# Type-I Generative Adversarial Attack

Shenghong He, Ruxin Wang, *Member, IEEE*, Tongliang Liu, *Senior Member, IEEE*, Chao Yi, *Member, IEEE*, Xin Jin, *Member, IEEE*, Renyang Liu, and Wei Zhou, *Member, IEEE*

**Abstract—**Deep neural networks are vulnerable to adversarial attacks either by examples with indistinguishable perturbations which produce incorrect predictions, or by examples with noticeable transformations that are still predicted as the original label. The latter case is known as the Type I attack which, however, has achieved limited attention in literature. We advocate that the vulnerability comes from the ambiguous distributions among different classes in the resultant feature space of the model, which is saying that the examples with different appearances may present similar features. Inspired by this, we propose a novel Type I attack method called generative adversarial attack (GAA). Specifically, GAA aims at exploiting the distribution mapping from the source domain of multiple classes to the target domain of a single class by using generative adversarial networks. A novel loss and a U-net architecture with latent modification are elaborated to ensure the stable transformation between the two domains. In this way, the generated adversarial examples have similar appearances with examples of the target domain, yet obtaining the original prediction by the model being attacked. Extensive experiments on multiple benchmarks demonstrate that the proposed method generates adversarial images that are more visually similar to the target images than the competitors, and the state-of-the-art performance is achieved.

**Index Terms—**Type I attack, resultant feature space, similar features, adversarial attack, generative adversarial network

## 1 Introduction

Most of existing deep neural networks (DNNs) have proven to be vulnerable to attacks by adversarial examples [1], [2]. This poses a great threat to the working conditions of the DNN-based real applications, such as autonomous driving [3], [4], object intrusion detection systems [5], [6] and face recognition [7], [8]. Hence, the research on the robustness of DNNs, including adversarial attacking and adversarial training, has recently attracted a great attention from the community. As for the adversarial attack,which can be roughly classified into Type I attack and Type II attack, it is aims at finding a modified example that can fool a target model with or without a defense mechanism. Type II attack tries to disturb the input data with an imperceptible perturbation, which can cause a misclassification by the attacked model [9], [10]. Currently, most of existing attack methods belong to Type II. An alternative attack manner is called Type I attack [11], however, which is seldomly focused in literatures. This attack targets at making significant changes to the input but the result is predicted as the original label by the attacked model. This is very dangerous in some scenario. For example, there are many images and videos on social networks, and the cost of manual review is too high and inefficient. Hence, various platforms will use deep learning models to detect and filter illegal images and videos. However, the platform's detection system will fail if someone uses a Type I attack to convert ordinary videos and images into unlawful content, which results in severely negative consequences. Existing research [12] shows that similar to the Type II attack, the Type I attack can also be used to exploit the weakness of the deep models.

Mathematically, assume that we have an example $x$ with the ground-truth label as $y$ and a to-be-attacked model $f$ that makes a correct prediction on $x$, i.e., $y = f(x)$. In the Type II attack, the adversarial example $x'$ is generated by adding an imperceptible perturbation to $x$, resulting in an incorrected prediction by $f$, i.e., $y \neq f(x')$. Regarding the Type I attack, we use $x$ to generate the adversarial example $x'$ which, however, exhibits a totally different appearance to $x$ and has a complete different label $y' \neq y$. A successful attack makes the model $f$ unconscious of the changes of the input category, i.e., $y = f(x')$. Hence, the Type I attack can be viewed as a task of input transformation which cannot be easily solved by, for example, adding noises to the input.

The above analyses inform us that the transformation from $x$ to $x'$ is indeed equivalent to the transformation between the distributions of two classes, that is, mapping the data distribution that $x$ follows to the data distribution that $x'$ follows while keeping the features by $f$ consistent. The possibility of such a transformation relies on the ambiguity of the model $f$ on the examples of these two classes.


- *Shenghong He, Ruxin Wang, Chao Yi, Xin Jin, and Wei Zhou are with the School of Software and the Engineering Research Center of Cyberspace, Yunnan University, Kunming, Yunnan 650500, China. E-mail: heshenghong@mail.ynu.edu.cn, rosinwang@gmail.com, {yichao, xinjin, zwei}@ynu.edu.cn.*
- *Tongliang Liu is with the UBTECH Sydney Artificial Intelligence Centre, School of Computer Science, Faculty of Engineering, University of Sydney, Darlington, NSW 2008, Australia. E-mail: tongliang.liu@sydney.edu.au.*
- *Renyang Liu is with Information Science and Engineering School, Yunnan University, Kunming, Yunnan 650500, China. E-mail: ryliu@mail.ynu.edu.cn.*


*Manuscript received 16 January 2022; revised 9 June 2022; accepted 17 June 2022. Date of publication 28 June 2022; date of current version 13 May 2023. This work was supported in part by the National Natural Science Foundation of China under Grants 62162067 and 62101480, in part by the Yunnan Province Science Foundation under Grants 202005AC160007 and 202001BB050076, in part by Yunnan Foundational Research Project under Grants 202201AT070173 and 202201AU070034, in part by the Platform for Situational Awareness of Industrial Internet Security in Yunnan Province, in part by the Open Foundation of Key Laboratory in Software Engineering, and in part by the Research and Development of Artificial Intelligence-Based Infrared Target Detection and Recognition Technology.*
*(Corresponding author: Wei Zhou.)*
*Digital Object Identifier no. 10.1109/TDSC.2022.3186918*

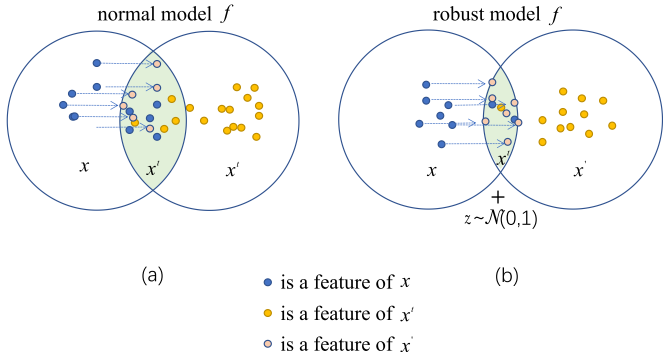



Fig. 1. Illustration of the Type I attack on the normal and robust models. The normal model indicates that there is no intended robust consideration during learning while the robust model is optimized by robust learning or adversarial learning. The overlapped green regions (where the model is ambiguous) are different for the normal model (the region is large) and the robust model (the region is small). For either model, if the two classes (i.e. $x$ and $x'$ have an overlapped region ($x^t$), it is possible to transform the examples of one class ($x$) to the examples located in this region. Hence, we advocate that the adversarial examples are located in this region, and we can transform $x$ to $x^t$ as indicated by the arrows.

Specifically, if the feature distributions of the two classes are overlapped with each other, the model will have low confidence on the examples located in the overlapped region and then, it is possible to transform the appearance of the example while keeping the feature unchanged, as shown in Fig. 1a. On the other hand, if the model well characterizes the decision boundaries of the two classes (for example, there is a large margin between the boundaries), it will be very difficult to make a transformation between them. We regard the Type I attack as exploiting the inter-class diversity property of the model, and as a counterpart, the Type II attack is exploiting the intra-class aggregation property of the model. Hence, during attack, we are reasonably motivated to employ generative functions to implement the distribution transformation.

Furthermore, consider that the model $f$ may have enhanced robustness owing to robust learning or some defense mechanisms, which means that the model has low ambiguity on different classes. In this case, the transformation is limited because the overlapped region is suppressed, as shown in Fig. 1b. To be clear, by involving robust learning, the resultant model exhibits improved discriminating ability between difference classes. Hence, the overlap between classes is suppressed and it becomes more difficult to transform a clean example to an adversarial example. Here, we advocate that it is encouraged to involve some randomness in the transformation process such that the generated examples exhibit more details relating the target distribution (i.e., $x'$). This is reasonable because when the overlapped region is limited, the information that is transformable is also constrained. To enrich the information, randomness is a good choice for modern generative models, which has already been verified in the task of face generation [13].

In this paper, we follow the above ideas and propose a novel Type I attack method, called generative adversarial attack (GAA). Specifically, we employ a generative adversarial network (GAN) architecture to learn the transformation from the original distribution to a pre-defined target distribution. To ensure the attack ability of the generated example, we employ the auxiliary classifier GAN (AC-GAN) loss to encourage the label of the generated data to be the same as the original data, which is expected by the attack objective. Simultaneously, a feature similarity constraint is imposed to keep feature consistency. Moreover, randomness is introduced in the generator to enrich the transformation and to diversify the generated examples. We conduct a series of examples on multiple public datasets, which demonstrate the superiority of the proposed model by comparing with the existing state-of-the-arts. Our contributions can be summarized as follows:

1) We propose a novel Type I attack method called generative adversarial attack (GAA). Specifically, we model the generation of adversarial examples as a discriminative inference process, while the well-trained generative model can be used to generate adversarial example very efficiently. This is different from previous works, where the adversarial example is generally generated via an optimization process which is time and resource-consuming. Notably, GAA is 60 times faster than the previous optimization-based method.
2) To ensure a smooth translation process from the original image to its counterpart adversarial example and to enrich the generated information in the target class, we develop an encoder-decoder-based generator $G$ by involving a random noise in the hidden feature space. This allows to generate visually pleasing adversarial examples. At the same time, GAA improves the black-box attack capability by imposing a soft constraint on the encoded features of intra classes.
3) Compared with the existing Type I attacks, the experiments indicate that the proposed model produces the adversarial examples with more naturalness and more features of the original image. The attack success rates on normal models and robust models are largely improved.

