II, and the proposed approaches are presented in Section III. The experimental results and analyses are shown in Section IV. Finally, Section V shows conclusions.

II. Related Work

In this section, we briefly describe different types of adversarial examples and review existing studies on the privacy-preserving MLaaS.

A. Adversarial Examples

Adversarial examples mainly include two types: the Type I adversarial example and the Type II adversarial example. Methods for generating Type II adversarial examples aim to minimize the difference between the benign example and the adversarial example while misleading DNNs to produce wrong results:

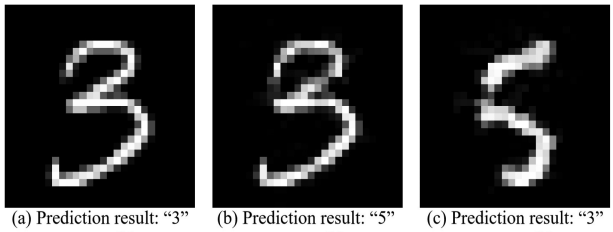
$$\min \ ||r||_p \quad s.t. \quad f(x+r,\theta) \neq y. \tag{1}$$

And the objective of producing Type I adversarial samples can be described as:

$$\begin{aligned} & x' = G(x) \\ \text{subject to} \quad & f_1(x) = f_1(x') \\ & f_2(x) \neq f_2(x'), \end{aligned} \tag{2}$$

where f_1 can be a DNN-based classifier, and f_2 can be a group of human annotators. Fig. 2 shows an intuitive explanation, we can see that the Type II adversarial image is still "3" but is identified as "5", and the Type I adversarial image is really changed to "5" but the neural network still thinks of it as "3".

Many advanced approaches have been presented to generate Type II adversarial examples [3], [4], [7], [8], [9], [10], [11], [12], [13]. For example, Goodfellow et al. [4] presented FGSM that performs a one-step update along the direction of the gradient at each pixel to produce adversarial images. Dong et al. [9] introduced MIM, in which they added a momentum term in the iteration process to generate adversarial images. Carlini and



(a) Prediction result: "3" 99.9% confidence (b) Prediction result: "5" 85.6% confidence (c) Prediction result: "3" 87.7% confidence

Fig. 2. Image and its adversarial versions. (a) Is an original image, (b) is (a)'s Type II adversarial version, and (c) is (a)'s Type I adversarial version. The subtitle of each image is the prediction result and confidence. The original image can be correctly identified as "3" with 99.9% confidence, its Type II adversarial version is still "3" but is recognized as "5" with 85.6% confidence, and its Type I adversarial version is really transformed to "5" but is still identified as "3" with 87.7% confidence.

Wagner [10] treated the task of producing adversarial samples as an optimization problem, and transformed adversarial samples into the $arg tanh$ space for the flexible use of optimization solvers. Zhao et al. [11] introduced Natural GAN based on WGAN [28] to generate natural adversarial examples.

Unlike the Type II adversarial example, the Type I adversarial example has received very limited attention. Tang et al. [29] introduced a supervised extension of the original variational autoencoder (VAE) [30], named supervised variational autoencoder (SVAE), to generate Type I adversarial examples. Similarly, He et al. [31] designed a supervised extension of the original generative adversarial network (GAN) [32] to generate Type I adversarial examples. These two methods integrate the image transformation task and the original prediction task into a unified training process, which is unstable and computationally expensive. In this study, we propose two methods, termed HRG and GAG, to generate Type I adversarial examples. HRG simply transforms original images into noise. And GAG treats the image transformation task and the original prediction task as separate subtasks, which simplifies the training difficulty and reduces the training cost.

B. Privacy-Preserving MLaaS

With the rapid development of Big Data, cloud computing and mobile internet, the scope of personal confidential information has become extensive and vague. Data such as health data [33], [34] and traffic sensor data [35] may pose a threat to personal privacy. The purpose of privacy-preserving MLaaS is to allow users to use online machine learning services without worrying about security issues. Existing methods for the privacy-preserving MLaaS can be roughly divided into three categories: Secure Multi-Party Computation (SMC), Trusted Computing Base (TCB), and Homomorphic Encryption (HE). SMC-based methods such as SecureML [19], DeepSecure [20] and EzPC [21] utilize two-party computation techniques to design scalable and low-latency systems for secure neural network inference. However, SMC-based methods require a large amount of communication between participating parties. TCB-based methods [22], [23] provide hardware-based primitives that enable users to execute their private data in shielded execution environments. Unfortunately, it has been demonstrated that several attacks can allow unprivileged programs to extract content from memory that only privileged programs can access [36]. HE-based methods focus on applying encrypted data to machine learning models. Graepel et al. [37] demonstrated that it is possible to implement confidential machine learning based on the leveled homomorphic encryption scheme. Gilad-Bachrach et al. [24] converted the floating-point numbers in neural networks to integers with appropriate scaling, and replaced pooling operations and activation functions using polynomial approximations. Similarly, Hesamifard et al. [25] proposed CryptoDL that performs both training and inference on encrypted data. Takabi et al. [26] improved the efficiency of cryptographic neural networks by using optimization techniques and efficient GPU-based implementations. However, HE-based methods need to modify DNNs and bring huge computation overhead, which makes their performance far from satisfactory in practical applications. In this work, we introduce a brand-new strategy that takes advantage of Type I adversarial examples to mask data information. In this way, no modifications to DNNs, and no additional computation and communication overhead are incurred during the inference process.

III. Methodology

The basic idea behind our solution can be described as utilizing Type I adversarial examples for privacy preservation. We adopt the autoencoder architecture as generators and propose two methods, high-level representation guiding (HRG) and generative adversarial guiding (GAG), to train generators. An autoencoder consists of an encoder and a decoder. The encoder G_{enc} projects x into the latent representation z, i.e., $z = G_{enc}(x)$, then the decoder G_{dec} reconstructs x based on z, i.e., $x_r = G_{dec}(z + \xi)$.

A. High-Level Representation Guiding

For the generated Type I adversarial examples, they should be able to effectively hide data information and obtain good prediction accuracy. To achieve this goal, we propose high-level representation guiding (HRG) similar to [38] to guide the training of generators. The specific embodiment corresponds to the high-level representation pairing loss. We try three schemes to calculate the high-level representation pairing loss, i.e., Log-Likelihood Loss, L_2 Norm, and Kullback-Leibler (KL)-Divergence. Fig. 3 shows the illustrations of the three schemes.

