IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 5, 2020, accepted November 6, 2020, date of publication November 11, 2020,
date of current version November 24, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3037329

A Two-Stage Generative Adversarial Networks With Semantic Content Constraints for Adversarial Example Generation

JIANYI LIU[1], YU TIAN[1], RU ZHANG[1], (Member, IEEE), YOUQIANG SUN[1], AND CHAN WANG[2]
[1]Beijing University of Posts and Telecommunications, Beijing 100876, China
[2]State Grid Information and Telecommunication Branch, Beijing 100761, China

Corresponding author: Ru Zhang (zhangru@bupt.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2019QY2202; and in part by the National Natural Science Foundation of China under Grant U1836108, Grant U1936216, Grant 62002197, and Grant 62001038.

ABSTRACT Deep neural networks (DNNs) have achieved great success in various applications due to their strong expressive power. However, recent studies have shown that DNNs are vulnerable to adversarial examples, and these manipulated instances can mislead DNN into making false predictions. The existing methods of generating adversarial examples include pixel-level perturbation or spatial transformation of images, which cannot consider concurrently with the semantic quality of adversarial examples or success rate of attack. These methods are computationally bulky and slow to generate the adversarial examples. To solve this kind of issue, a two-stage generative adversarial networks (TSGAN) with semantic content constraints is proposed in this paper. The first-stage uses the original example dataset to train generator G, which can help the generator learn the distribution of real examples. Then, the example semantic quality constraint loss function, the adversarial loss function and the distance loss function are adopted in the second-stage, so that the generator G can continue to learn to search the distribution of the adversarial examples, and train the new generator G_{adv}. The adversarial examples generated by generator G_{adv} are better fit the distribution of real examples, and have targeted black-box attack capability. The experiments show that the adversarial examples generated by TSGAN can achieve the success rate of attack at 98.40% in target model, 29.40% success rate in defense-oriented model. And 77.58% success rate is obtained in the transfer test attack. The results show that the adversarial examples generated by the proposed model, which has a highly attack success rate and more difficult to defense. Meanwhile, the improved adversarial examples have stronger transfer ability than the existing models. The proposed model can effectively reduce the expression of target category features of the adversarial examples, and the generated adversarial examples have better semantic quality than others.

INDEX TERMS Generative adversarial networks, adversarial example attack, semantic content.

I. INTRODUCTION

In recent years, deep neural networks (DNNs) have been widely used in various scenarios and tasks, however, the security problem of DNNs also arouses people's attention constantly. Szegedy *et al.* [1] found that deep neural network models are susceptible to the adversarial examples, in particular, the effect of carefully designed adversarial examples is more obvious. Only by simply adding a small disturbance to the image, the model can make the error classification with high confidence, since then, the adversarial example attack problem was proposed. The essence of adversarial example attack problem is to generate a series of input examples of deep learning model, which cannot be detected by human eyes but can confuse model judgment by adding small perturbations. Adversarial example attack can be applied in many fields, such as target recognition, automatic driving, face recognition, intrusion detection systems (IDS) and other applications.

Goodfellow *et al.* [2] attempted to test the input adversarial examples by photos taken with mobile phones, and were able to successfully realize the adversarial example attacks

The associate editor coordinating the review of this manuscript and approving it for publication was Kathiravan Srinivasan.

against the classifier. Sharif *et al.* [3] designed a spectacle frame and made it into a real object, and the wearer of the frame can successfully confuse the facial recognition software. Liu *et al.* [4] achieved confusion in the physical world for the Faster Region-CNN and YOLO 9000 target detector. Eykholt *et al.* [5] used adversarial stickers next to the sign of STOP, which enabled the YOLO detector to fail to detect the STOP sign in almost all frames of the video.

The traditional adversarial example attacks such as the earliest method L-BFGS (the BFGS, which proposed by Broyden, Fletcher, Goldfarb and Shanno, with limited memory called L-BFGS), the most representative method FGSM (Fast Gradient Sign Method), and these various variants. These kinds of methods generate the adversarial examples by accessing the gradient information of target model and adding the calculated perturbations to the original examples. But these methods require huge computation, and most of them are white-box attack. And the specific information is needed when calculating the perturbation, therefore, the applicability of such attack methods is relatively narrow.

Nowadays, a tremendous amount works has been done to the adversarial example attack by using the deep learning, such as using the generative adversarial networks (GAN) [6] to generate the adversarial examples. The speed of adversarial example generation based on the GAN is general faster than the traditional methods. Although many different works are based on the deep learning, there are still many problems in the adversarial example attack, such as the low-quality of the generated adversarial examples, the distinct disturbances of examples, low-success rate of attack and low-transfer. To solve these problems, this paper proposed a two-stage generative adversarial network (TSGAN) attack method with semantic content constraints. TSGAN trains the adversarial example generation model twice. During the first training stage, the generation model is required to obtain the distribution of the original data, and make the distribution of the generated adversarial examples as close to the original distribution as possible. This stage can help the generation model generate the adversarial examples with high attack success rate. In the rear training stage, the additional semantic content constraints are added to help generation model improve the generation quality. Constraining the semantic information of the generated adversarial examples by using the content feature extraction network not only improve the attack success rate to the image classifier, but also improve the quality of the generated adversarial examples and reduce the human perception. The well-trained generation model can be got after two-stage training.

II. RELATED WORK

The adversarial example attack refers to a given machine learning model f and the original examples $x \in y$, among the examples, y is the category of the examples x belongs, then generate the adversarial examples x'. The adversarial examples should be sufficiently similar to the original examples. In traditional attack methods, the norm distance L_2 or other norm distance L is used as the measurement of perceived example similarity [7]. The generated adversarial examples can make the classifier model f deviate from the original classification result $f(x') = y$, or make the classifier model classify it into the specified category $f(x') = y'$. The adversarial example attack's purpose is to obfuscate the output of the classifier model. And the attack can be divided into three categories: one-step attack, iteration attack and generation attack.