The rest of this paper is organized as follows. In Section 2, we briefly analyze the exiting adversarial attack methods and the defense methods. Section 3 presents the problem definition. The techniques of the Type-I generative adversarial attack are given in Section 4. In Section 5, we discuss the experiments, including the basic settings, the compared methods, and the experimental results. Section 6 draws the conclusions of this paper.

## 2 Related Work

Deep Neural Networks have a powerful ability of modeling complex problems and achieve superior performance, hence being widely used in real applications. Due to the unexplainability and the data bias, most of the deep models have been confirmed to be vulnerable to adversarial examples in computer vision tasks [14]. Advanced techniques for generating adversarial examples could produce a high success rates in attacking while remaining imperceptible perturbations in the examples [9], [15]. To perform attack in real world, the physical adversarial attack is also researched [15], [16], [17], especially in the topics of object detection [18], [19], [20], semantic segmentation [21], [22], [23], and natural language processing [24], [25], [26]. Here,

we focus on the adversarial attack on image classification, while is briefly reviewed in this section.

### 2.1 Type II Attack

Assume a classifier $f(x) : x \in X \longrightarrow y \in Y$, which outputs the label $y$ as the prediction of an input $x$. The Type II attack aims to find a small perturbation $\delta$ which is added to $x$, such that the generated input misleads the classifier output $f(x+\delta) \neq y$. The perturbation $\delta$ is usually constrained by $L_{p(p=1,2...\infty)}$ norm, i.e., $\|\delta\|_p \leqslant \epsilon$. Then, the constrained optimization problem can be written as:

$$argmaxJ(x+\delta, y) \qquad \|\delta\| \leqslant \epsilon, \tag{1}$$

where $J$ is, for example, the cross-entropy loss.

Adversarial attack is an active area in recent years that has witnessed extensive publications of advanced techniques. As mentioned previously, the research community consistently express interests on the Type II attack and have published a series of advanced algorithms, including the gradient-based methods [9], [15], [27], [28],such a fast gradient sign method (FGSM) [29], FGSM is a simple adversarial attack method, and it attacks the picture by maximizing the loss function $J$.Subsequently, it evolved into iterative FGSM [15], and can be formulated as:

$$\begin{aligned} x'_0 &= x, \\ x'_{t+1} &= x'_t + \boldsymbol{\alpha} \cdot sign(\bigtriangledown_x J(\theta, x, y)), \end{aligned} \tag{2}$$

where the $J$ is the loss function, $x$ and $y$ represent the input image and the true label, $\theta$ represents the network parameters, $\alpha$ is the step size, and $sign()$ is the sign function. As a seminal work, momentum iterative FGSM (MI-FGSM) [28] is proposed,which intergrates the momentum term into the iterative process for attacks to ensure the noise adding direction more smooth:

$$\begin{aligned} g_{i+1} &= \mu \cdot g_i + \frac{\bigtriangledown_x J(x'_i, y)}{\| \bigtriangledown_x J(x'_i, y)\|_1}, \\ x'_{i+1} &= Clip_{x,\epsilon}\{x'_i + \alpha \cdot sign(g_{i+1})\}, \end{aligned} \tag{3}$$

where $\mu$ is the decay factor of the momentum term, and the $Clip$ function clips the input values to a specified permissible range i.e $[x-\epsilon, x+\epsilon]$ and [0,1] for images.Compared with classical FGSM, MI-FGSM is able to craft a better adverarial examples.

As a optimization-based method, C & W [30] first treats adversarial examples as variables, then it try to optimize and diminish the discrepancy between adversarial examples and clean examples, as well as increase the probability of misclassification. It can be expressed as:

$$minD(x, x+\delta) + c \cdot f(x+\delta), \tag{4}$$

where the $\delta$ is the disturbance, $D$ is the distance between the clean image and the adversarial example, $f$ is the classification model, and $c$ is the hyperparameter. And the local-region attack methods [31], [32], for example, One Pixel Attack [32], an extreme adversarial attack method, can attack the classification model by only changing the value of one pixel in the image.

### 2.2 Type I Attack

Unlike the Type II attack, the Type I attack tries to make a significant change to clean image $x$, but still fools the classifier to produce the original label. Mathematically, this process can be described as

$$\begin{aligned} x' &= G(x), \\ f_1(x') &= f_1(x), \\ f_2(x') &\neq f_2(x), \end{aligned} \tag{5}$$

where $f_1$ is the classifier to be attacked while $f_2$ is the attacker, which could be an oracle classifier, e.g., a human eyes.

However, The Type I attack has received very limited attention. For example, [11] proposed that the existing deep models were vulnerable to both Type I and Type II attacks. [33] explained the relationship between the two types of attacks, and provided a supervised variational auto-encoder (SVAE) model. Based on the original Variational Auto-encoder (VAE) [34], this method proposed a fake latent variable that followed the standard Gaussian distribution [35]. The true and fake latent variables were distinguished by using a discriminator such that the model was sensitive in generating normal examples and adversarial examples. Then, the fake latent variable helped the perturbation generation to be controllable. It can be expressed as:

$$\begin{aligned} J = &-KL[q(z|x)||p(z)] + E_{z\sim q(z|x)}[log(p(y|z))] \\ &+ E_{z\sim q(z|x)}[log(p(x|z))], \end{aligned} \tag{6}$$

where the $KL$ is Kullback-Leibler, $q(z)$ is an arbitrary distribution in hidden space and $q(z|x)$=$N(\mu(x:\theta_{enc}), \sigma(x:\theta_{enc}))$ is variance,and $\theta_{enc}$ represents the encoder.

Both types of attack try to analyze the imperfect characterization of data distribution by the to-be-attacked model. The difference says that the Type II attack focuses on the improper intra-class aggregation and finds the examples that do not follow the distribution of a certain class, through different manners. Instead, the Type I attack is interested in the unpleasing inter-class diversity and generates the examples that may simultaneously follow the distribution of two classes, thus confusing the model. The proposed method is based on this understanding and tries to exploit the ambiguity of the model to different classes, resulting in a novel generative model based on the generative adversarial network. In this paper, we design a generative adversarial attack model (GAA), which is the supervised extension from the original Generative Adversarial Network (GAN). The generator tries to transform the clean image to its corresponding adversarial example by editing the hidden feature space based on the original example.

The generator try to transform the clean image to its corresponding adversarial example by editing the hidden feature space based on the original example.

### 2.3 Defend Against Adversarial Examples

The vulnerability of neural networks poses a serious security problem for applying deep neural networks in real applications. Recently, many methods have been proposed to defend against adversarial examples. [9], [36] proposed to inject adversarial examples into the training data to

increase the network robustness. Tramer *et al.* [37] pointed out that such adversarially trained models are still vulnerable to new adversarial examples, and proposed an ensemble adversarial training scheme, which augmented the training data with the examples transferred from other models. [38], [39] applied random transformation to the model inputs at inference time to mitigate the adversarial effects. Dhillon *et al.* [40] pruned a random subset of activations according to their magnitude to enhance network robustness. Prakash *et al.* [41] proposed a framework which combined pixel deflection with soft wavelet denoising to defend against adversarial examples. [42], [43], [44] leveraged generative models to purify the adversarial images such that the examples could follow the distribution of the clean images. Bin Liang *et al.* [45] considered the perturbation to images as a kind of noise and introduced two classic image processing techniques, including scalar quantization and smoothing spatial filter, to reduce the attack effect of adversarial examples.

These defense methods are against the Type II attack, whereas no defense methods are proposed for the Type I attack. Hence, we will employ these Type II defense methods to test the performance of defending the Type I attack in Section 5.6.

### 2.4 GAN for Adversarial Attacks and Defenses

There are already many GAN-based methods for Type II attack. For example, Mangla *et al.* [46] proposed AdvGAN++ which used the hidden layer vector in the classifier as the input of GAN to generate adversarial examples. This method contained the target model M, the feature extractor F, the generator G, and the discriminator D. The clean image was processed by F to obtain the feature vectors which were used as the prior information. The features and the noise vector z were cascaded and input to G to generate adversarial examples. Natural GAN [47] was an innovative method based on the WGAN framework, which focused on finding the hidden vector of the adversarial example in the low-dimensional hidden feature space such that the generated adversarial example was natural to human recognition. This method contained two stages, where the first stage established a correspondence between the sample space and the hidden feature space, while the second stage searched for the hidden representation of the expected adversarial examples. Liu *et al.* [48] proposed RobGAN which introduced adversarial examples in the training of GAN and strengthened the discriminating ability of the discriminator D. This method learnt the adversarial factors to improve the quality of the generated adversarial examples.

Regarding adversarial defense, there have already been many GAN-based methods. Jin *et al.* [49] considered that the imperceptible disturbance caused the misclassification problem and proposed an APE-GAN algorithm to eliminate the adversarial disturbance from the input image. The authors used WGAN to reconstruct the adversarial image being similar to the original image. Similar to APE-GAN, Samangouei *et al.* [50] proposed Defense-GAN which employed WGAN to reconstruct the adversarial examples for defense. The difference of these two methods on the training process were that APE-GAN input the clean images and the adversarial samples into the discriminator and the generator for training, while Defense-GAN used random noise instead of the adversarial examples.