1) Log-Likelihood Loss: Log-Likelihood Loss is one of the most widely used loss functions in deep learning classification tasks, and is usually used in conjunction with the Softmax activation function, by the following formula:

$$Loss_L = -\sum_{i=1}^{C} y_i ln\hat{y}_i, \tag{3}$$

here, y_i denotes the value of category i in the true label, and $\hat{y}_i$ is the value of category i in the predicted label. One-hot encoding

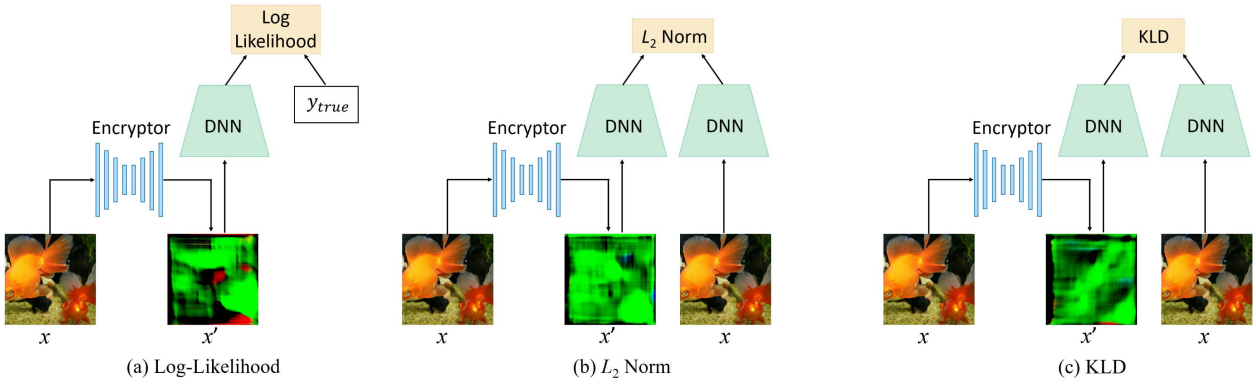


Fig. 3. Three training schemes for HRG. x is an original image, and x' stands for its Type I adversarial version. DNN is the model deployed on the Cloud. The parameters of the DNN are shared and fixed. (a): Using Log-Likelihood Loss to train the generator. (b): L_2 norm is leveraged as the loss function. (c): KL-Divergence is adopted as the loss function.

is usually utilized to encode data's true labels, that is, the value of the correct category in a label is 1, and the other values are 0. Thus, $Loss_L$ can also be written as:

$$Loss_L = -ln\hat{y}_t, \tag{4}$$

where $\hat{y}_t$ is the value of the true category in the predicted label.

2) L_2 Norm: L_2 Norm measures the gap between the outputs activated by the original image and the generated Type I adversarial image. As shown in Fig. 3(b), for an image x, it is first fed into a generator to obtain its Type I adversarial image x', then x and x' are fed into the target DNN to obtain their high-level representations, and finally, the distance between their high-level representations is calculated by the following formula:

$$Loss_{L_2} = ||H(x') - H(x)||_2. \tag{5}$$

3) KL-Divergence: KL-Divergence (KLD) is used to measure the difference between two probability distributions. Assuming $P(Z)$ is a true distribution and $Q(Z)$ is the approximate distribution used to fit $P(Z)$, the KLD is expressed as:

$$KL[P(Z)||Q(Z)] = \sum_{z\in Z}\left[P(z)\log\frac{P(z)}{Q(z)}\right]. \tag{6}$$

In this study, $H(x)$ is the true distribution, and $H(x')$ is the approximate distribution used to fit $H(x)$ (see Fig. 3(c)). Therefore, KLD is formulated as:

$$Loss_{KL} = KL[H(x)||H(x')]. \tag{7}$$

HRG is very simple and converges easily in the training process, but it has no control over the generated Type I adversarial examples and can only hide data information into noise.

B. Generative Adversarial Guiding

The proposed generative adversarial guiding (GAG) consists of two generators G^t and G^n, a discriminator D, and a target DNN f that is named the function model. As shown in Fig. 4, the training process of GAG consists of two steps: training G^t and training G^n. G^t and D form a generative adversarial network that optimizes the transformation of images from the original space to the target space. And the transformation should not be random, but from one class to another in the same distribution. Therefore, we add the label information of images to the generative process, and build the training loss based on ACGAN:

$$\begin{aligned}(\hat{\omega},\hat{\phi})_{G^t,D} &= \underset{\omega}{argmin}\,\underset{\phi}{argmax}\,\widehat{D}_\phi(q(x_t)||p(x_t|x)) \\ &= \mathbb{E}_{x_t\sim q(x_t)}\log\left[D_\phi(x_t)\right] \\ &\quad + \mathbb{E}_{x\sim p(x)}\log\left[1-D_\phi(G^t_\omega(x))\right] \\ &\quad + \lambda Loss_c,\end{aligned} \tag{8}$$

here, x_t is the target example of x, $q(x_t)$ is the data distribution of x_t, $p(x)$ is the data distribution of x, λ denotes a hyper-parameter, and $Loss_c$ is the classification loss that is defined as:

$$\begin{aligned}Loss_c &= \mathbb{E}[\log P(c=y|x_t)] \\ &\quad + \mathbb{E}[\log P(c=y|G^t_\omega(x))].\end{aligned} \tag{9}$$

In the training process, we randomly sample x with the label y from the entire dataset, and x_t is randomly sampled from the class y_t which can be calculated using $(y+1)\%C$, where $\%$ means taking the remainder. Training G^t is considered the first step of GAG, which ensures that an original image can be transformed into a completely different natural image. Next, we fix the parameters of G^t and train G^n. G^n is responsible for generating imperceptible perturbations to the transformed image so that an original image and its transformed image have the same outputs on the target model. We adopt a strategy similar to HRG to train G^n, i.e., using L_2 norm to measure the difference between high-level representations of original images and their Type I versions, which is demonstrated to be more effective than Log-Likelihood Loss and KL-Divergence in experiments. However, HRG will create unexpected perturbations that destroy the naturalness of transformed images. To limit the generated perturbations, we clip them before they are added to transformed images. Mathematically, the loss function for the training of