Where the classical algorithms of the adversarial example attack are one-step attack, iteration attack. The L-BFGS attack was first found by Szegedy *et al.* [1] in 2013, when a small perturbation is added to the input examples, the result of prediction model can be easily changed. Although this method can generate the adversarial examples stably and effectively by linear search the manifold space, the defects of L-BFGS are high-complexity of computation and slow-generation. Goodfellow *et al.* [8] proposed a fast gradient symbol attack FGSM, which can quickly generate adversarial examples against the target model. This method requires a small amount of calculation, and only one back-propagation and one gradient calculation are required during the perturbation. However, the target cannot be specified in the FGSM attack. Subsequently, Kurakin *et al.* [2] modified the FGSM algorithm, and proposed three iteration attack algorithms: the basic-iterative algorithm BIM (Basic Iterative Method), iterative algorithm FGSM and the least possible iterative algorithm ILLC (Iteratice Least-Likely Class Method). These algorithms can achieve a better attack result than the previous algorithm FGSM. Dong *et al.* [9] added the momentum to the FGSM algorithm, and generate adversarial examples by iteration. The improved FSGM called MI-FSGM (momentum iteration of FSGM), and this method accelerates the speed of adversarial example attack, meanwhile, the MI-FSGM can achieve target-less and targeted attack. Carlini and Wagner [10] proposed C&W attack against the distillation of neural network distillation, and proved that the distillation does not substantially improve the robustness of neural network. The attack examples generated by C&W are aggressive and has small perturbations. Seyed-Mohsen *et al.* [11] proposed a Deep Fool attack, which attack generates the adversarial examples according to the decision boundary of the classifier and the minimum perturbation between the example and hyperplane. Comparing with the FGSM, the Deep Fool attack is more effective, faster and less disturbance, but it can only find the disturbance that causes the sample to cross the decision surface and unable to specify for attack.

Most of the traditional adversarial example attacks are white-box attacks, which attacks need the target model's architecture and other information. So, the white-box attack has low applicability and low transfer. The emerging researches suggest that the adversarial example attacks are broader adopted the generative adversarial networks. The inputs examples are generated by the disturbances, which are directly superimposed or generated as adversarial

examples. Compared with the traditional attacks, the adversarial examples generated by GAN have higher success rate and faster generation speed, and the examples also have higher transfer and anti-robustness. Baluja *et al.* [12] proposed an adversarial transformation network (ATN), which can generate a greater variety of adversarial examples by constraining the distance of generated adversarial example to original example and attack effect of the generated adversarial example. Inspired by the ATN, Avishek *et al.* [13] proposed a method to solve the constraint optimization problem by using the GAN. When building the network, this paper adopts the ATN as the generator. The result show that this method is effective in the generated face adversarial examples, and the generation speed is faster than the fastest traditional attack method FGSM. Zhao *et al.* [14] used an inverter to approximate a potential vector for input example. A natural adversarial example can be generated by adding perturbation to the potential vector and using the generator reproduce the example. Xiao *et al.* [15] have made an approximate standard contribution to the adversarial example attack via GAN. The authors used the least square GAN (LSGAN) as the basic structure, combined the generator G, discriminator D and target model F to the network. The generator generated disturbance is superimposed on the original example as the adversarial example, and the discriminator is responsible for distinguishing the adversarial example from the original example, and the target model is used as the direction to guide the generator. The trained generator can generate target and no target adversarial examples in batch, which achieved a very high attack success rate. Mangla *et al.* [16] improved the AdvGAN (a GAN which can generate adversarial examples called AdvGAN), and proposed AdvGAN++. The authors thought that when generated the adversarial example, the potential characteristics of the original examples should be full of use, and generated adversarial examples should also be close to the input distribution. The experiments showed that the attack success rate and anti-robustness of AdvGAN++ were superior than the standard AdvGAN. Song *et al.* [17] proposed an unrestricted method for generating adversarial examples based on the ACGAN (Auxiliary Classifier GAN). Wang *et al.* [18] proposed an anti-transfer generative adversarial network (AT-GAN), which generates unlimited adversarial examples from random noise by estimating the distribution of examples. Chen *et al.* [19] proposed FF-GAN, which adopted the mask mechanism in the generator of the GAN to solve the constraint problems. Because FF-GAN selectively disturbed the important features of the classifier, the disturbance range is reduced. Zhao *et al.* [20] proposed an unsupervised adversarial attack GAN (UAA-GAN) to solve the content-based image retrieval (CBIR), face search and person re-identification (ReID) problem, which method focused on the deep visual features of disturbed examples. Jiang *et al.* [21] proposed a cycle-uniform adversarial GAN (CycleAdvGAN), which trained two generators to adversarial examples generation and adversarial examples recovery, that could help the model training. Lin *et al.* [22] proposed IDSGAN for IDS detection system, which used black-box IDs as the target model to generate adversarial examples of flow data.

III. ADVERSARIAL EXAMPLE GENERATION OF TSGAN

The problem of adversarial example generation can be defined as: set the target model f to be attacked, and training a generator G to learn the distribution P_{adv} of the adversarial examples against the target model f. So that can generate the adversarial example $x' = G(z)$ from the random noise, and thus $f(G(z)) = y_{target}$, y_{target} is the misjudgment category of the target model that under the attack. And the distribution of the generated examples P_{adv} are as close as possible to the real examples P_{data}, that can make the generated examples similar to the originals.

In order to enhance the attack success rate and quality of the adversarial examples that generate by GAN, this paper adopts a two-stage training strategy. The first-stage is based on the original examples, during this stage, training a generator G which satisfies $P_g \approx P_{data}$, P_g is the distribution of the generated examples and P_{data} is original examples' distribution. Making the adversarial examples as realistic as possible is the purpose of this stage. The second-stage is using the black-box attack to continue training the generator. During the training process, extra adversarial loss L_{adv} and distance loss $L_{perturb}$ are added for constraint, that can help the generator G to search around the P_{data} distribution. Finally, the adversarial examples $x' = G_{adv}(z)$ generated by generator G_{adv} which are close to the distribution of the original examples $P_g \approx P_{data}$, and have the target attack ability. The target attack means that the adversarial examples which generated from the generator G_g with random noise can be classified into a specific wrong category $f(G_g(z)) = y_{target}$.