## 3 Problem Definition

Given a clean image $x$ with the ground-truth label $y$, our task is to synthesize an adversarial example $x'$ with the oracle-determined label $y^t \neq y$, while the attacked model $f$ predicts $x'$ as $y = f(x')$. Let $x^t$ denote the sample that has the same label $y^t$ as $x'$. The transformation function from $x$ to $x'$ is denoted as $x' = G(x;\theta)$, where $\theta$ is the parameter of the transformation. Assume that the all the examples including $(x, y)$, $(x', y^t)$, and $(x^t, y^t)$ follow the distribution $P_o$ which is determined by an oracle classifier, e.g. human. Then, our objective is

$$\begin{aligned} &min_\theta \log\left(P_o(x'|y;\theta)\right) - \log\left(P_o(x'|y^t;\theta)\right) \\ &s.t.\ f(x) = f(x'). \end{aligned} \tag{7}$$

The above problem informs that the generated adversarial example $x'$ is different to those examples belonging to the class $y$, but is highly similar to the examples belong to $y^t$. As expected, the examples of the two classes exhibit noticeable differences on their appearances. By keeping the prediction of $f$ on $x$ and $x'$ consistent, a successful Type I attack is achieved.

Based on the above formulation, the minimization is implemented by our proposed generative adversarial attack method which is detailed in the following.

## 4 Generative Adversarial Attack

In this section, we introduce the framework and the training details of the proposed GAA.

### 4.1 GAA Structure and Loss Function

The proposed generative adversarial attack (GAA) is composed of three important models, i.e., a generator network $G$, a discriminator network $D$, and a to-be-attacked model $f$ which is termed as the function model. The whole architecture is illustrated in Fig. 2. The generator and the discriminator compose of a generative adversarial network, which optimizes the transformation from the original image $x$ to the images in the target domain. During the transformation process, we note that the label of the generated example is important for generating class-related image details. Hence, we build up the training loss based on the AC-GAN loss [51]. Mathematically, the loss function is:

$$\begin{aligned} L_{GAN} &= \min_G \max_D V(D,G) \\ &= \mathbb{E}_{x^t \sim Pdata(x^t)}[\log D(x^t)] \\ &\quad + \mathbb{E}_{(\upsilon) \sim Pdata(x,z)}[\log\left(1 - D(G(\upsilon))\right)] + \lambda L_C, \end{aligned} \tag{8}$$

where the $\upsilon$ comes from the original image $x$ and Gaussian noise $z$, and $x^t$ is the target image, generator computes as $x' = G(\upsilon)$, the discriminator classifies the example $x^t$ of the $y^t$-th class as real and the generated example $x'$ as fake, and $\lambda$ is a hyper-parameter that balances the influence of the category loss $L_C$:

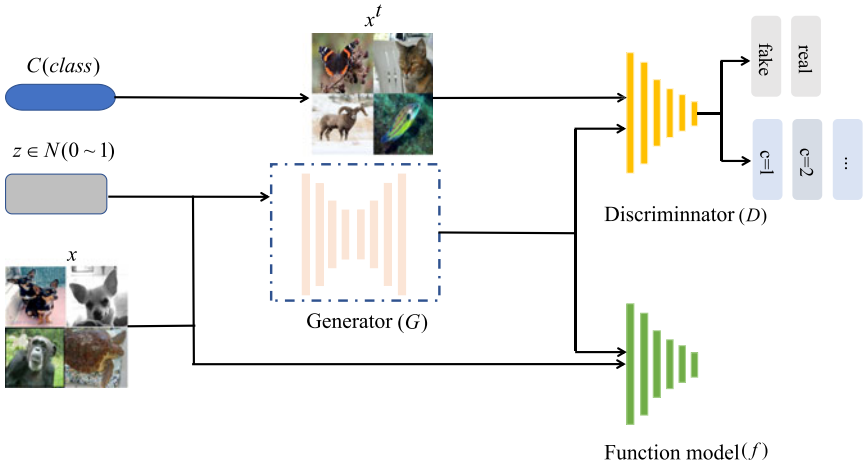



Fig. 2. Framework of the generative adversarial attack. It is composed of the generator $G$, the discriminator $D$, and the function model $f$.

$$L_C = \mathbb{E}[\log P(c = y^t|x^t)] + \mathbb{E}[\log P(c = y|x')], \tag{9}$$

where the discriminator forces the label of the generated sample to be the same as the target example $x^t$. Eq. (8) can be decomposed into:

$$\begin{aligned} \max_D V(D,G) &= \mathbb{E}_{x^t \sim Pdata(x^t)}[\log D(x^t)] \\ &+ \mathbb{E}_{(\upsilon) \sim Pdata(x,z)}[\log(1 - D(G(\upsilon)))] + L_C \end{aligned} \tag{10}$$

$$\min_G V(D,G) = \mathbb{E}_{(\upsilon) \sim Pdata(x,z)}[\log(1 - D(G(\upsilon)))] + L_C \tag{11}$$

where needs to maximize $D$. For the true distribution $x^t$, $D(x^t)$ should be close to 1, and for the generated distribution $x'$, $D(x')$ should be close to 0. $G$ should be minimized, generating samples $x'$ can fool $D$, and making $D(x')$ close to 1. To simplify the transformation process and generate more vivid details, we propose an operation based on the residual learning strategy. As seen in Fig. 3, the generated example $x'$ is computed as

$$x' = G(x, z). \tag{12}$$

We need to convert from $x$ to $x'$, $x'$ and $x$ have a huge gap, and $x'$ is the same category as the target image $x^t$, while still be misleading $f$ classifies $x$ category. The conversion process of this attack is fundamentally different from the Type II attack. At the same time, this is much more difficult than Type II attacks. The difference states that on one hand, we do not constrain the perturbation size within a predefined threshold and on the other hand, we restrict the resultant label of $x'$ to be $y$. With this understanding, we could view that the Type I attack and the Type II attack are coupled attacks on different classes.

In the implementation of the generator G, we employ a U-net architecture composed of an encoder and a decoder. The encoder is fully convolutional while the decoder is a deconvolutional neural network. By simply using the encoder-decoder model, we encounter a problem that the synthesized images are noisy as show in the second column of Fig. 4, even though the noisy image could attack the model $f$ successfully. This is perceptually unacceptable in real attack scenarios since the adversarial example can be easily detected by an oracle. While existing research [13] suggests that adding randomness could improve the vividness of the synthesized details, we propose to add a Gaussian random variable on the embedded code. Let $G_{enc}$ denote the encoder part of $G$ and $G_{dec}$ denote the decoder part, the Gaussian variable $z \sim \mathcal{N}(0, 1)$ is added as

$$G(x) = G_{dec}(G_{enc}(x) + z). \tag{13}$$

In this way, the generated images would exhibit more perceptible details, as illustrated in the last column of Fig. 4.

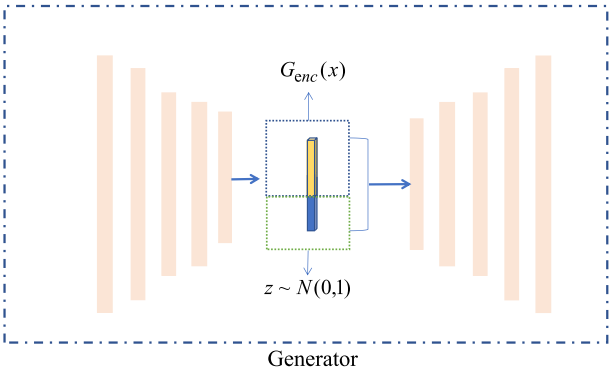



Fig. 3. After using $G_{enc}$ to extract the image features, we will feature and Gaussian noise z in the RGB channel for feature concatenation and inputting to $G_{dec}$.

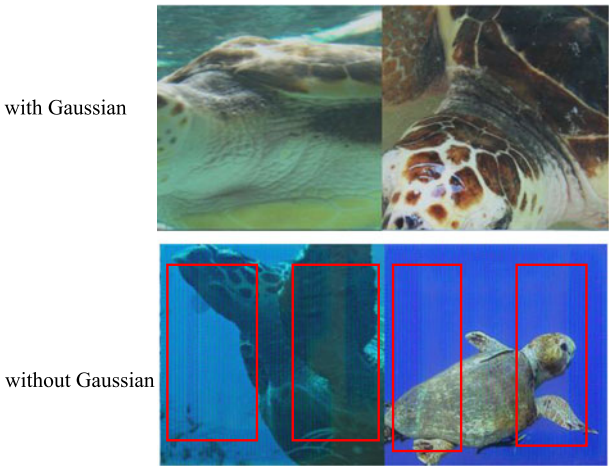



Fig. 4. The adversarial examples generated by GAA with and without the Gaussian randomness.

The above optimized GAN guarantees to synthesize a vivid image similar to images of class $y^t$. Now consider that another objective is to enforce the prediction of $x'$ by $f$ being $y$. A possible manner is to directly minimize $L_y = J(y, f(x'))$, where $J$ denotes the loss function measuring the difference between $y$ and $f(x')$, for example the cross-entropy loss. But we empirically find that this loss cannot produce strong transferability of the adversarial examples, which may be caused by the hard constraint on labels. Alternatively, we know that when $x$ and $x'$ have the same label, their features extracted by $f$ should be similar to each other. Then, it is natural to minimize the distance between the features, i.e.,

$$L_f = \|f_e(x) - f_e(x')\|^2, \tag{14}$$

where $f_e$ is the feature extraction function in the model $f$, for example the output of the stage-2 in Resnet [52]. This loss imposes a soft constraint instead of a hard constraint between $x$ and $x'$, which is demonstrated to be effective in experiments.

By combining the above introduced losses, we obtain the final loss for training the whole model:

$$L = L_{GAN} + \gamma L_f, \tag{15}$$

where $\gamma$ is a hyper-parameter to balance the influence of $L_f$.

### 4.2 Training Details

Here, we introduce a specific setting in our method. From the previous discussion, we know that the proposed model transforms the image $x$ belong to class $y$ to the image $x^t$ belonging to class $y^t$. Hence, in the training process, we need to sample a pair $(x, x^t)$ from two classes. Specifically, assume that a dataset has $C$ classes in total. We randomly draw a sample $x$ from the whole dataset, which has the ground-truth label $y$. Then, the target image $x^t$ is randomly selected from the class $y^t = (y+1)\%C$. The Gaussian vector $z$ is randomly sampled at each time. Note that the function model $f$ is fixed during training since it is the model to be attacked and is used to measure the distance between the features of $x$ and $x'$. The other training settings of the proposed model follow most of existing GAN settings, which will be detailed in Section 5.1.