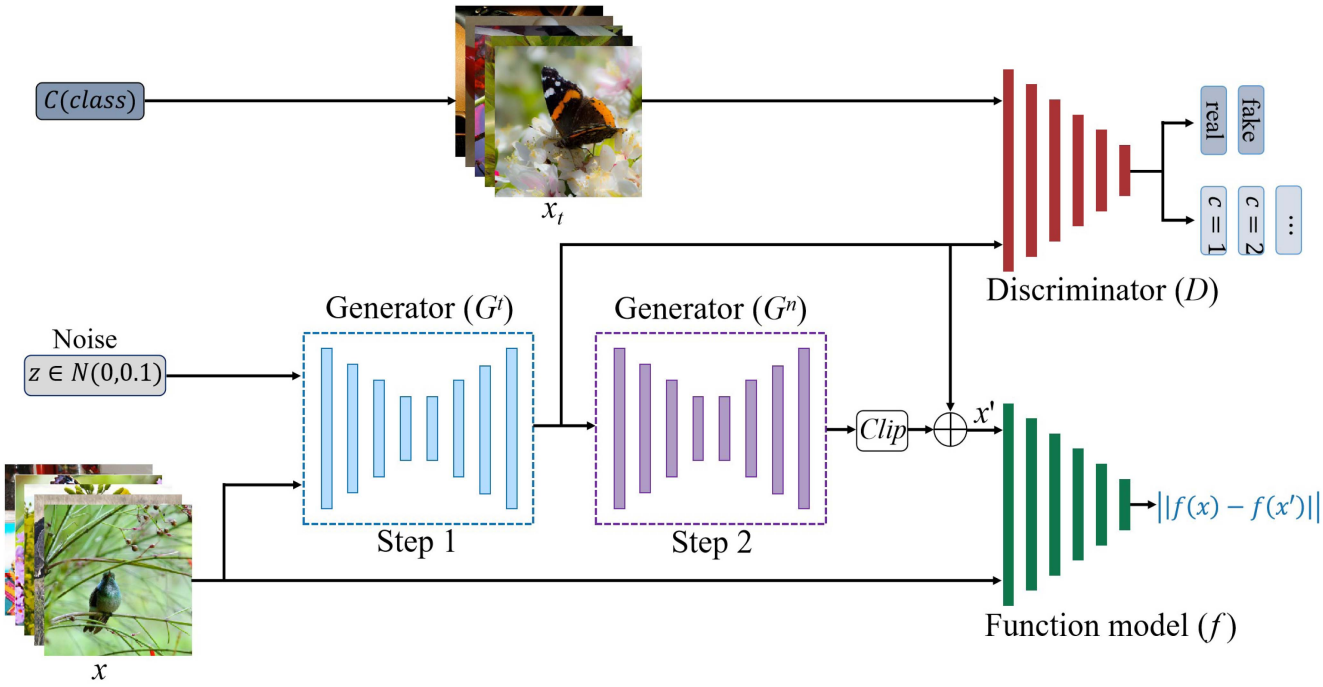


Fig. 4. Framework of GAG. It is composed of two generators G^t and G^n, a discriminator D, and a function model f. In the first step, G^t and D form a generative adversarial network that optimizes the transformation of x from the original space to the target space. Then, G^t is fixed, and G^n is optimized by the high-level representation guiding to generate perturbations that are added to the transformed images after clipping to obtain original predictions.

G^n is:

$$Loss_n = ||f(x) - f(Clip_\varepsilon\{G^n(G^t(x))\} + x)||_2, \quad (10)$$

here, $Clip_\varepsilon\{.\}$ performs per-pixel clipping of generated perturbations. In the inference phase, an original image can get a natural image that is completely different from it but has the same prediction result after passing through the two generators.

IV. Experiments

In this section, we first present the experimental settings, then comprehensively evaluate our approaches in different aspects. The code of this study is available at: https://github.com/Gaoyitu/HRG_GAG.

A. Experimental Settings

1) Datasets: We evaluate our proposed methods on MNIST, CIFAR-10, and ImageNet. MNIST contains 60000 training images and 10000 testing images with image shape (28×28). CIFAR-10 contains 50000 training images and 10000 testing images with shape $(32 \times 32 \times 3)$. For ImageNet, limited by computing resources, we select 10 categories, i.e., axolotl, ice cream, chameleon, admiral, ostrich, rapeseed, goldfish, hummingbird, violin, and teapot, from ILSVRC2012, each class contains 1300 training images and 50 testing images. The size of images in ImageNet is different, so we first crop the long edge or pad the short edge to square all images and then resize the images to $(224 \times 224 \times 3)$ for HRG. On this basis, we further resize all images to $(64 \times 64 \times 3)$ for GAG.

TABLE I
Classifier Architectures

C1	C2	C3
Conv(32,3,1), ReLU	Conv(32,3,1), ReLU	Conv(64,3,1), ReLU
MaxPooling	Conv(32,3,1), ReLU	Conv(64,3,1), ReLU
Conv(64,3,1), ReLU	MaxPooling	MaxPooling
MaxPooling	Conv(64,3,1), ReLU	Conv(128,3,1), ReLU
FC-200	Conv(64,3,1), ReLU	Conv(128,3,1), ReLU
Softmax 10	MaxPooling	MaxPooling
	FC-200	FC-256
	FC-200	FC-256
	Softmax 10	Softmax 10

These three classifiers are designed by us, so it is specifically explained here in the form of a table.

TABLE II
Classifiers for MNIST, CIFAR-10, and ImageNet

MNIST	CIFAR-10	ImageNet
C1	C3	VGG16
C2	VGG16	MobileNet

2) Classifiers: To reduce the randomness of experimental results and test the transferability of generated Type I adversarial images, we select two classifiers for each dataset. Table I shows the detailed structures of three classifiers that we design for MNIST (C1 and C2) and CIFAR-10 (C3). Table II shows all classifiers that are utilized in our experiments. We remove fully-connected layers of VGG16 [39] and MobileNet [40], and re-add a fully-connected layer with 10 neurons on them. All classifiers are trained by Adam optimizer [41] ($\beta_1 = 0.9, \beta_2 = 0.999$) with a batch size of 128, a learning rate of 0.0001, and epochs of 50.