In the past, most of the previous adversarial example attack methods do not consider the semantic of example image, mostly used the superimposed disturbance or constrained by supervised learning. Because the content features represent the semantics of the image, if the content features are kept unchanged, the generated adversarial examples will become more invisible to human beings. So, in this paper, the semantic content features $x_{content}$ of the image are considered in the generation of adversarial examples, and the feature extraction network $N_{feature}$ is also used in the network. In this way, the generated adversarial examples can not only deceive the target model, but also maintain the original semantic features.

A. THE SEMANTIC CONTENT FEATURES EXTRACTION

In order to make the adversarial examples generated by generator G_{adv} satisfy the original examples' distribution P_{data}, and it's necessary to define and measure the loss of image disturbance. In the traditional and latest adversarial example attack methods, the constraints on generated examples' quality usually adopt the L_2 norm or others norm distance.

FIGURE 1. The feature visualization of VGG-16. The first color image is the original example, the remaining grayscale image are visualized feature map.

The definition of the L_2 norm is the equation (1):

$$D_{L_2} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{1}$$

The L_2 norm is calculating the difference of the pixels in two images, x_i and y_i are pixels in different images. Although the result can indicate the difference between two images on the pixel level, the L_2 norm distance cannot describe the difference between two images on the semantic level perfectly. L_2 norm is adopted in the training process, which caused a variety of obvious different content features change. G_{adv} tends to generate target category features of the image to confuse the classifier, and undetectable to humans. Therefore, L_2 norm is contrary to the premise that is hard for human to perceive in the adversarial example attack.

Inspired by the work of image style transfer [23]–[25], human's recognition depends on the content features of objects. So, to improve the features of the adversarial examples which are difficult to be perceived by human beings, this paper adopted the content feature extraction method of the image style transfer to enhance the generated examples' quality. In order to ensure that the content features of the adversarial examples x' generated by generator G_{adv} are as consistent as possible with the expression of generator G.

Suppose the example image is and the content features are $x_{content}$, and CNN (convolutional neural networks) has the ability of feature extraction, we use a feature extraction network $N_{feature}$ based on CNN to extract the content feature $x_{content}$, and adopt a content loss function $L_{content}$ to constrain the content feature during the training. This paper used the trained VGG-16 model [26] as the feature extraction network $N_{feature}$. The VGG-16 contains 16 hidden layers (13 convolution layers and 3 fully connected layers), including 5 convolution structures. And used multiple of 3×3 convolution kernels instead of the large convolution kernels, which can reduce parameters of the network and conducive to maintain the image features [27]. For CNN, the result of each convolution structure can be regarded as a feature map of the input image, so it can use the activation value of the feature map to represent the image's content feature, i and j respectively represent the j *th* feature map of the convolution structure:

$$x^{ij}_{content} = N^{ij}_{feature}(x) \tag{2}$$

In order to check the output feature map of the VGG-16 model, an image of 224×224 resolution was used for calculation. The visualization output of each convolution structure after ReLU (rectified linear unit) activation function was obtained from left to right, as shown in Fig. 1. In the Fig. 1, the four gray-scale from the left to right are the visualization of the activation value of the feature map from the first convolution structure to the fourth convolution structure. It can be found that the feature map output by the VGG-16 model has a good content representation ability for the image, and can be extracted into the content features that conform to human eye observation.

With the output of feature extraction network as the representation of image content features, we can calculate the feature map i with size $C \times H \times W$ for each picture on two pictures x_1 and x_2. Therefore, the content feature loss function $L^i_{content}$ of the two images can be defined as equation (3):

$$L^i_{content}(x_1, x_2) = \left\| \frac{N^i_{feature}(x_1) - N^i_{feature}(x_2)}{C_i \times H_i \times W} \right\|^2_2 \tag{3}$$

B. TSGAN

The WGAN-GP (Wasserstein GAN with gradient penalty) network [28] is used as the basic structure to generate the adversarial examples in this paper. Normal training and adversarial training are adopted in the training of TSGAN, which have different purpose and architecture. The two-stage are introduced in this part.

The normal training and adversarial training are sequential, the first-stage is normal training and the second-stage is adversarial training, and both normal and adversarial stage are training the generator. After the first-stage training, the generator which trained in the normal training keep both architecture and parameters intact, then put the generation model into the second-stage to continue training. Through the two-stage training, the adversarial examples generated by generator are more aggressive and better quality than the existing methods.

The part of the normal training is aiming at generating a serial of normal adversarial examples. During the training process, the generator can learn the data distribution of original examples, and the architecture of the normal training is shown in Fig. 2 (a). In this figure, the normal training part including generator G_{norm} and discriminator D.