The proposed model follows the convergence property of a typical GAN, which plays a Nash equilibrium problem. That is saying, the generator and the discriminator are optimized alternately until there is no incentive for both models to deviate from their states. As well known, training such a GAN model is not a stable process. Hence, to alleviate this issue, we adjust the learning rate dynamically when updating the parameters of $G$. Fig. 5 illustrates the curves of the training losses on different datasets. It can be seen that the models on MINST and CIFAR-10 converge quickly, being stable at around 1000 epochs and 2000 epochs, respectively. The models on ImageNet require more epochs for convergence. Fig. 6 displays the attack success rate on the verification set of MNIST, CIFAR-10, and ImageNet, in which similar convergence effects are exhibited.

### 4.3 Generation of Adversarial Examples

Given the well-trained generator $G$ and an original image $x$, we generate the adversarial example as following. A random vector $z$ is drawn from the standard Gaussian distribution. To perform attacking based on $x$, we manually identify its label $y$ and then determine the target label according to the class pairs used in training, i.e., $y^t = (y+1)\%C$. The target image $x^t$ is randomly sampled from the data set of the $y^t$-th class. Finally, the adversarial example is computed as:

$$x' = G_{dec}(G_{enc}(x) + z). \tag{16}$$

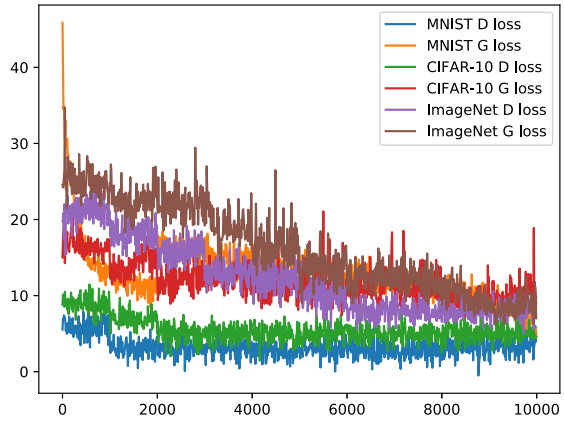



Fig. 5. It is the training loss value of GAA on MNIST, CIFAR-10 and ImageNet. The x-coordinate is an epoch,y-coordinate is the loss value.

## 5 Experiments

In this section, we conduct a series of experiments on multiple datasets to validate the effectiveness of the proposed method. The competitor for comparison is selected as the existing Type I attack method [33], which introduces two models including SVAE and StyleGAN. It is verified that the existing Type II attack defense method cannot withstand Type I attack.

### 5.1 Settings

#### 5.1.1 Datasets

To examine the attack performance of the proposed method on different data, we involve four datasets, including MNIST[1], CIFAR-10[2], ImageNet ILSVRC 2016 validation set[3], and CelebA[4].

#### 5.1.2 Attacked Models

The models to be attacked include FC which is composed of 5 fully connected layers, CFC which is composed of 3 convolutional layers and 2 fully connected layers, VGG16[5], ResNet18, ResNet50[6], ResNet101[7], DenseNet121[8], InceptionV3[9], and EfficientNetB0[10]. On each dataset, we select

1. http://yann.lecun.com/exdb/mnist/
2. http://www.cs.toronto.edu/~kriz/cifar.html
3. http://image-net.org/challenges/LSVRC/2016/index
4. http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html
5. https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5
6. https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
7. https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet101_weights_tf_dim_ordering_tf_kernels.h5
8. https://storage.googleapis.com/tensorflow/keras-applications/densenet/densenet121_weights_tf_dim_ordering_tf_kernels.h5
9. https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels.h5
10. https://storage.googleapis.com/keras-applications/efficientnetb0.h5

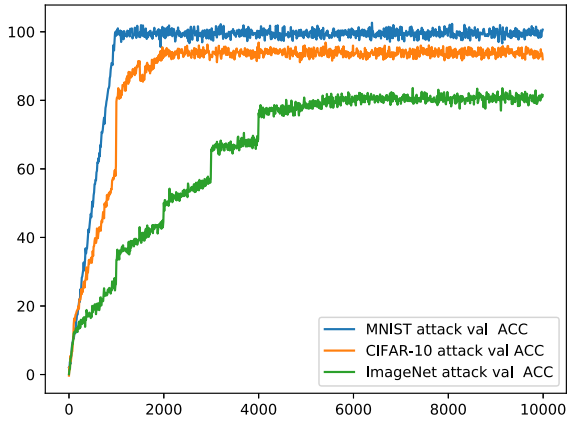



Fig. 6. The x-coordinate is an epoch,y-coordinate is the attack acc(%).

from the above models such that the model size matches with the size of the dataset, to avoid underfitting or overfitting. The models with public links are pretrained on ImageNet. Given a certain dataset, the model is trained from scratch or finetuned on the dataset before being attacked, except for using an ImageNet-pretrained model on the ImageNet dataset.

#### 5.1.3 Metrics

To measure the success rate of Type I attack, we compute the recognition rate of each attacked model on the generated adversarial examples.

$$P = \frac{TP}{TP + FP}, \tag{17}$$

where $TP$ is true positive, which indicates that the detection of the adversarial examples is the result of the original images label, $FP$ is false positive, which represents the result of detecting that the adversarial examples are not recognized as the original images label. Ideally, if the performance of the attack method is pleasing, the recognition rate remains almost unchanged between the original examples and the adversarial examples. In each case of the following comparisons, we randomly select 1000 images from the corresponding dataset as the original images and then compute the recognition rate. This process is repeated for 3 times in each comparison and the reported performance is an averaged value.

#### 5.1.4 Implementations

In the proposed model, $G_{enc}$ is implemented as the stacked convolutional layers of VGG16 without the fully connected layers. $G_{dec}$ is built as the architecture [ConvT-512/BN, ConvT-256/BN, ConvT-128/BN, ConvT-128/BN, ConvT-64/BN, ConvT-32/BN, ConvT-3/Tanh] with the kernel size of 3. $D$ is realized as VGG16 with proper modification according to the class numbers in those datasets. Considering that the data variance of MNIST is relatively low, $G_{enc}$, $G_{dec}$, and $D$ are implemented as lightweight models which are detailed in the Table 1.

The feature distance between the original image $x$ and the adversarial example $x'$ is computed based on $f_e$. In different models, the selected layers for computing $f_e$ are listed in Table 2. Note that for all models, we use multiple layers to extract features such that the similarity between the two examples can be enhanced during training.

TABLE 1
The Architecture of GAA Used in MNIST

| G | | D | |
|---|---|---|---|
| IN($28 \times 28$ image) | IN($G_{enc}(x) + z$) | IN($28 \times 28$ image) | |
| Conv1-64<br>BN<br>Maxpool | FC-6086 | Conv1-64<br>BN<br>Maxpool | |
| Conv-128<br>BN<br>Maxpool | ConvT-256 BN | Conv-128<br>BN<br>Maxpool | |
| Conv-256<br>BN<br>Maxpool | ConvT-128 BN | Conv-256<br>BN<br>Maxpool | |
| Conv-512<br>BN<br>Maxpool | ConvT-64 BN | Conv-512<br>BN<br>Maxpool | |
| Conv-512<br>BN | ConvT-32 BN | Conv-512<br>BN | |
| FC-512<br>FC-100 | ConvT-1<br>Tanh | FC-1 | FC-10 |

For the hyper-parameters, we set $\gamma = 0.8$ for MNIST and CIFAR-10, $\gamma = 1$ for ImageNet and CelebA. The whole model is trained by using the Adam algorithm [53]. The learning rate is set to 0.0002 for MNIST and CIFAR-10, 0.0001 for ImageNet and CelebA. The batch size is set to 512 for MNIST, 128 for CIFAR-10, and 32 for ImageNet and CelebA. Training runs for 100,000 iterations on all datasets. All the experiments are conducted on a GPU server with one Intel Xeon E5 2620 v4, 128 GB RAM, and two NVIDIA RTX

TABLE 2
The Selected Layers of Different Models for Computing $f_e$

| Model | Layers |
|---|---|
| FC | fc_3,fc_5 |
| CFC | conv2<br>conv3 |
| VGG16 | block2_conv1<br>block3_conv1<br>block4_conv1 |
| ResNet50 | conv2_block1_1_conv<br>conv3_block1_1_conv<br>conv4_block1_1_conv |
| DenseNet121 | conv2_block1_1_conv<br>conv3_block1_1_conv<br>conv4_block1_1_conv |
| ResNet101 | conv2_block1_0_conv<br>conv3_block1_0_conv<br>conv4_block1_0_conv<br>conv5_block1_0_conv |
| InceptionV3 | conv2d_1<br>conv3d_1<br>conv4d_1 |
| EfficientNetB0 | block2a_expand_conv<br>block3a_expand_conv<br>block4a_expand_conv<br>block5a_expand_conv |

TABLE 3
Attack Performance on MNIST

| | GAA | SVAE | ORIGINAL |
|---|---|---|---|
| FC | 96.9% | **97.1%** | 99.2% |
| CFC | **97.6%** | 97.1% | 99.3% |
| ResNet18 | **98.3%** | 97.9% | 99.6% |

*The GAA and SVAE columns represent the attack's success rate, and the ORIGINAL column represents the accuracy of the clean image on models such as ResNet18.*

2080 TI GPUs. The implementation is under CentOS 7 with Python 3.6 and Tensorflow-GPU 2.0.

## 5.2 Stability of G Structure

In this part, we explore the impact of the $G$ structures on MNIST, CIFAR-10, and ImageNet. A layer in our implementation consists of the Conv, BN, and Maxpool operations. The number of layers in $G_{enc}$ and $G_{dec}$ are equal. We test the cases that $G$ contains 8, 12, and 16 layers in MNIST and CIFAR10, which are denoted as $G_1$, $G_2$, and $G_3$, respectively. The experimental results are shown in Table 4. On ImageNet, we use the layers of VGG13, VGG16, and VGG19 as $G_{enc}$, which are denoted as $G_{V13}$, $G_{V16}$, and $G_{V19}$, respectively, while the layer numbers in $G_{dec}$ are the same as $G_{enc}$. The experimental results are listed in Table 5. These results show that the best performance is obtained when the model complexity matches the data complexity.