TABLE III
THREE GENERATORS WITH DIFFERENT ARCHITECTURES FOR MNIST, CIFAR-10, AND IMAGENET

MNIST		CIFAR-10		ImageNet	
Encoder(G_{enc})	Decoder(G_{dec})	Encoder(G_{enc})	Decoder(G_{dec})	Encoder(G_{enc})	Decoder(G_{dec})
Conv(64,3,2) BN, LeakyReLU	FC-2048	Conv(128,3,2) BN, LeakyReLU	ConvT(1024,3,2) BN, ReLU	Conv(128,3,2) BN, LeakyReLU	ConvT(1024,3,2) BN, ReLU
Conv(128,3,2) BN, LeakyReLU	ConvT(256,3,1) BN, ReLU	Conv(256,3,2) BN, LeakyReLU	ConvT(512,3,2) BN, ReLU	Conv(256,3,2) BN, LeakyReLU	ConvT(1024,3,2) BN, ReLU
Conv(256,3,2) BN, LeakyReLU	ConvT(128,3,2) BN, ReLU	Conv(512,3,2) BN, LeakyReLU	ConvT(256,3,2) BN, ReLU	Conv(512,3,2) BN, LeakyReLU	ConvT(512,3,2) BN, ReLU
Conv(512,3,2) BN, LeakyReLU	ConvT(64,3,2) BN, ReLU	Conv(1024,3,2) BN, LeakyReLU	ConvT(128,3,2) BN, ReLU	Conv(1024,3,2) BN, LeakyReLU	ConvT(256,3,2) BN, ReLU
Conv(512,3,1) BN, LeakyReLU	ConvT(32,3,2) BN, ReLU		Conv(3,5,1) Tanh		ConvT(128,3,2) BN, ReLU
FC-512	Conv(1,5,1) Tanh				Conv(3,5,1) Tanh
FC-64					

TABLE IV
THE ARCHITECTURES OF GENERATORS AND DISCRIMINATORS FOR MNIST AND CIFAR-10

MNIST				CIFAR-10			
G^t/G^n		D		G^t/G^n		D	
Encoder	Decoder			Encoder	Decoder		
Conv(64,3,2) BN, LeakyReLU	ConvT(256,4,1) BN, ReLU	Conv(64,3,2) LeakyReLU Dropout 0.5		Conv(128,5,2) BN, LeakyReLU	ConvT(512,5,2) BN, ReLU	Conv(128,3,2) LeakyReLU Dropout 0.5	
Conv(128,3,2) BN, LeakyReLU	ConvT(128,3,2) BN, ReLU	Conv(128,3,2) BN, LeakyReLU Dropout 0.5		Conv(256,5,2) BN, LeakyReLU	ConvT(256,5,2) BN, ReLU	Conv(256,3,2) BN, LeakyReLU Dropout 0.5	
Conv(256,3,2) BN, LeakyReLU	ConvT(64,3,2) BN, ReLU	Conv(256,3,2) BN, LeakyReLU Dropout 0.5		Conv(512,5,2) BN, LeakyReLU	ConvT(128,5,2) BN, ReLU	Conv(512,3,2) BN, LeakyReLU Dropout 0.5	
	Conv(1,5,1) Tanh	Sigmoid 1	Softmax 10		Conv(3,5,1) Tanh	Sigmoid 1	Softmax 10

3) Generators and Discriminators: Table III presents the structural details of generators designed for HRG. The images in MNIST are relatively simple, so we design a complex generator to compress original images into a small space to ensure that the image information can be successfully hidden. For CIFAR-10 and ImageNet, we adopt fully convolutional networks to reduce model parameters. Tables IV and V show the structural details of generators and discriminators designed for GAG. For the same dataset, G^t and G^n have the same structures.

4) Baselines: Since most of existing methods for the privacy-preserving MLaaS only have experimental results on MNIST, we first compare HRG with SecureML [19], DeepSecure [20], EzPC [21], CryptoNets [24] and CryptoCNN [25] (where a temporary name is used here for convenience) on MNIST, and then evaluate the performance of HRG and GAG through extensive experiments on different datasets.

5) Implementation Details: All generators and discriminators are trained by Adam optimizer with $\beta_1 = 0.5$, $\beta_2 = 0.999$. For HRG, we set the batch size to 256 for MNIST and CIFAR-10, and 32 for ImageNet. The learning rate $= 0.0001$ and epochs $= 100$ for all datasets. For GAG, we set learning rate $= 0.002$, epochs $= 5000$, batch size $= 256$ for all datasets. Our experiments are performed on a PC with an Intel Core i7-10700 K CPU, 32 GB RAM, and a Nvidia GeForce RTX 3090.

TABLE V
ARCHITECTURES OF TWO GENERATORS AND A DISCRIMINATOR FOR IMAGENET

ImageNet			
G^t/G^n		D	
Encoder	Decoder		
Conv(128,52) BN, LeakyReLU	ConvT(1024,5,2) BN, ReLU	Conv(128,5,2) LeakyReLU Dropout (0.5)	
Conv(256,5,2) BN, LeakyReLU	ConvT(512,5,2) BN, ReLU	Conv(256,5,2) BN, LeakyReLU Dropout (0.5)	
Conv(512,5,2) BN, LeakyReLU	ConvT(256,5,2) BN, ReLU	Conv(512,5,2) BN, LeakyReLU Dropout (0.5)	
Conv(1024,5,2) BN, LeakyReLU	ConvT(128,5,2) BN, ReLU	Conv(1024,5,2) BN, LeakyReLU Dropout (0.5)	
	Conv(3,5,1) Tanh	Sigmoid 1	Softmax 10

B. Experimental Design

To comprehensively evaluate the performance of our new attempt, we design different experiments for the two proposed methods. 1) Comparison with the baselines. Due to expensive

TABLE VI
COMPARISON WITH THE STATE-OF-THE-ART SOLUTIONS ON MNIST

Method	Accuracy	Running Time (s)	Data Transfer	Predictions/Hour
SecureML*	93.40%	759	N/A	738
DeepSecure*	98.95%	6328	6479 GB	379
EzPC*	99.00%	41779	4104 GB	709
CryptoNets	98.95%	570	596 MB	51739
CryptoCNN	99.25%	320	437 MB	163840
HRG (ours)	**99.26%**	**2**	**61 MB**	**22346000**

*: The values are extrapolated.

TABLE VII
CLASSIFICATION ACCURACY OF HRG WITH DIFFERENT TRAINING SCHEMES ON MNIST, CIFAR-10, AND IMAGENET

Training Scheme	MNIST		CIFAR-10		ImageNet	
	C1	C2	C3	VGG16	VGG16	MobileNet
NA	99.30%	99.42%	87.25%	90.65%	93.60%	98.40%
Log-Likelihood Loss	99.26%	99.25%	84.57%	85.15%	86.40%	89.20%
L_2 Norm	**99.32%**	**99.41%**	**85.03%**	**85.36%**	**87.60%**	**89.80%**
KL-Divergence	**99.32%**	99.37%	82.52%	84.82%	84.20%	84.00%

NA means no training scheme.

computation and communication overhead, traditional methods, such as SMC and HE, are mostly validated on MNIST. Therefore, we compare and analyze HRG with traditional methods in accuracy, running time, data transfer and predictions per hour on MNIST. 2) Performances of HRG on different datasets. Evaluating the performances of HRG on MNIST, CIFAR-10 and ImageNet, and using three different training schemes to verify HRG's feasibility on images at different scales. In addition, we verify the ability of HRG to mask the original information of input images by visualizing transformed images. 3) Hiding data information in natural images. Verifying the feasibility of GAG based on the accuracy of transformed images under different perturbation intensity on the target classifier. And evaluating and analyzing the performances of GAG for privacy preservation by visualizing transformed images.