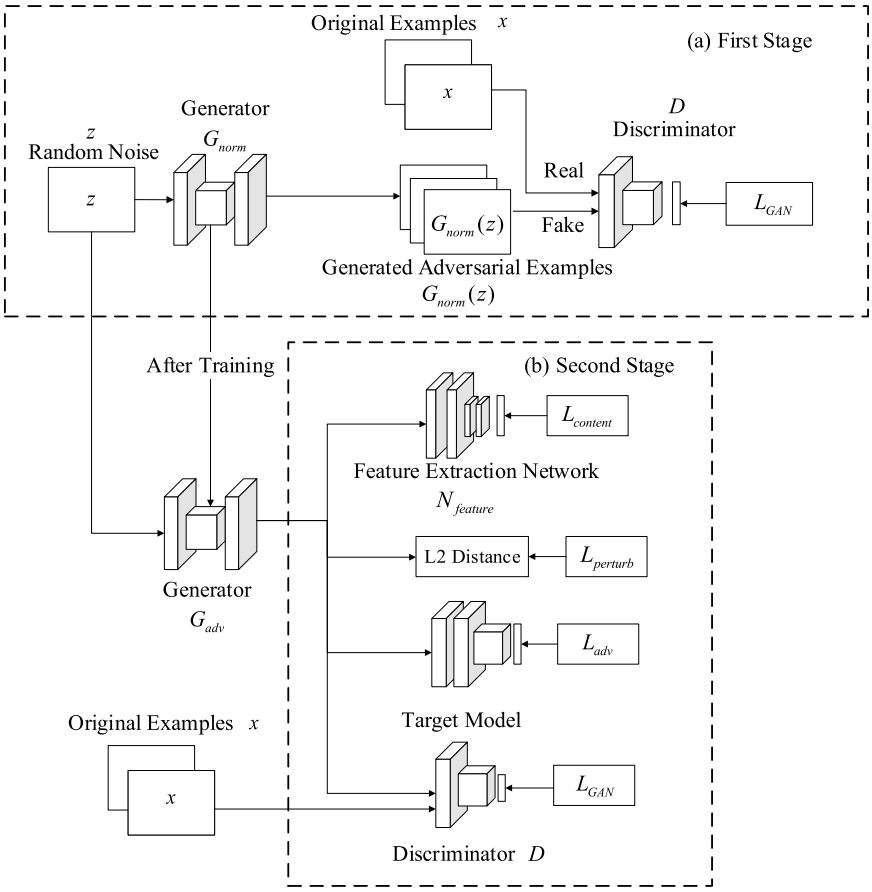

FIGURE 2. The architecture of the TSGAN. Including the first-stage (normal) training and second-stage (adversarial) training. The second-stage training is continued when the first-stage training is finished.

The generator G_{norm} is responsible for generating adversarial examples $G_{norm}(z)$ from random noise z. And the discriminator D is used for distinguishing the authenticity of the image $G_{norm}(z)$ generated by generator and the original example x, it can also help the generator produce a serial of more realistic images. The trained generator G_{norm} and the corresponding discriminator D can be obtained after training, this part of the training is based on supervised learning, and it's the basic of the next training.

The purpose of the adversarial training part is to make the generator G_{adv} learn the distribution of the adversarial examples from the random noise based on the previously trained generator. The Fig. 2 (b). shows the architecture of the adversarial training model, including generator G_{adv}, discriminator D, target model f and feature extraction network $N_{feature}$. The generator G_{norm} and discriminator D are the trained in the first-stage. In the second-stage, the generator G_{adv} keep training with four different loss functions. In addition to the loss function L_{GAN} of the WGAN-GP, the loss function L_{adv} of the target model f, the loss function $L_{perturb}$ of L_2 norm distance and the loss function $L_{content}$ of feature extraction model $N_{feature}$. Using the adversarial examples $G_{adv}(z)$ generated by generator G_{adv} as the input of the target model, the output of the adversarial part is defined as the adversarial loss $L_{adv}^{y_{target}}$, the y_{target} is the category of the attacked, $L_{adv}^{y_{target}}$ represents the distance between the category attacked of generated adversarial examples $G_{adv}(z)$ and the predicted category by target model. The definition of the $L_{adv}^{y_{target}}$ is:

$$L_{adv}^{y_{target}} = E_{zP_{G_{adv}(z)}} \log_f(G_{adv}(z), y_{target}) \tag{4}$$

Meanwhile, in order to limit the disturbance range of the adversarial examples $G_{adv}(z)$, the disturbance evaluation is defined to measure the loss of disturbance. This paper used the distance of L_2 norm as the loss function of disturbance assessment $L_{perturb}$. And the loss function is prevented the excessive disturbance, the detail equation is following:

$$L_{perturb} = \|G_{norm}(z) - G_{adv}(z)\| \tag{5}$$

The feature extraction network $N_{feature}$ is shown in Fig. 3. As for $N_{feature}$, the activation function in the third convolution structure of VGG-16 is selected to calculate the content features. In this paper, the generated adversarial examples $G_{adv}(z)$ and the normal examples $G_{norm}(z)$ should be compared, so put $G_{adv}(z)$ and $G_{norm}(z)$ as the inputs of the $N_{feature}$. Then using the MSE loss to construct the loss function

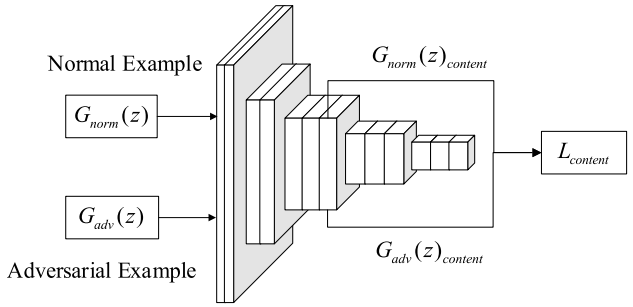

FIGURE 3. The architecture of feature extraction network in VGG-16.

of content feature $L_{content}$, the loss function which used in $N_{feature}$ is keeping the semantic content, the loss function is equation (6):

$$L_{content}(G_{norm}(z), G_{adv}(z)) = \left\| N_{feature}(G_{norm}(z)) - N_{feature}(G_{adv}(z)) \right\|_2^2 \quad (6)$$

So, the total loss function of the second-stage training is the equation (7), and the λ_1, λ_2 and λ_3 are the hyperparameters that control the different loss functions. The L_{total} is replaced the original one in generator G_{adv}.

$$L_{total} = L_{GAN} + \lambda_1 L_{adv} + \lambda_2 L_{perturb} + \lambda_3 L_{content} \quad (7)$$

The pseudocode of the two-stage training is given in the following part. In the first-stage training, the random noise is used as the input of generator, the generated adversarial examples $G_{norm}(z)$ and the original examples x are input to the discriminator D. Then initialize the generator G_{norm} and discriminator D, and only use the L_{GAN} as the loss function to train the network. The Table 1. Shows the pseudocode of the first-stage training:

The second-stage training also uses the random noise as the generator input. The generated adversarial examples $G_{adv}(z)$ and normal examples $G_{norm}(z)$ as the discriminator input, and the attack category y_{target} is given. The generator G_{norm} and discriminator D that have been trained in the first step are used instead of initialization. The loss function of discriminator keeps unchanged, and the adversarial loss function $L_{adv}^{y_{target}}$, disturbance loss function $L_{perturb}$ of generating samples and content feature loss function $L_{content}$ are added as the total loss of second-stage. And keep training the network model, the pseudocode of second-stage training is given in Table 2.