$G_3$'s attack success rate fluctuates greatly on the MINST. Others are relatively stable. This may be because $G_3$ is too large compared to MNIST, which makes it prone to overfitting during network training.

## 5.3 Comparison of Attack Performance

### 5.3.1 Results on MNIST

On this dataset, we employ FC, CFC, and ResNet18 as the attacked models, which are trained from scratch. The attack performance is listed in Table 3, we can observe that GAA outperforms SVAE by 0.5% and 0.4% on the CFC and ResNet18 model attack, from which it is seen that GAA produces a slightly higher accuracy than SVAE when attacking CFC and ResNet18, yielding lower influence on the model performance. Since the data of MNIST is relatively simple, the transformation from the original images to the adversarial examples is easy to be finished. We also plot the generated examples by GAA in different iterations of training, as illustrated in Fig. 7, which shows consistent appearances between the generated examples and the targets.

TABLE 4
The Attack Success Rate of Different G-Structures on MNIST and CIFAR-10

| | MNIST | | | CIFAR-10 | | |
|---|---|---|---|---|---|---|
| | FC | CFC | ResNet18 | CFC | VGG16 | ResNet50 |
| $G_1$ | 96.5% | **97.8%** | **98.3%** | 85.7% | 92.7% | 91.1% |
| $G_2$ | **96.9%** | 97.6% | **98.3%** | **86.7%** | 92.8% | 90.7% |
| $G_3$ | 76.4% | 79.3% | 73.4% | 86.8% | **93.2%** | **91.2%** |

*The structures contain 8, 12, and 16 layers, which are denoted as $G_1$, $G_2$, and $G_3$, respectively.*

TABLE 5
The Attack Success Rate of Different G-Structures on ImageNet

| | VGG19 | ResNet101 | InceptionV3 | DenseNet121 |
|---|---|---|---|---|
| $G_{V13}$ | 71.5% | 70.6% | **76.2%** | 75.3% |
| $G_{V16}$ | 75.6% | **75.7%** | 74.5% | **77.8%** |
| $G_{V19}$ | **75.7%** | 75.4% | 74.5% | 77.7% |

*The structures contain VGG13, VGG16, and VGG19 as $G_enc$, which are denoted as $G_{V13}$, $G_{V16}$, and $G_{V19}$, respectively, while the layer numbers in $G_dec$ are the same as $G_enc$.*

### 5.3.2 Results on CIFAR-10

In this dataset, CFC, VGG16, ResNet50, and DenseNet121 are selected as the attacked models. Here, we conduct the experiment of within-dataset attack, i.e., both the original images and the target images coming from CIFAR-10. Fig. 8 shows the visual result of the attack. The comparison results are listed in Table 6, which shows GAA is only 0.1% lower than SVAE when attacking ResNet50. While attacking other models, the attack success rate of GAA is 2% to 4% higher than SVAE, which indicates that GAA produces noticeable improvement compared with SVAE.

### 5.3.3 Results on ImageNet

In this experiment, we attack the models including ResNet50, InceptionV3, ResNet101, DenseNet121, and EfficientNetB0. The accuracy is computed as the top-1 performance and the results are reported in Table 7. We can see that GAA is significantly better than SVAE when attacking large-scale images. The attack success rate of GAA is 2%, 4.3%, 2.3%, and 2.5% higher than SVAE on ResNet50, ResNet101, DenseNet121 and EfficientNetB0, respectively.

### 5.3.4 Results on CelebA

The experiment on CelebA focuses on how the Type I attack affects the face recognition rate. We follow the settings of [33] and use FaceNet [54] trained on CelebA to perform face recognition. FaceNet verifies two images to be the same person if the face feature distance is smaller than a threshold (e.g. 1.06). To attack FaceNet, both SVAE and GAA try to change the gender of the character in an image but fool FaceNet to make the original recognition. Alternatively, StyleGAN and GAA can generate a face image without the gender constraint, where only is the identity required to be changed. We make comparisons on these two cases, and obtain the results listed in Table 8 which validates the superiority of GAA. Several generated examples are plotted in Fig. 9. It is interesting that the examples by GAA have lower feature distances to the original images than the competitors, and StyleGAN may produce non-face images even through the other image exhibits high-quality details.

While the above results validate the Type I attack performance, we also note that the adversarial examples generated by GAA can be used to perform Type II attack, i.e., using $x'$ to attack $f$ with respect to $x^t$. Due to page limitation, more high-quality results can be found in the supplementary material where we quantify the perturbation, showing that the perturbation is imperceptible as required by Type II attack. Hence, it may be possible to optimize both Type I and Type II attacks simultaneously, which could be an interesting topic.

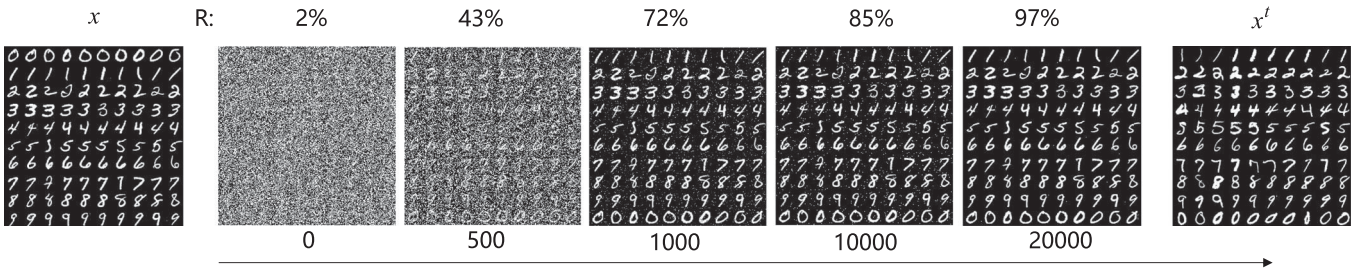



Fig. 7. Illustration of the generated MNIST examples at different iterations. The $x$ column plots the original images and the $x^t$ column plots the target images. The recognition rates by $R$ at different iterations are marked on the top of the images.

### 5.4 Outlier Attack

We identify an example as an outlier to the attacked model $f$ if it is selected from a dataset on which $f$ is not optimized. Regarding this, it is challenging to perform the cross-dataset attack, which states that the original images come from CIFAR-10 while the target images are selected from the Comic Avatar dataset[11]. This helps us to test whether the attack is successful when using an image of different styles. The results are presented in Table 9, where we see much better accuracies of GAA than SVAE. Several generated adversarial examples are exhibited in Fig. 10, which illustrates that GAA produces clean images but SVAE generates noisy images. All those images are nevertheless labelled as the labels of the original images by the model ResNet50. This experiment demonstrates the attackability of the deep models by external data with different styles. To improve model robustness, it would be encouraged to consider the resistance to style variances in a robust learning process.

### 5.5 Transferability Analyses

In this part, we examine the influence of the loss functions (i.e., $L_y = J(y, f(x'))$ and $L_f = \|f_e(x) - f_e(x')\|^2$) on the adversarial examples, which relates to Section 4.1. The baseline is selected as SVAE. The experiment is conducted on ImageNet, with the results shown in Table 10. $\text{GAA}_y$ indicates the model using $L_y$, while $\text{GAA}_f$ denotes the model using $L_f$. We generate the adversarial examples based on ResNet20, ResNet101, and InceptionV3, and use other models for testing the transferability. It can be seen from Table 10 that the adversarial examples generated by $\text{GAA}_f$ possess improved transferability over the others. In contrast, the adversarial examples generated by $\text{GAA}_y$ and SVAE produce limited success rates on attacking different black-box models. Similar results could be observed from Fig. 12. In addition, as shown in Fig. 11, we use Grad-CAM to observe the receptive fields that the model pays attention to when extracting example features. It can be seen that the receptive fields are extremely similar when extracting features from the original image and its corresponding adversarial example. However, the receptive fields of the adversarial example and target image are completely different, although they are very similar. The $L_f$ imposes the transferability on the feature space instead of the label space, which suggests that the feature similarity is a key factor in generating adversarial examples. In the following context, we use GAA to denote the specific $\text{GAA}_f$ unless otherwise noted.

11. https://github.com/chenyuntc/pytorch-book

### 5.6 Attacks Versus Defense Models

To explore whether the existing Type II adversarial training defense models have a defensive effect on the Type I adversarial examples generated by GAA, we employ three adversa-


Fig. 8. On the CIFAR-10 data, we successfully converted the dog category image into the cat category and let the classifier think that $x'$ is still a dog.