C. Quantitative Evaluation

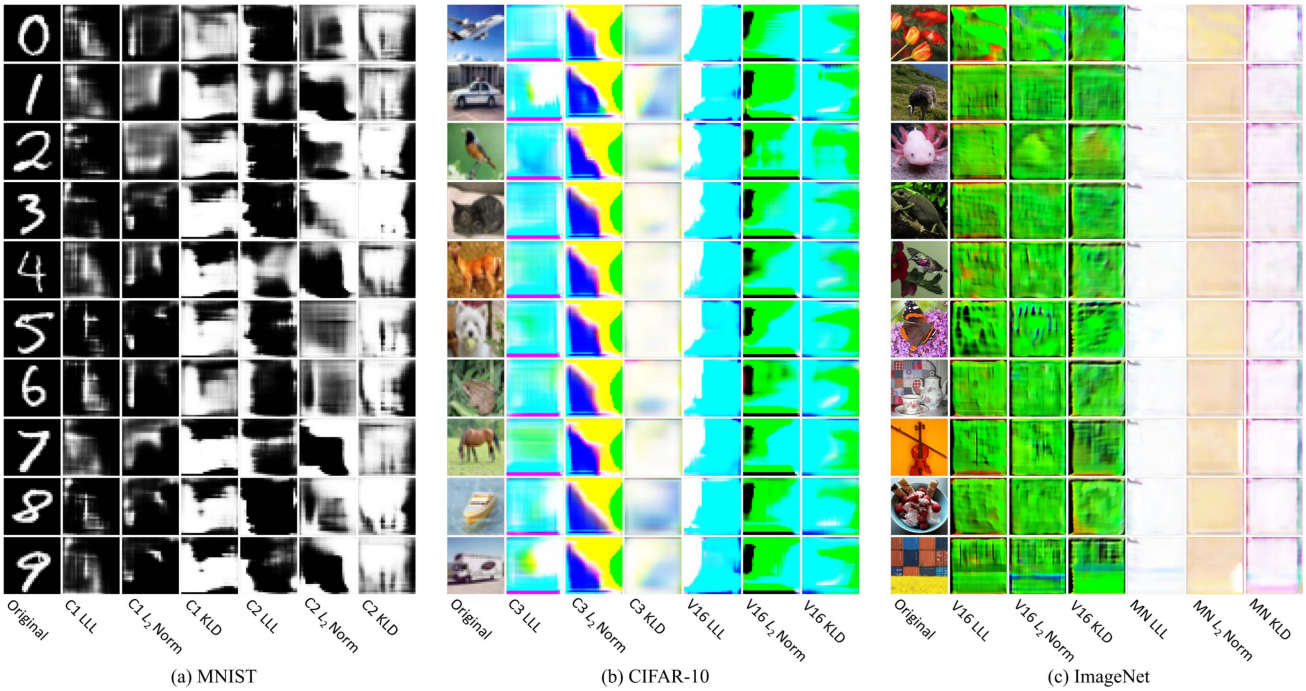
In this section, we comprehensively analyze and exhibit the performances of HRG and GAG following the Section "Experimental Design".

1) Comparison With the Baselines: We adopt C1 as the target model deployed on the Cloud and Log-Likelihood Loss to train a generator. Table VI shows the performances of different methods on MNIST. We can see that HRG achieves the highest classification accuracy, the least running time, the least data transfer, and the most predictions within an hour. SMC-based methods (SecureML, DeepSecure, and EzPC) require interactions between the client and server for each operation and therefore incur a huge amount of communication overhead. We can see that the data transfer of DeepSecure is 6479 GB and the data transfer of EzPC is 4104 GB. Meanwhile, SMC-based methods have low prediction efficiency. It can be seen that even the most efficient SecureML only makes 738 predictions per hour. Compared with SMC-based methods, the efficiency of HE-based methods (CryptoNets and CryptoCNN) has been greatly improved. CryptoNets can make more than 50000 predictions per hour and CryptoCNN can make more than 160000 predictions per hour. Even so, their efficiency is much lower than HRG. Compared with baselines, HRG adopts a completely different way for privacy preservation, and it does not require any additional communication and computation overhead for providing privacy-preserving predictions. We can see that HRG is extremely efficient and can make more than 20 million predictions per hour.

2) Performances of HRG on Different Datasets: Table VII shows the classification accuracy of HRG with different training schemes on different datasets. The row where NA is located shows the classification accuracy of original images. On MNIST, HRG achieves extraordinary classification accuracy on either C1 or C2, especially on C1, the classification accuracy of Type I adversarial images produced by the generators trained with L_2 Norm and KLD exceeds that of original images. On CIFAR-10, L_2 Norm performs the best, followed by Log-Likelihood Loss and KLD. Similarly, L_2 Norm achieves the best performance, followed by Log-Likelihood Loss and KLD on ImageNet. Overall, the generator trained by L_2 Norm is better than the generators trained by Log-Likelihood Loss and KLD.