IV. THE EXPERIMENTS AND ANALYSES

A. DATASETS

The MNIST and CIFAR-10 dataset are used in this paper. MNIST data set [29] is a classic handwritten numeral data set in the field of machine learning. And 10 categories in the dataset, 0 through 9, each example is a 28×28 pixel grayscale image. CIFAR-10 dataset [30] is also a classic dataset in the field of machine learning. There are 10 categories, and each example is a 32×32 RGB image.

Several attack models and corresponding classifiers are trained for different target models, and different attack models' performance are compared in this part. Meanwhile, other attack methods are also used to compare with the proposed model.

TABLE 1. The Pseudocode of The first-stage training.

Algorithm 1: Normal training in the first-stage.

Input the random noise z, original examples x, batch size, the α, β_1, β_2 in Adam optimizer and other parameters, initialize the parameters of the generator θ_0 and the gradient punishment ω_0;
While θ not converged:
  For i in the penalty period of generator:
    For j in the batch size:
      The generator generates adversarial examples $\tilde{x} \leftarrow G_{norm}(z)$ from random noise z;
      Linear interpolation $\hat{x} \leftarrow \varepsilon x + (1-\varepsilon)\tilde{x}$ for original examples x and generated adversarial examples $\tilde{x}$;
      Calculate the gradient punishment $\left\|\nabla_{\hat{x}} D(\hat{x})\right\|_2$;
      Calculate the loss of discriminator $L_D^i = D_\omega(\tilde{x}) - D_\omega(x) + \lambda\left(\left\|\nabla_{\hat{x}} D_\omega(\hat{x})\right\| - 1\right)^2$;
    End
    Update gradient punishment $\omega \leftarrow \mathrm{Adam}\left(\nabla_{\hat{x}} 1/m \sum_{i=1}^{m} L_D^i, \omega, \alpha, \beta_1, \beta_2\right)$;
  End
  Select batch size of random noise $\{z^i\}_{i=1}^m p(z)$;
  Calculate the loss of generator $L_G = -D_\omega(G_\theta(z))$;
  Update the parameters of the generator $\theta \leftarrow \mathrm{Adam}\left(\nabla_\theta 1/m \sum_{i=1}^{m} -D_\omega\left(G_\theta(z)\right), \theta, \alpha, \beta_1, \beta_2\right)$;
End
Output the generated adversarial examples $G_{norm}(z)$.

B. THE ATTACK PERFORMANCE OF TSGAN

For the reasonableness of the experiments, the algorithms which used in this paper are all based on python, and using the PyTorch to build the deep learning network. The hardware environment of the experiment is AMD 3700X CPU, 16GB memory, and NVIDIA RTX2080 graphics card is used for accelerated training.

This section contains three experiments: attack experiment on the target model's classifier after normal training, attack experiment on the target model's classifier after adversarial training and transfer attack on different models.

1) THE SETTING OF EXPERIMENT

The purpose of this paper is to attack the trained image classifier, according to the proposed attack model, the calculation of the adversarial loss function needed the target model. As for MNIST dataset, LeNet5 adopted, and the ResNet20, ResNet32 and Wide Resnet28 [31] are used for training the CIFAR-10 dataset; the Adm optimizer [32] is adopted in the different models.

TABLE 2. The Pseudocode of the second-stage training.

Algorithm 1: Adversarial training in the second-stage.

Input the random noise z, original examples x, batch size, the α, β_1, β_2 in Adam optimizer and other parameters, inherit the trained parameters of the generator θ_0 and initialize the gradient punishment ω_0;
While θ not converged:
  For i in the penalty period of generator:
    For j in the batch size:
      The generator generates adversarial examples $x' \leftarrow G_{adv}(z)$ from random noise z;
      Linear interpolation $\hat{x} \leftarrow \varepsilon x + (1-\varepsilon)x'$ for original examples x and generated adversarial examples x';
      Calculate the gradient punishment $\left\|\nabla_{\hat{x}} D(\hat{x})\right\|_2$;
      Calculate the loss of discriminator
      $$L_D^i = D_\omega(x') - D_\omega(x) + \lambda\left(\left\|\nabla_{\hat{x}} D_\omega(\hat{x})\right\|_2 - 1\right)^2;$$
    End
    Update gradient punishment
    $$\omega \leftarrow \mathrm{Adam}\left(\nabla_{\hat{x}} 1/m \sum_{i=1}^{m} L_D^i, \omega, \alpha, \beta_1, \beta_2\right);$$
  End
  Select batch size of random noise $\{z^i\}_{i=1}^m \sim p(z)$;
  The trained generator from second-stage generates adversarial examples $x' \leftarrow G_{adv}(z)$ from random noise z;
  The trained generator from first-stage generates adversarial Examples $\tilde{x} \leftarrow G_{norm}(z)$ from random noise z;
  Calculate the loss of generator $L_G = -D_\omega\left(G_{adv_\theta}(z)\right)$;
  Calculate the distance of L_2 norm $L_{perturb} = \left\|\tilde{x} - x'\right\|_2$;
  Enter the adversarial examples x' and normal examples $\tilde{x}$ to the feature extraction network $N_{feature}$;
  Calculate the loss of content features
  $$L_{content} = \left\|N_{feature}(x') - N_{feature}(\tilde{x})\right\|_2^2;$$
  Enter the adversarial examples x' to the target model f;
  Calculate the predicted value as the adversarial loss
  $$L_{adv} = \log_f\left(x', y_{target}\right);$$
  Update the parameters of the generator
  $$\theta \leftarrow \mathrm{Adam}\left(\nabla_\theta 1/m \sum_{i=1}^{m} L_G, L_{perturb}, L_{content}, L_{adv}, \theta, \alpha, \beta_1, \beta_2\right);$$
End
Output the generated adversarial examples $G_{adv}(z)$.

Among them, parameters used in different datasets are shown in the Table 3. and Table 4. And the Table 5. shows architecture and parameters of the LeNet5.