TABLE 6
Attack Performance on CIFAR-10 While Using CIFAR-10 as the Source of Target Images

| | GAA | SVAE | ORIGINAL |
|---|---|---|---|
| CFC | **86.7%** | 85.6% | 90.6% |
| VGG16 | **93.2%** | 89.5% | 94.3% |
| ResNet50 | 91.2% | **91.3%** | 92.7% |
| DenseNet121 | **93.6%** | 89.9% | 95.3% |

*The GAA and SVAE columns represent the attack's success rate, and the ORIGINAL column represents the accuracy of the clean image on different models.*

TABLE 7
Attack Performance on ImageNet

| | GAA | SVAE | ORIGINAL |
|---|---|---|---|
| ResNet50 | **75.2%** | 73.2% | 76.0% |
| InceptionV3 | 73.9% | **74.6%** | 78.8% |
| ResNet101 | **76.8%** | 72.5% | 80.9% |
| DenseNet121 | **77.5%** | 75.3% | 77.9% |
| EfficientNetB0 | **76.2%** | 74.7% | 85.0% |

*The GAA and SVAE columns represent the attack's success rate, and the ORIGINAL column represents the accuracy of the clean image on different models.*

TABLE 8
Attack Performance on CelebA

| | Acc | | Acc |
|---|---|---|---|
| GAA | **71.2%** | GAA | **85.6%** |
| SVAE | 69.3% | StyleGAN | 69.3% |

*The left comparison considers changing the gender of $x$, while the right comparison considers changing the identity of $x$.*

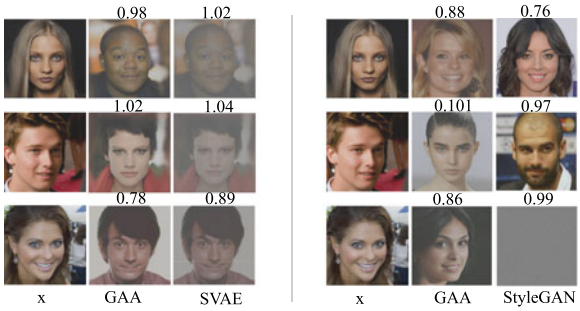



Fig. 9. Illustration of the adversarial examples generated by SVAE, StyleGAN, and GAA. The feature distances between $x$ and the generated examples are marked on top of the images.

TABLE 9
Attack Performance on CIFAR-10 While Using Comic Avatar as the Source of Target Images

| | GAA | SVAE | ORIGINAL |
|---|---|---|---|
| CFC | **63.7%** | 53.6% | 90.6% |
| VGG16 | **76.8%** | 70.2% | 94.3% |
| ResNet50 | **70.5%** | 68.9% | 92.7% |
| DenseNet121 | **68.5%** | 62.5% | 95.3% |

*The GAA and SVAE columns represent the attack's success rate, and the ORIGINAL column represents the accuracy of the clean image in models such as VGG16.*

rially trained networks in [37], including ens3-adv-Inception-v3($Inc-v3_{ens3}$), ens4-adv-Inception-v3 ($Inc-v3_{ens4}$), and ens-adv-Inception-ResNet-v2($IncRes-v2_{ens}$). Note that the

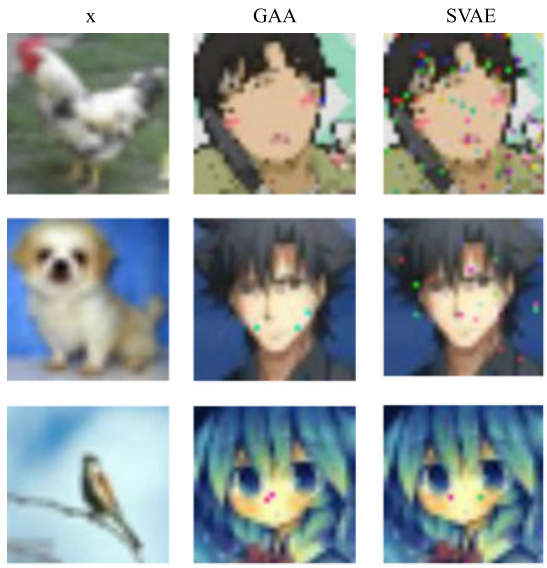



Fig. 10. Illustration of the adversarial examples generated by SVAE and GAA in the case of cross-dataset attack.

experiment is under a black-box setting. As shown in Table 11, we notice that these defense models for Type II attacks are difficult to resist Type I attacks.

Inspired by Cihang Xie [55], we expect to improve the robustness of the model by using the Type I adversarial examples during training. At this regard, the adversarial examples generated by SVAE and GAA are used for adversarial training of ResNet50, VGG16, and DenseNet121 on CIFAR-10 and ImageNet. The resultant models are denoted as $\text{ResNet50}_R$, $\text{VGG16}_R$, and $\text{DenseNet121}_R$, respectively. The training details are as follows, the initial value of the learning rate is 0.02. We set learning rate decay to 0.0001 as the standard scheme in Keras, the batch-size is 256 and 64 respectively, and the label of the adversarial examples $x'$ will be set to the corresponding target class $x^t$. The whole training process stops when it is close to overfitting. The training accuracy is shown in Table 12. We change the target

TABLE 10
The Source Model is a White-Box Model That Generates Adversarial Examples, and the Others are Black-Box Models That Test Adversarial Examples

| Source model | Attack | ResNet50 | ResNet101 | InceptionV3 | DenseNet121 | EfficientNetB0 |
|---|---|---|---|---|---|---|
| ResNet50 | $\text{GAA}_f$ | 75.2%* | **71.2%** | **70.5%** | 69.3% | **70.3%** |
| | $\text{GAA}_y$ | **76.6%*** | 50.3% | 3.2% | 43.3% | 2.9% |
| | SVAE | 73.2%* | 55.3% | 6.5% | 12.3% | 10.2% |
| ResNet101 | $\text{GAA}_f$ | **70.9%** | **76.8%*** | **71.5%** | 70.1% | **69.5%** |
| | $\text{GAA}_y$ | 45.6% | 75.7%* | 5.6% | 41.9% | 1.9% |
| | SVAE | 51.2% | 72.5%* | 7.2% | 15.6% | 11.8% |
| InceptionV3 | $\text{GAA}_f$ | **69.8%** | 69.5% | 73.9%* | **70.2%** | **67.6%** |
| | $\text{GAA}_y$ | 5.6% | 2.3% | 74.5%* | 33.2% | 19.7% |
| | SVAE | 13.5% | 4.3% | **74.6%*** | 24.7% | 14.5% |
| DenseNet121 | $\text{GAA}_f$ | **70.3%** | **72.6%** | **72.5%** | 77.5%* | **69.8%** |
| | $\text{GAA}_y$ | 47.8% | 51.2% | 33.9% | **77.8%*** | 19.9% |
| | SVAE | 23.6% | 35.8% | 18.7% | 75.3%* | 23.5% |
| EfficientNetB0 | $\text{GAA}_f$ | **72.1%** | **71.8%** | **71.3%** | **72.8%** | 76.2%* |
| | $\text{GAA}_y$ | 15.5% | 23.4% | 34.5% | 5.6% | **77.1%*** |
| | SVAE | 13.9% | 24.5% | 10.8% | 13.4% | 74.7%* |

*The data in the table all represent the success rate of the adversarial sample's attack on the model. $*$ means the attack success rate of the source model.*

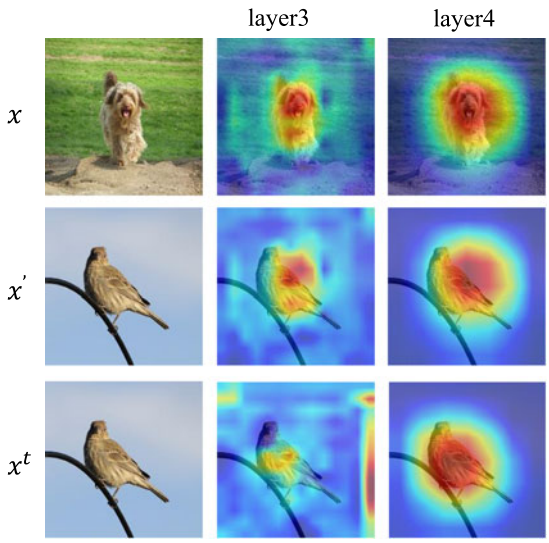



Fig. 11. There are three rows of images, which are clean image $x$, adversarial example $x'$ and target image $x^t$. We leverage Grad-CAM to visualize the layers3 and 4 of ResNet101.

TABLE 11
The Attack Success Rate of GAA on Attacking Type II Defense Methods

| Source model | $Inc-v3_{ens3}$ | $Inc-v3_{ens4}$ | $IncRes-v2_{ens}$ |
|---|---|---|---|
| InceptionV3 | 72.6% | 71.5% | 69.7% |
| ResNet50 | 70.1% | 69.9% | 69.5% |

*We use the source models including InceptionV3 and ResNet50 for generating the adversarial examples.*

TABLE 12
The Training Accuracy of ResNet50, VGG16, and DenseNet121 by Feeding an Equal Number of Adversarial Examples and Clean Images During Adversarial Training

| Dataset | $ResNet50_R$ | $VGG16_R$ | $DenseNet121_R$ |
|---|---|---|---|
| CIFAR-10 | 65.6% | 61.5% | 67.2% |
| IamgeNet | 65.3% | 62.4% | 65.6% |

category of the original image and generate new adversarial examples to perform white-box attacks on $ResNet50_R$, $VGG16_R$ and $DenseNet121_R$. The attack effect is shown in the Table 13.

As seen, it is difficult to improve the defense ability against the Type I attacks through adversarial training. In addition, we also perform black-box attack on $ResNet50_R$, $VGG16_R$ and $DenseNet121_R$. Specifically, we reset the target category of the original image to $y^t = (y+2)\%C$, and generate adversarial examples based on ResNet50, VGG16, and DenseNet121. The examples are input to $ResNet50_R$, $VGG16_R$ and $DenseNet121_R$. The test results are illustrated in Table 14.