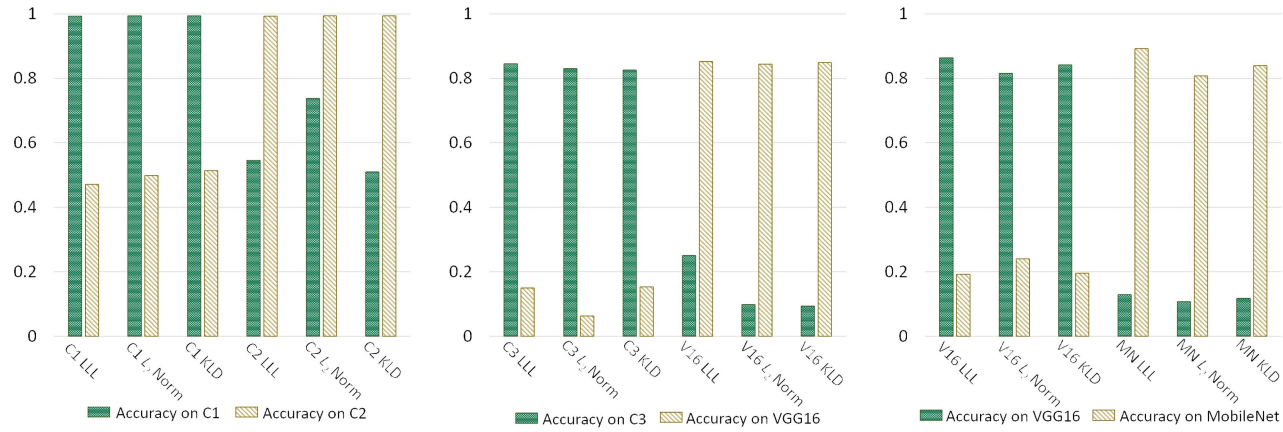
Besides reasonable classification accuracy, a strong information-hiding effect is also necessary. We randomly select an image from each category in each dataset and transform them into Type I adversarial images. Fig. 5(a) shows the original images and their Type I adversarial images from MNIST. We can see that the Type I adversarial images can effectively hide the original image information into noise in all cases. Fig. 5(b) shows the original images and their Type I adversarial images from CIFAR-10, and Fig. 5(c) shows the original images and their Type I adversarial images from ImageNet. As we can see, although the fully convolutional architecture is adopted for generators, the generated Type I adversarial examples still perfectly hide the original information of images.

In addition, from Fig. 5(c), we find that the Type I adversarial images generated by generators trained with different schemes for the same classifier are relatively similar, which means the form of Type I adversarial examples mainly depends on the target model. In other words, a well-trained generator is not



(a) MNIST (b) CIFAR-10 (c) ImageNet

Fig. 5. Examples of original images and their Type I adversarial images in MNIST, CIFAR-10 and ImageNet. The Type I adversarial images are produced by HRG with different loss functions. LLL is Log-Likelihood Loss, V16 is VGG16, and MN is MobileNet. In each sub-figure, the first column shows the original images and the second to seventh columns show the transformed images. Obviously, the generated Type I adversarial images can effectively hide the original image information into noise.



(a) MNIST (b) CIFAR-10 (c) ImageNet

Fig. 6. Transferability of HRG on MNIST, CIFAR-10 and ImageNet. LLL is log-likelihood loss, V16 denotes VGG16, and MN denotes MobileNet. C1 LLL represents the C1-specific generator trained by log-likelihood loss. It is can be seen that HRG has poor transferability, the Type I adversarial images produced by a generator specific to one classifier have poor classification accuracy on another classifier.

transferable, a generator trained for a particular model is only valid for that model. To verify our conjecture, we test the transferability of different generators. As shown in Fig. 6(a), shows the transferability of different generators on MNIST, the green bar chart represents the accuracy of transformed images on C1 and the yellow bar chart represents the accuracy of transformed images on C2. We can intuitively see that the images generated by the generator trained with C1 as the target model have good accuracy on C1, but poor accuracy on C2. Similarly, the images generated by the generator trained with C2 as the target model have good accuracy on C2, but poor accuracy on C1. This characteristic is more pronounced on CIFAR-10 and ImageNet, that is, the Type I adversarial images produced by a generator specific to one classifier have poor classification accuracy on another classifier. This is a good property that a generator is paired with a target model. A potential attacker must obtain both the transformed data and the desired model to get prediction results, but still cannot obtain users' original data.

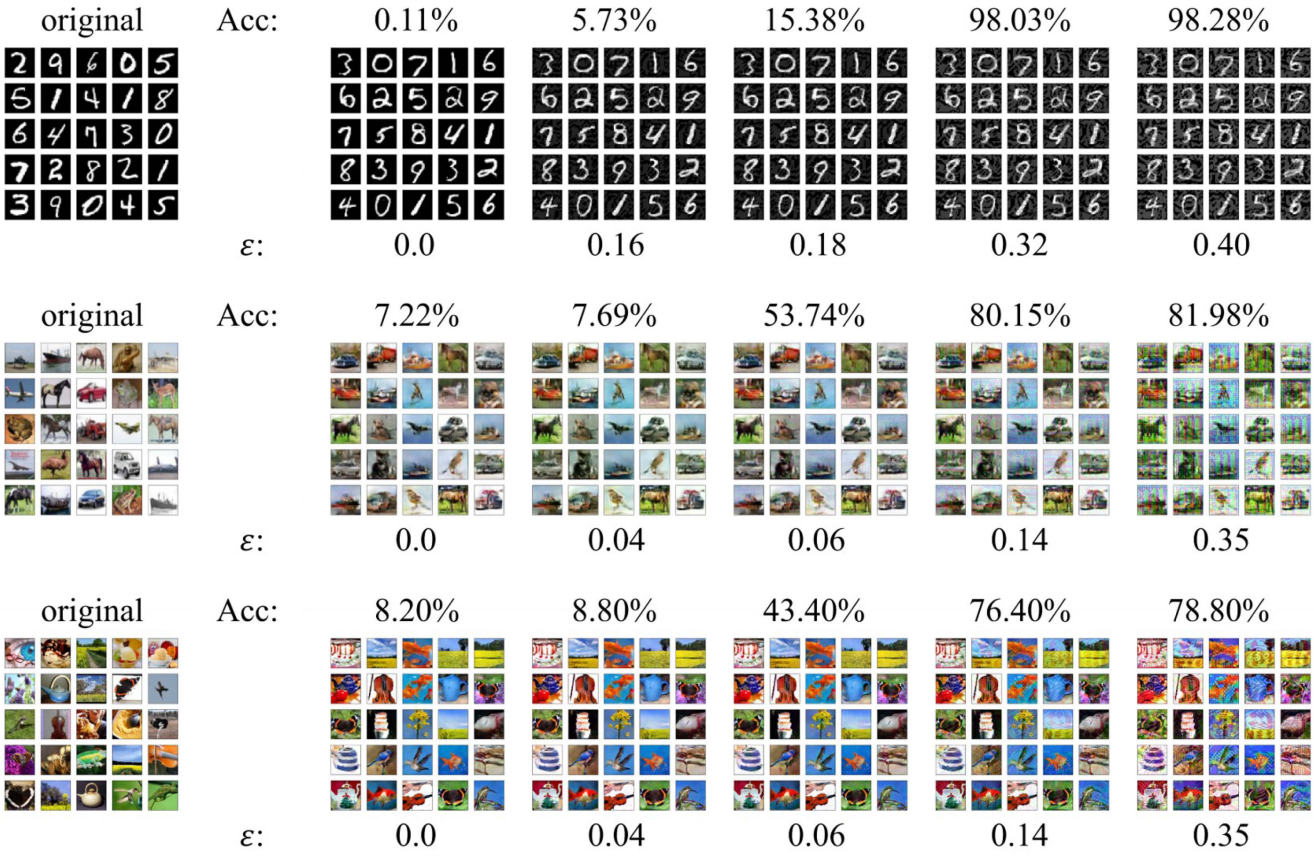


Fig. 7. Examples of the generated Type I adversarial images by GAG under different perturbation intensity. C1, C3 and VGG16 are the target model for MNIST, CIFAR-10 and ImageNet, respectively. The recognition rates at different perturbation intensity are marked on the top of the images, and the values of ε are marked on the bottom of the images. We can see that GAG generates natural images to hide data's original information.