The LeNet5 and ResNet32 as the target model for MNIST and CIFAR-10, then TSGAN is trained on these datasets. Fig. 4. shows the adversarial examples generated by TSGAN, (a) is generated from MNIST dataset with LeNet5, and the category of the attack is the number 0; (b) is generated from CIFAR-10 with ResNet32, and the category of the attack is the plane.

TABLE 3. The setting of the MNIST dataset.

Parameters	Value
Batch Size	128
Normal Learning Rate	2E-04
Adversarial Learning Rate	1E-05
λ_1	0.4
λ_2	0.2
λ_3	0.2
Epoch in Normal Training	800
Epoch in Adversarial Training	100

λ_1, λ_2 and λ_3 are the coefficients of the total loss function.

TABLE 4. The setting of the CIFAR-10 dataset.

Parameters	Value
Batch Size	64
Normal Learning Rate	2E-04
Adversarial Learning Rate	2E-04
λ_1	1
λ_2	0.5
λ_3	1
Epoch in Normal Training	3000
Epoch in Adversarial Training	1000

λ_1, λ_2 and λ_3 are the coefficients of the total loss function.

TABLE 5. The architecture of LeNet5.

Parts	Process	Value
Input	/	$1\times28\times28$
Layer1	Conv	$32\times28\times28$
Layer2	Conv	$34\times14\times14$
Layer3	FC	1×200
Output	FC	1×10

2) THE ATTACK OF THE NORMAL CLASSIFIER

First, attack the target model after normal training. The four target model networks are trained in respective datasets as the target of attack. Then use the four different models as the target model to train the attack model TSGAN on MNIST and CIFAR-10. After two-stage training, take out the generators of the TSGAN as the adversarial example attack tools to generate the adversarial examples. Then put the generated examples to the four different trained classification model, and test the attack effect.

The adversarial example attacks in this part are used the generated adversarial examples to attack the target classifier model. When the classifier model is well-trained, the classifier model has the ability of classify. The generated adversarial examples as the input data to the trained classifier model, and then the classifier gives a different category from human, it means the generated adversarial examples trick the classifier model and the attack is effective. Three additional methods FGSM, PGD (Projected Gradient Descent) and AdvGAN are added in the experiment to compare with the TSGAN.

TABLE 6. The attack success rate of different methods on normal training.

Attack Methods	MNIST	CIFAR-10		
	LeNet5	ResNet20	ResNet32	Wide ResNet28
FGSM	82.63%	79.68%	84.37%	68.75%
PGD	98.29%	92.81%	93.20%	94.29%
AdvGAN	88.66%	94.96%	92.13%	95.28%
TSGAN	95.31%	95.83%	98.28%	98.40%

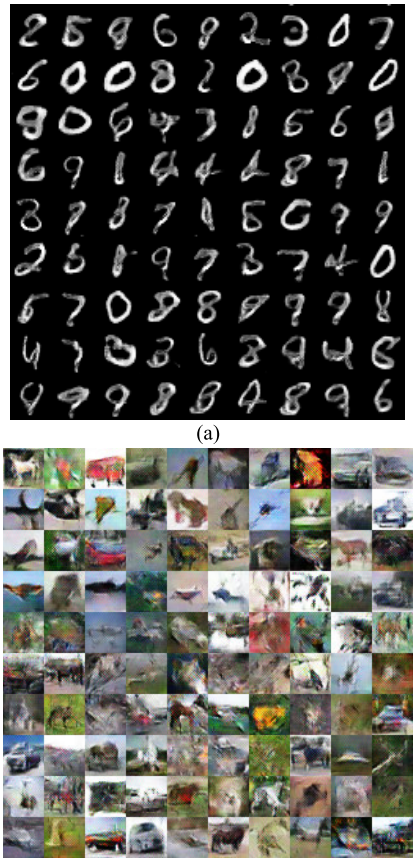

FIGURE 4. The generated adversarial examples on MNIST and CIFAR-10 by TSGAN. (a) is generated with the MNIST, (b) is generated with the CIFAR-10.

By calculating the derivative of the model with the input example and its specific gradient direction, and then multiplying by a step-length coefficient, the resulting disturbance is added to the original input example, finally, the adversarial example is obtained under the FGSM attack. PGD is a variant of BIM that produces an adversarial example by searching for input random noise, and is similar to the gradient descent algorithm, while the PGD contains projection operation. AdvGAN maps the original example to adversarial disturbance through the generator, and then add the disturbance to the original example to synthetize an adversarial example. Table 6. shows the attack success rate of different attack algorithms.

From the Table 6, the attack effect of the TSGAN is better than FGSM and AdvGAN, but slightly worse than PGD on the MNIST dataset. As for bigger CIFAR-10, the attack effect of the TSGAN is better than the other traditional attack methods in three different classification networks. Because the PGD attack [33] is an iterative attack, and this attack is the strongest first-order attack, the attack effect on small datasets is better.

Due to the characteristics of the GAN, after training, it can quickly generate a large number of unconstrained adversarial examples from random noise. So, the speed of generation is faster than the traditional methods without disturbance calculation.

3) THE ATTACK OF THE ADVERSARIAL CLASSIFIER

At present, there are already defense methods to adversarial example attack. The defense model which after training can obtain partial resistance to attack and reduce the attack success rate. To further test the attack ability of the proposed method in this paper, the models that have been trained for defense are used to be attacked. The LeNet5 with MNIST and different ResNet with CIFAR-10 are also used as the target models.

First step, four different models (LeNet5, ResNet20, ResNet32 and Wide Resnet28) were trained on their respective dataset, then training the TSGAN with the four target models. Secondly, the four models are retrained by the adversarial methods of FGSM and Iterative, and eight trained target models are obtained. Because the model has been adversarial trained, it will lose some recognition ability of original example while improving the defense ability. Finally, using the TSGAN to generate the adversarial examples and attack the eight target models.

The FGSM and PGD methods are used in this experimental part, and attack method AdvGAN based on GAN is also adopted to compare with our proposed model. The results are given in the Table 7 and 8.