Through the above experiments, we conclude that it is not feasible to improve the model robustness against Type I attack through adversarial training. The reason is that the Type II attack limits the perturbation, whereas the Type I attack tries to maximize the perturbation. Hence, models are challenging to fit adversarial examples and clean images. By definition, the Type I attack could generate examples that poison the training process, yielding problematic models, whereas the Type II attack could be used to improve the model robustness.

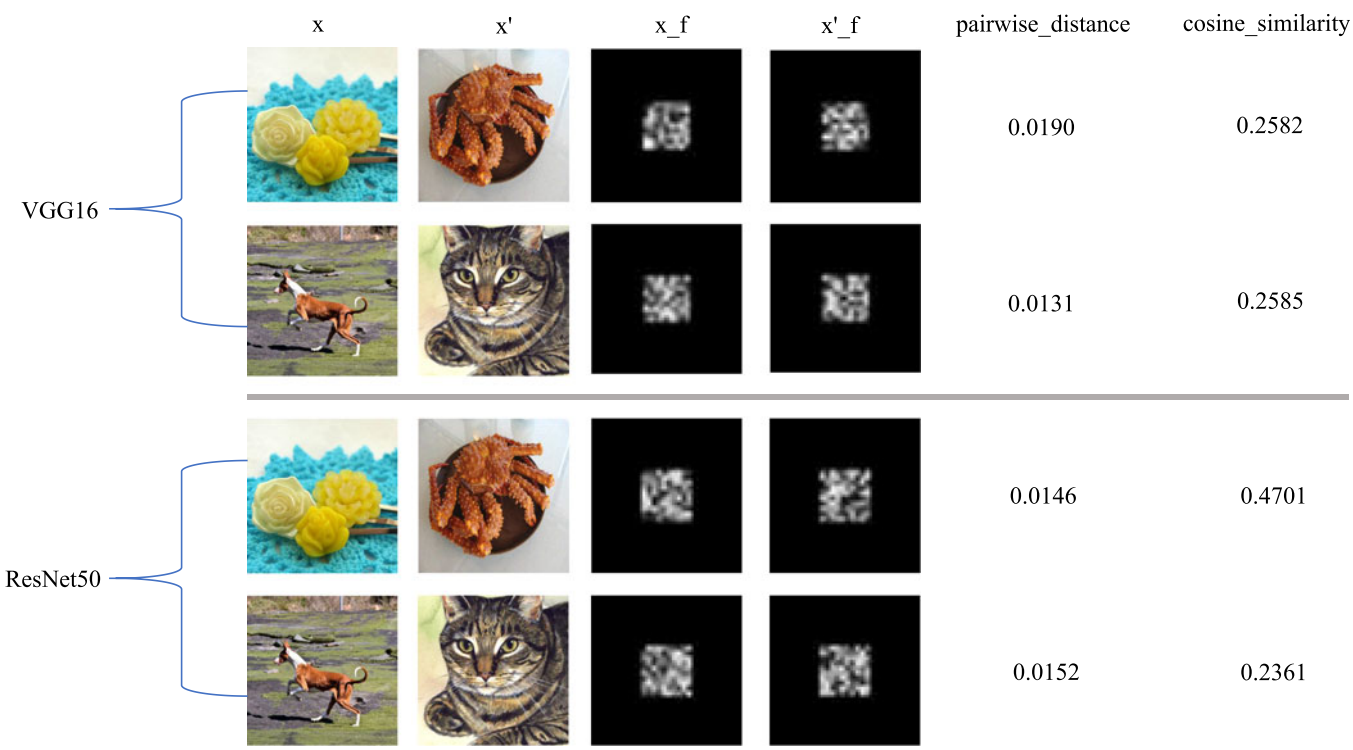



Fig. 12. Illustration of the transferability. We use the last convolutional layer of VGG16 and ResNet50 to detect the feature similarity between the original image ($x$) and the adversarial example ($x'$) generated by GAA. $x_f$ represents the deep features of $x$. $x'_f$ represents the deep features of $x'$. The pairwise_distance is the average of the pixel-wise Euclidean distance, and the cosine_similarity is the average of the cosine similarity.

TABLE 13
The Attack Success Rate of GAA and SVAE on Attacking $\text{ResNet50}_R$, $\text{VGG16}_R$, and $\text{DenseNet121}_R$

| Dataset | Source model | SVAE | GAA |
|---|---|---|---|
| CIFAR-10 | $\text{ResNet50}_R$ | 64.8% | **65.4%** |
| | $\text{VGG16}_R$ | 60.5% | **61.3%** |
| | $\text{DenseNet121}_R$ | **67.1%** | 66.8% |
| IamgeNet | $\text{ResNet50}_R$ | 63.2% | **64.5%** |
| | $\text{VGG16}_R$ | 60.7% | **61.9%** |
| | $\text{DenseNet121}_R$ | 62.7% | **65.5%** |

TABLE 14
GAA Black Box Attacks on $\text{ResNet50}_R$, $\text{VGG16}_R$ and $\text{DenseNet121}_R$

| Dataset | Source model | $\text{ResNet50}_R$ | $\text{VGG16}_R$ | $\text{DenseNet121}_R$ |
|---|---|---|---|---|
| CIFAR-10 | ResNet50 | 64.8% | 60.6% | 66.7% |
| | VGG16 | 63.5% | 61.1% | 66.3% |
| | DenseNet121 | 64.3% | 60.9% | 67.1% |
| IamgeNet | ResNet50 | 65.1% | 64.4% | 65.3% |
| | VGG16 | 63.9% | 61.5% | 63.4% |
| | DenseNet121 | 64.8% | 62.1% | 65.9% |

TABLE 15
The Running Time of Generating an Adversarial Example

| | MNIST | CIFAR-10 | ImageNet |
|---|---|---|---|
| GAA | 0.75 s | 0.77 s | 8.37 s |
| SVAE | 1.96 s | 2.16 s | 506.58 s |

### 5.7 Efficiency Analyses

We note that the proposed GAA requires only a single forward pass of the generator when making an adversarial example, whereas SVAE conducts an iterative optimization process. Here, we compare the time-consuming generation of adversarial examples on the same datasets (MNIST, CIFAR-10 and ImageNet) with the identical experimental platform, and calculate the time from the first to the last adversarial example generation. A comparison on the time cost by SVAE and GAA is listed in Table 15, which verifies that GAA has higher efficiency than SVAE, especially when the images have large sizes, e.g., in ImageNet.

## 6 Conclusion

The Type I attack views the ambiguity on different classes as the weakness of a deep model. Adversarial examples can be generated via transformation from the original images to those located in the ambiguity region. This motivates us to propose a novel Type I attack method, called generative adversarial attack. Based on AC-GAN, we develop a framework that employs the to-be-attacked model to constrain the learning process of the generator. A specialised architecture of the generator is designed by involving randomness. Extensive experiments demonstrate the effectiveness of the proposed method and furthermore, the efficiency is much higher than the existing optimization-based methods.

Notably, the defense experiments tell us that the adversarial examples produced by GAA successfully deceive the existing defense models including the Type II defense methods. Moreover, adversarial training with the Type I adversarial examples is not a feasible way to improve the model robustness. Generally, the Type I adversarial attack could be used to poison the model training process, resulting in weak deep models, while the Type II attack is helpful for enhancing the models.

## References


[1] K. Demuynck and F. Triefenbach, "Porting concepts from DNNs back to GMMs," in *Proc. Workshop Autom. Speech Recognit. Understanding*, 2013, pp. 356–361.

[2] K. Thangthai, R. W. Harvey, S. J. Cox, and B. Theobald, "Improving lip-reading performance for robust audiovisual speech recognition using DNNs," in *Proc. Conf. Auditory-Vis. Speech Process.*, 2015, pp. 127–131.

[3] J. Liu and J. Park, ""Seeing is not always believing': Detecting perception error attacks against autonomous vehicles," *IEEE Trans. Dependable Secur. Comput.*, vol. 18, no. 5, pp. 2209–2223, Sep./Oct. 2021.

[4] Z. Wei, D. Lee, B. E. Nelson, and J. K. Archibald, "Hardware-friendly vision algorithms for embedded obstacle detection applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 11, pp. 1577–1589, Nov. 2010.

[5] X. Li *et al.*, "Sustainable ensemble learning driving intrusion detection model," *IEEE Trans. Dependable Secur. Comput.*, vol. 18, no. 4, pp. 1591–1604, Jul./Aug. 2021.

[6] N. Mohammed, H. Otrok, L. Wang, M. Debbabi, and P. Bhattacharya, "Mechanism design-based secure leader election model for intrusion detection in MANET," *IEEE Trans. Dependable Secur. Comput.*, vol. 8, no. 1, pp. 89–103, Jan./Feb. 2011.

[7] Y. Li, Y. Li, K. Xu, Q. Yan, and R. H. Deng, "Empirical study of face authentication systems under OSNFD attacks," *IEEE Trans. Dependable Secur. Comput.*, vol. 15, no. 2, pp. 231–245, Mar./Apr. 2018.

[8] X. Zou, P. Xiao, J. Wang, L. Yan, S. Zhong, and Y. Wu, "Towards unconstrained facial landmark detection robust to diverse cropping manners," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 5, pp. 2070–2075, May 2021.

[9] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. Int. Conf. Learn. Representations*, 2015.

[10] D. Wang, C. Li, S. Wen, S. Nepal, and Y. Xiang, "Man-in-the-middle attacks against machine learning classifiers via malicious generative models," *IEEE Trans. Dependable Secur. Comput.*, vol. 18, no. 5, pp. 2074–2087, Sep./Oct. 2021.