3) Hiding Data Information in Natural Images: Hiding original images in natural images is much more difficult than hiding them in noise. Accordingly, the accuracy of GAG is slightly lower than that of HRG. G^t is responsible for transforming original images into other natural images to mask data information. However, the generated natural images do not gain the original predictions on the target model. G^n is responsible for generating perturbations that are added to the generated natural images to obtain original predictions. Large perturbations can improve the accuracy of the generated Type I adversarial images on the target model but will destroy their naturalness. To balance accuracy and naturalness, it is necessary to select an appropriate superior limit of perturbations, i.e., selecting the appropriate value of ε. We select the value of ε with an interval of 0.02 between $(0, 0.4]$. Fig. 8 shows the line charts of perturbation intensity and corresponding accuracy. On MNIST, C1 is used as the target model, we can see that large perturbations are required to obtain high accuracy. When ε is less than 0.14, the accuracy hardly improves, and when ε is greater than 0.32, the accuracy becomes stable. On CIFAR-10 and ImageNet, C3 and VGG16 are used as the target models, respectively. When ε is greater than 0.04, the accuracy increases rapidly, and when ε is greater than 0.14, the accuracy levels off. On the whole, ε is best set to 0.32 for MNIST and 0.14 for CIFAR-10 and ImageNet. Fig. 7 shows some transformed images from MNIST, CIFAR-10 and ImageNet, we pick out some important values of ε to exhibit. $\varepsilon = 0.0$ means no perturbations are added to the transformed images. The first column shows the original images and the second to sixth columns show the generated Type I adversarial images with different perturbation intensity. Obviously, as ε increases, the accuracy of the generated Type I adversarial images increases, and accordingly the perturbations in these images are more pronounced. In a word, GAG generates

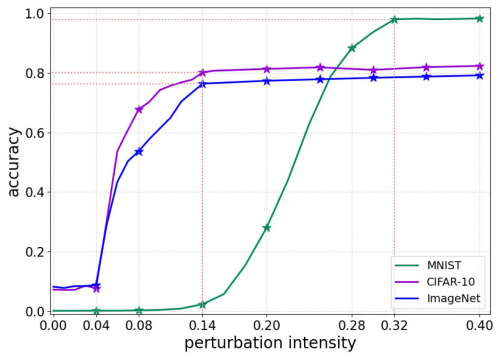


Fig. 8. Classification accuracy of GAG under different perturbation intensity. C1, C3 and VGG16 are used as the target model for MNIST, CIFAR-10 and ImageNet, respectively.

more natural images to hide data information at the cost of losing some prediction accuracy.

V. Conclusion

In this paper, we explore a new solution for preserving the privacy of data by implementing deep neural networks on transformed data. Our solution leverages Type I adversarial examples to hide the original information of raw data, and we propose two methods, HRG and GAG, to generate Type I adversarial examples. Our solution can effectively hide data information without incurring additional communication and computation overhead in the prediction process. We conduct extensive experiments on three real datasets and six DNN-based classifiers. The results show that our solution provides efficient, accurate, and scalable privacy-preserving predictions, and is suitable to be applied on large-scale images.

One shortcoming of our solution is that the generator needs to be deployed locally by users, which is unfriendly to non-expert users due to lack of domain expertise and computing resources. In addition, our solution directly returns raw prediction results, which reduces the privacy of user data. In our future work, we will investigate how to improve the friendliness of our solution for non-expert users and explore effective ways to hide the information of the prediction result.

References

[1] D. D. Lopes et al., "Machine learning partners in criminal networks," *Sci. Rep.*, 2022, doi: 10.1038/s41598-022-20025-w.

[2] Z. Gao et al., "Complex networks and deep learning for EEG signal analysis," *Cogn. Neurodynamics*, vol. 15, pp. 369–388, 2021.

[3] C. Szegedy et al., "Intriguing properties of neural networks," in *Proc. 2nd Int. Conf. Learn. Represent.*, 2014.

[4] I. J. Goodfellow, J. Shlen, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. 3rd Int. Conf. Learn. Representations*, 2015.

[5] K. Eykholt et al., "Robust physical-world attacks on deep learning visual classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1625–1634.

[6] S. Gao, S. Yu, L. Wu, S. Yao, and X. Zhou, "Detecting adversarial examples by additional evidence from noise domain," *IET Image Process.*, vol. 16, pp. 378–392, 2022.

[7] X. Chen, X. Gao, J. Zhao, K. Ye, and C. Xu, "AdvDiffuser: Natural adversarial example synthesis with diffusion models," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2023, pp. 4539–4549.

[8] B. Chen, J. Yin, S. Chen, B. Chen, and X. Liu, "An adaptive model ensemble adversarial attack for boosting adversarial transferability," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2023, pp. 4466–4475.

[9] Y. Dong et al., "Boosting adversarial attacks with momentum," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9185–9193.

[10] N. CaLRini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE 38th Symp. Secur. Privacy*, 2017, pp. 39–57.

[11] Z. Zhao, D. Dua, and S. Singh, "Generating natural adversarial examples," in *Proc. 6th Int. Conf. Learn. Representations*, 2018.

[12] Z. Li et al., "Sibling-Attack: Rethinking transferable adversarial attacks against face recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 24626–24637.

[13] J. Zhang, Y. Huang, W. Wu, and M. R. Lyu, "Transferable adversarial attacks on vision transformers with token gradient regularization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 16415–16424.

[14] S. Gao, S. Yao, and R. Li, "Transferable adversarial defense by fusing reconstruction learning and denoising learning," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2021, pp. 1–6.

[15] C. Xie, Y. Wu, L. Maaten, A. Yuille, and K. He, "Feature denoising for improving adversarial robustness," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 501–509.

[16] C. Song, K. He, J. Lin, L. Wang, and J. Hopcroft, "Robust local features for improving the generalization of adversarial training," in *Proc. 8th Int. Conf. Learn. Representations*, 2020.