From the Tables, the attack success rate of four different attack methods are obtained in eight models. The attack effects of the proposed method are almost better than the AdvGAN method, except the ResNet20 model which uses Iterative method to adversarial training. So, compare with the AdvGAN, the proposed attack method in this paper is more difficult to defend. As mentioned above, PGD method is an iterative attack with strong attack force, and neither TSGAN nor AdvGAN method exceeds this method. However, as for attack speed, PGD attack is calculated by n iterations, and the time complexity is $O(n)$, while TSGAN generates adversarial examples directly after training, and the complexity is $O(1)$. The PGD method of 20 iterations is used in the experiments, therefore, the speed of PGD generation is well below than that of TSGAN method in this paper.

TABLE 7. The attack success rate of different methods on adversarial training in MNIST dataset.

Attacked Model	Defense	MNIST			
		FGSM	PGD	AdvGAN	TSGAN
LeNet5	Adv	10.62%	94.98%	11.61%	45.84%
	Iter	14.23%	11.54%	10.02%	23.10%

Adv means the model that under the FGSM adversarial training, and the Iter means that the model under the Iterative adversarial training.

TABLE 8. The attack success rate of different methods on adversarial training in CIFAR-10 dataset.

Attacked Model	Defense	CIFAR-10			
		FGSM	PGD	AdvGAN	TSGAN
ResNet20	Adv	13.69%	76.88%	13.74%	18.13%
	Iter	33.96%	39.14%	16.16%	12.41%
ResNet32	Adv	7.55%	80.02%	13.40%	18.25%
	Iter	32.08%	39.37%	16.63%	18.84%
Wide ResNet28	Adv	6.03%	85.79%	15.55%	23.40%
	Iter	28.79%	38.80%	15.80%	29.40%

Adv means the model that under the FGSM adversarial training, and the Iter means that the model under the Iterative adversarial training.

TABLE 9. The adversarial example transfer attack on different target model.

Target Model (Training)	Target Model (Attacking)	AdvGAN	TSGAN
ResNet20	ResNet32	40.68%	73.09%
	Wide ResNet28	44.02%	44.36%
ResNet32	ResNet20	94.31%	77.58%
	Wide ResNet28	83.61%	50.30%
Wide ResNet28	ResNet20	14.05%	73.69%
	ResNet32	15.16%	77.24%

TABLE 10. Comparison of attack results with and without content feature constraints.

Attack Model	MNIST	CIFAR-10
	LeNet5	ResNet32
TSGAN (Without Feature)	Wide ResNet28	44.36%
TSGAN (With Feature)	ResNet20	77.58%

4) THE TRANSFER OF TSGAN

The target model of training and attacking is different in the adversarial example transfer attack. Because the adversarial example transfer attack is aimed at different target models, the transfer ability can also show the black-box attack ability of the attack method.

In order to test the transfer attack of adversarial examples generated by the proposed model, the ResNet20, ResNet32 and Wide ResNet28 are used as the target model to train the TSGAN. Three attack models are obtained, then the attack models exchange the attack target models and observe the attack effect. The results of the attack are shown in Table 9.

From the three groups of experimental results, by comparison with the attack on different target models' success rate, the adversarial examples generated by proposed method have strong transfer ability. And has got 77.58% success rate of the transfer attack. Compared with AdvGAN, except ResNet32 model, TSGAN get higher attack success rate in the other experimental groups, which shows that the adversarial example produced by TSGAN is more stable. When the Wide ResNet28 model with more parameters is used as the target model to training the TSGAN, and the higher attack success rate in the transfer attack than other two ResNet models. So, a stronger attack ability can be obtained when training the TSGAN with a complex network. The above results show that the adversarial examples generated by the proposed model when training with a target model have a strong black-box attack ability to other classification models. And can use the proposed model to attack the other black-box model.

C. THE GENERATION QUALITY OF THE TSGAN

To investigate the role of content features in the generated adversarial examples' quality and attack effect, compare the TSGAN with the content feature to without content feature.

1) THE COMPARATION OF ATTACK EFFECT

In this section, the content feature constraint in TSGAN attack method is removed to compare with the standard TSGAN. Attack with target models that have not been adversarial trained, the LeNet5 and ResNet32 are used as trained model on MNIST and CIFAR-10 datasets. The attack success rates are given in Table 10.

From the table, after removing the content feature, the attack success rate on MNIST data set decreased by 0.13%, and the attack success rate on CIFAR-10 data set increased by 0.92%. From the results, the content features have no effect on attack effect, and even have a slight reaction.

2) THE COMPARATION OF QUALITY

To verify the relationship between content features and the quality of generated adversarial examples, the examples'

quality of generated by WGAN-GP, TSGAN without content features and TSGAN with content features are compared. For compare the quality of the generated examples, after training, a quantity of adversarial examples according to MNIST and CIFAR-10 dataset. The details of examples are shown in Fig. 5.

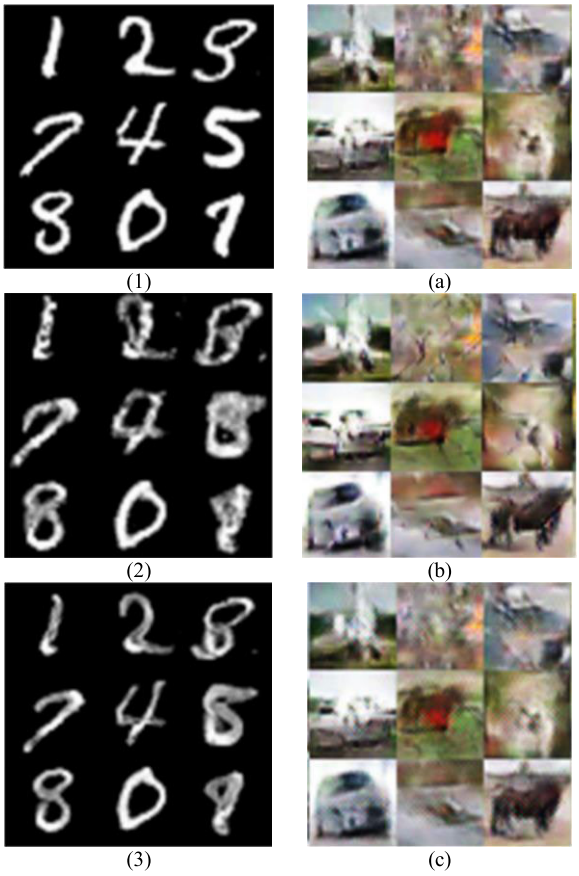

FIGURE 5. The quality of the generated adversarial examples on MNIST and CIFAR-10. (1) and (a) are generated by WGAN-GP in first-stage, (2) and (b) are generated by TSGAN without content feature, (3) and (c) are generated by TSGAN with content feature.