[11] A. M. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 427–436.

[12] C. Sun, S. Chen, J. Cai, and X. Huang, "Type I attack for generative models," in *Proc. Int. Conf. Image Process.*, 2020, pp. 593–597.

[13] Y. Shen, P. Luo, J. Yan, X. Wang, and X. Tang, "Faceid-GAN: Learning a symmetry three-player GAN for identity-preserving face synthesis," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 821–830.

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

[15] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Proc. 5th Int. Conf. Learn. Representations*, 2017.

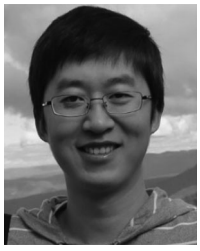
[16] K. Eykholt *et al.*, "Robust physical-world attacks on deep learning visual classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1625–1634.

[17] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," *Proc. 35th Int. Conf. Mach. Learn.*, J. G. Dy and A. Krause, Eds., vol. 80, pp. 284–293, 2018.

[18] J. Tu *et al.*, "Physically realizable adversarial examples for lidar object detection," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 13716–13725.

[19] A. Braunegg *et al.*, "APRICOT: A dataset of physical adversarial attacks on object detection," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 35–50.
[20] A. Saha, A. Subramanya, K. Patil, and H. Pirsiavash, "Role of spatial context in adversarial robustness for object detection," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 3403–3412.
[21] T. Vu, H. Jain, M. Bucher, M. Cord, and P. Pérez, "ADVENT: Adversarial entropy minimization for domain adaptation in semantic segmentation," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2517–2526.
[22] L. Samson, N. van Noord, O. Booij, M. Hofmann, E. Gavves, and M. Ghafoorian, "I bet you are wrong: Gambling adversarial networks for structured semantic segmentation," in *Proc. Int. Conf. Comput. Vis. Workshops*, 2019, pp. 951–960.
[23] R. Xiao, B. Deng, F. Li, M. YuLiu, and D. Song, "Characterizing adversarial examples based on spatial consistency information for semantic segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 217–234.
[24] H. Xu *et al.*, "Adversarial attacks and defenses in images, graphs and text: A review," *Int. J. Autom. Comput.*, vol. 17, pp. 151–178, 2020.
[25] Y. Xie, Z. Gu, B. Zhu, L. Wang, W. Han, and L. Yin, "Adversarial examples for chinese text classification," in *Proc. Int. Conf. Data Sci. Cyberspace*, 2020, pp. 238–245.
[26] J. X. Morris, E. Lifland, J. Y. Yoo, J. Grigsby, D. Jin, and Y. Qi, "TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP," in *Proc. Conf. Empir. Methods Natural Lang. Process. Syst. Demonstrations*, Q. Liu and D. Schlangen, Eds., 2020, pp. 119–126.
[27] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2574–2582.
[28] Y. Dong *et al.*, "Boosting adversarial attacks with momentum," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9185–9193.
[29] I. J. Goodfellow *et al.*, "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, 2020.
[30] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. Symp. Secur. Privacy*, 2017, pp. 39–57.
[31] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 86–94.
[32] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *Trans. Evol. Comput.*, vol. 23, no. 5, pp. 828–841, 2019.
[33] S. Tang, X. Huang, M. Chen, and J. Yang, "Adversarial attack type i:cheat classifiers by significant changes," *Comput. Res. Repository*, 2018, pp. 1100–1109.
[34] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length and helmholtz free energy," in *Proc. Adv. Neural Inf. Process. Syst.*, 1993, pp. 3–10.
[35] N. O'Donoughue and J. M. F. Moura, "The complex double gaussian distribution," in *Proc. Int. Conf. Acoust. Speech Signal Process.*, 2012, pp. 3593–3596.
[36] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *Proc. 5th Int. Conf. Learn. Representations*, 2017.
[37] F. Tramèr, A. Kurakin, N. Papernot, I. J. Goodfellow, D. Boneh, and P. D. McDaniel, "Ensemble adversarial training: Attacks and defenses," in *Proc. Int. Conf. Learn. Representations*, 2018.
[38] C. Guo, M. Rana, M. Cissé, and L. van der Maaten, "Countering adversarial images using input transformations," in *Proc. 6th Int. Conf. Learn. Representations*, 2018.
[39] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. L. Yuille, "Mitigating adversarial effects through randomization," in *Proc. 6th Int. Conf. Learn. Representations*, 2018.
[40] G. S. Dhillon *et al.*, "Stochastic activation pruning for robust adversarial defense," in *Proc. 6th Int. Conf. Learn. Representations*, 2018.
[41] A. Prakash, N. Moran, S. Garber, A. DiLillo, and J. A. Storer, "Deflecting adversarial attacks with pixel deflection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8571–8580.
[42] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, " Pixeldefend: Leveraging generative models to understand and defend against adversarial examples," in *Proc. Int. Conf. Learn. Representations*, 2018.
[43] Meng and H. Chen, "MagNet: A two-pronged defense against adversarial examples," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2017, pp. 135–147.
[44] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-GAN: Protecting classifiers against adversarial attacks using generative models," in *Proc. Int. Conf. Learn. Representations*, 2018.
[45] B. Liang, H. Li, M. Su, X. Li, W. Shi, and X. Wang, "Detecting adversarial image examples in deep neural networks with adaptive noise reduction," *IEEE Trans. Dependable Secur. Comput.*, vol. 18, no. 1, pp. 72–85, Jan./Feb. 2021.
[46] S. Jandial, P. Mangla, S. Varshney, and V. Balasubramanian, "Advgan : Harnessing latent layers for adversary generation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops*, 2019, pp. 2045–2048.
[47] Z. Zhao, D. Dua, and S. Singh, "Generating natural adversarial examples," in *Proc. 6th Int. Conf. Learn. Representations*, 2018.
[48] X. Liu and C. Hsieh, "Rob-GAN: Generator, discriminator, and adversarial attacker," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 11234–11243.
[49] G. Jin, S. Shen, D. Zhang, F. Dai, and Y. Zhang, "APE-GAN: Adversarial perturbation elimination with GAN," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2019, pp. 3842–3846.
[50] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-GAN: Protecting classifiers against adversarial attacks using generative models," in *Proc. 6th Int. Conf. Learn. Representations*, 2018.
[51] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier GANs," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2642–2651.
[52] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[53] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015.
[54] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 815–823.
[55] C. Xie *et al.*, "Improving transferability of adversarial examples with input diversity," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2730–2739.

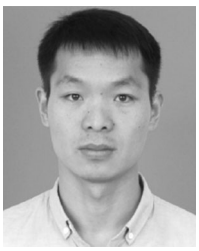
**Shenghong He** He is currently working toward the master's degree with the School of Software at Yunnan University. His main research direction is adversarial networks, adversarial attacks, and image classification.



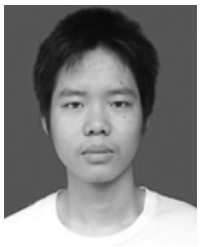
**Ruxin Wang** (Member, IEEE) received the BEng degree from Xidian University, the MSc degree from the Huazhong University of Science and Technology, and the PhD degree from the University of Technology Sydney. He is currently an associate professor with the National Pilot School of Software, Yunnan University, Kunming, China. His research interests include image restoration, deep learning, and computer vision. He focuses on the topic of image synthesis by using both discriminative models and generative models. He has authored and coauthored 20+ research papers including *IEEE Transactions on Neural Networks and Learning Systems*, *IEEE Transactions on Image Processing*, *IEEE Transactions on Cybernetics*, ICCV, and AAAI. He has received "the 1000 Talents Plan for Young Talents of Yunnan Province" award.

**Tongliang Liu** (Senior Member, IEEE) is currently a lecturer with the School of Computer Science and the faculty of engineering, and a core member with the UBTECH Sydney AI Centre, The University of Sydney. His research interests include machine learning, computer vision, and data mining. He has authored and coauthored more than 60 research papers including *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *IEEE Transactions on Neural Networks and Learning Systems*, *the IEEE Transactions on Image Processing*, ICML, NeurIPS, AAAI, IJCAI, CVPR, ECCV, KDD, and ICME, with best paper awards, e.g., the 2019 ICME Best Paper Award. He is a recipient of DECRA from Australian Research Council.

**Chao Yi** (Member, IEEE) received the MS and PhD degrees in computer science and technology from the Yunnan University in 2003 and 2009. He is currently a associate professor with the National Piolet School of Software,Yunnan University, China. His current research interests include artificial intelligence and Big Data computing.



**Xin Jin** (Member, IEEE) received the BS degree in electronics and information engineering from Henan Normal University, Xinxiang, China, in 2013, and the PhD degree in communication and information systems from Yunnan University, Kunming, China, in 2018. He is currently a associate professor with the School of Software, Yunnan University. His current research interests include pulse coupled neuralnetwork theory and its applications, image processing, optimization algorithms, and bioinformatics.



**Renyang Liu** received the BE degree in computer science from the Northwest normal University in 2017, He is currently working toward the PhD degree with the School of Information Science and Engineering, Yunnan University, Kunming, China. His current research interest includes deep learning, adversarial attack and generative models.

**Wei Zhou** (Member, IEEE) received the PhD degree from the University of Chinese Academy of Sciences. He is currently a full professor with the Software School, Yunnan University. His current research interests include the distributed data intensive computing and bio-informatics. He is currently a fellow of the China Communications Society, a member of the Yunnan Communications Institute, and a member of the Bioinformatics Group of the Chinese Computer Society. He won the Wu Daguan Outstanding Teacher Award of Yunnan University in 2016, and was selected into the Youth Talent Program of Yunnan University in 2017. Hosted a number of National Natural Science Foundation projects.