[17] S. Gao et al., "Detecting adversarial examples on deep neural networks with mutual information neural estimation," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 6, pp. 5168–5181, Nov./Dec. 2023, doi: 10.1109/TDSC.2023.3241428.

[18] X. Li, Z. Xin, and W. Liu, "Defending against adversarial attacks via neural dynamic system," in *Proc. 36th Annu. Conf. Neural Inf. Process. Syst.*, 2022, pp. 6372–6383.

[19] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *Proc. IEEE 38th Symp. Secur. Privacy*, 2017, pp. 19–38.

[20] B. D. Rouhani, M. S. Riazi, and F. Koushanfar, "DeepSecure: Scalable provably-secure deep learning," in *Proc. 55th Annu. Des. Automat. Conf.*, 2018, pp. 1–6.

[21] N. Chandran, D. Gupta, A. Rastogi, R. Sharma, and S. Tripathi, "EzPC: Programmable and efficient secure two-party computation for machine learning," in *Proc. IEEE Eur. Symp. Secur. Privacy*, 2019, pp. 496–511.

[22] O. Ohrimenko et al., "Oblivious multi-party machine learning on trusted processors," in *Proc. 25th Usenix Secur. Symp.*, 2016, pp. 619–636.

[23] T. Hunt, C. Song, R. Shokri, V. Shmatikov, and E. Witchel, "Chiron: Privacy-preserving machine learning as a service," 2018, *arXiv:1803.05961*.

[24] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. E. Lauter, M. Naehrig, and J. Wernsing, "CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proc. 33th Int. Conf. Mach. Learn.*, 2016, pp. 201–210.

[25] E. Hesamifard, H. Takabi, and M. Ghasemi, "Deep neural networks classification over encrypted data," in *Proc. 9th ACM Conf. Data Appl. Secur. Privacy*, 2019, pp. 97–108.

[26] D. Takabi, R. Podschwadt, J. Druce, C. Wu, and K. Procopio, "Privacy preserving neural network inference on encrypted data with GPUs," 2019, *arXiv:1911.11377*.

[27] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier GANs," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 2642–2651.

[28] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," 2017, *arXiv:1701.07875*.

[29] S. Tang, X. Huang, M. Chen, C. Sun, and J. Yang, "Adversarial attack type I: Cheat classifiers by significant changes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 3, pp. 1100–1109, Mar. 2021.

[30] D. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. 2nd Int. Conf. Learn. Representations*, 2014.

[31] S. He et al., "Type-I generative adversarial attack," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 3, pp. 2593–2606, May-Jun. 2023.

[32] I. J. Goodfellow et al., "Generative adversarial nets," in *Proc. 27th Annu. Conf. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[33] L. Kong et al., "Time-aware missing healthcare data prediction based on ARIMA model," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, early access, Sep. 8, 2022, doi: 10.1109/TCBB.2022.3205064.

[34] L. Kong, L. Wang, W. Gong, C. Yan, Y. Duan, and L. Qi, "LSH-aware multitype health data prediction with privacy preservation in edge environment," *World Wide Web*, vol. 25, pp. 1793–1808, 2022.

[35] F. Wang et al., "Privacy-aware traffic flow prediction based on multi-party sensor data with zero trust in smart city," *ACM Trans. Internet Technol.*, vol. 23, pp. 1–19, 2023, doi: 10.1145/3511904.

[36] G. Chen, S. Chen, Y. Xiao, Y. Zhang, Z. Lin, and T. Lai, "SgxPectre attacks: Leaking enclave secrets via speculative execution," in *Proc. IEEE Eur. Symp. Secur. Privacy*, 2019, pp. 142–157.

[37] T. Graepel, K. E. Lauter, and M. Naehrig, "ML confidential: Machine learning on encrypted data," in *Proc. 15th Inf. Secur. Cryptology*, 2012, pp. 1–21.

[38] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, "Defense against adversarial attacks using high-level representation guided denoiser," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1778–1787.

[39] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learn. Representations*, 2015.

[40] A. Howard et al., "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[41] D. Kingma and J. Ba, "ADAM: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations*, 2015.

Song Gao received the B.E. degree from the China University of Geosciences, Wuhan, China, in 2013, the M.E. and Ph.D. degrees from Yunnan University, Kunming, China, in 2017 and 2021, respectively. He is currently a Postdoctor with the System Science, Yunnan University, Kunming, China. His research interests include deep learning, computer vision, social computing, and adversarial attack and defense.

Xiaoxuan Wang received the B.E. degree from the Minzu University of China, Beijing, China, in 2014, and the Ph.D. degree from Yunnan University, Kunming, China, in 2021. She is currently a Lecture with Yunnan Normal University, Kunming, China. Her current research interests include Big Data, spatial data mining, and spatial database.

Bingbing Song is currently working toward the Ph.D. degree with Yunnan University, Kunming, China. His current research interests include computer vision, deep learning, and AI security.

Renyang Liu received the B.E. degree from Northwest Normal University, Lanzhou, China, in 2017. He is a currently working toward the Ph.D. degree with Yunnan University, Kunming, China. His current research interests include computer vision, deep learning, generative model, and adversarial attack.

Shaowen Yao received the B.S. and M.S. degrees from Yunnan University, Kunming, China, in 1988 and 1991, respectively, and the Ph.D. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2002. He is currently a Professor with the National Pilot School of Software, Yunnan University. His current research interests include cloud computing, neural network theory and application, and Big Data computing.

Wei Zhou (Member IEEE) received the Ph.D. degree from the University of Chinese Academy of Sciences, Beijing, China. He is currently a Professor with the National Pilot School of Software, Yunnan University, Kunming, China. His current research interests include bioinformatics, neural network theory, and the distributed data intensive computing. He is a member of the Yunnan Communication Institute, a Fellow of the China Communications Society, and was selected as the Youth Talent Program of Yunnan University in 2017.

Shui Yu (Fellow IEEE) received the Ph.D. degree from Deakin University, Geelong, VIC, Australia, in 2004. He is currently a Professor with the School of Computer Science, University of Technology Sydney, Ultimo, NSW, Australia. He has authored or coauthroed more than 500 technical papers at different venues, such as IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE/ACM TRANSACTIONS ON NETWORKING, INFOCOM, and IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING. His current h-index is 71. His research interests include mathematical modelling, cybersecurity, Big Data, and network science. He is a Distinguished Visitor of IEEE Computer Society, and an elected member of Board of Governors of IEEE VTS and ComSoc, respectively. He is a member of ACM and AAAS.