It can be found from the images, the adversarial examples generated by TSGAN without content feature has a trend towards semantic change. Because the target category attacked is 0 and plane, the Fig. 5. (1) and (a) are generated by the WGAN-GP after first-sate training as the standard adversarial examples. And the adversarial examples generated by TSGAN without content feature begin to show the content characteristics of the target category in the image. It's a bad phenomenon when generating the adversarial examples. In the Fig. 5. (2), the numbers 4, 7 and 5 all have the closed feature of the number 0; the images of Fig. 5. (b) appear the characteristics of sharp nose such as nose and wing in aircraft category. In Fig. 5. (3) and (c), these phenomena have been significantly improved.

The Fig. 6. shows the characteristics of some typical examples' change.

To further compare the image quality of the proposed method with other GAN based methods, the AdvGAN

FIGURE 6. The characteristics adversarial examples of the WGAN-GP.

and Adv-ACGAN are selected. AdvGAN generation is based on the calculation of disturbance superposition; and Adv-ACGAN is based on the direct generation of unrestricted examples. The comparison of the different attack adversarial example generation based on GAN is shown in Fig. 7. The first and second lines are extracted from the corresponding papers.

Through comparison, we can see that adversarial examples of AdvGAN have obvious noise due to the superposition of disturbance, while adversarial examples of Adv-ACGAN have a great change in the semantic content, such as numbers 3, 6, 9. In the third line of Fig. 7, the number examples generated by proposed method avoids the change of image semantics, and maintains the digital features. The disturbance superposition produced inside the outline of the numbers, which has no significant perceptible impact on the image feature information.

At the end, the quantitative descriptions of generated image's quality are compared, including the WGAN-GP, AdvGAN, TSGAN without content feature and TSGAN with content feature. Fréchet Inception Distance (FID) [34] and Structural Similarity (SSIM) [35] are the evaluations that commonly used in the image's quality. The metric of FID is the distance between the feature vectors of the real image and the result image, the smaller the index, the higher the image similarity; and the SSIM measures the similarity between the two images, when the index is close to 1, a high similarity of the two images. The Table 11 and 12 reveal the similarity of the adversarial examples.

The generated adversarial examples of the WGAN-GP in first-sate were used as the comparison criteria, because examples generated at this stage are resembled to real. From the FID and SSIM metrics in the tables, the adversarial examples generated by TSGAN with content feature are better than

TABLE 11. The quality evaluation of The MNIST dataset.

Model	FID	SSIM
WGAN-GP	/	/
TSGAN without content feature	276.63	0.7863
TSGAN with content feature	131.42	0.7878
AdvGAN	276.19	0.4772

TABLE 12. The quality evaluation of The CIFAR-10 dataset.

Model	FID	SSIM
WGAN-GP	/	/
TSGAN without content feature	37.08	0.6673
TSGAN with content feature	8.42	0.7975
AdvGAN	11.08	0.7665

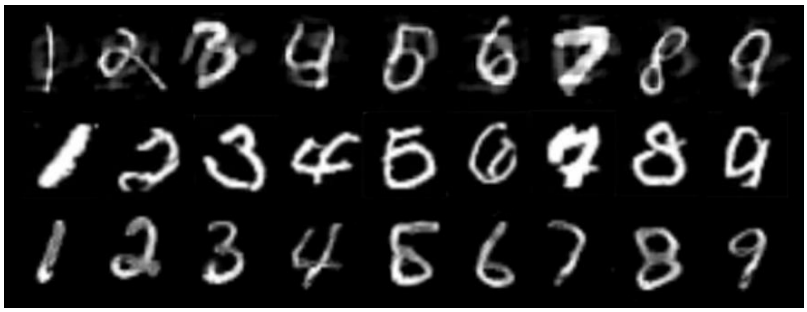

FIGURE 7. The comparison of adversarial examples' quality. The first line is generated by AdvGAN, the second line is generated by Adv-ACGAN, the third line is generated by TSGAN.

then TSGAN without content feature in the two datasets. Meanwhile, the better-quality assessments are obtained by comparing AdvGAN method. It illustrates that the TSGAN with content feature can generate a more realistic example. So, the TSGAN with content feature has higher quality of adversarial examples under the premise of ensuring the success rate of attack.

By comparing the quality of the adversarial examples of different generation methods, the TSGAN method proposed in this paper can better learn the distribution the adversarial examples through the unsupervised training. The generation of adversarial examples for attacked target category do not change the semantic information of the image, and more consistent with human judgment. By generating adversarial examples directly, the limitation of the original sample superposition perturbation method is avoided, and unlimited adversarial examples can be quickly generated in batches.

D. THE GENERATION EFFICIENCY OF THE TSGAN

Finally, the generation efficiency of the above attack models is compared by the 1000 adversarial examples generation time. The experimental results are shown in Table 13. FGSM, which is the fastest in traditional attack methods, takes about 0.66 seconds. PGD, which has high-complexity of computation and slow-generation for 20 iterations, needs 4.83 seconds. AdvGAN generated the examples by the disturbances and superposition, and the generation efficiency increases an order of magnitude and takes about 0.036 seconds. TSGAN directly generates adversarial examples, which is nearly three times faster than AdvGAN. It is shown that TSGAN after training can generate adversarial examples more quickly.

TABLE 13. The example generation time.

	FGSM	PGD	AdvGAN	TSGAN
Time(s)	0.66s	4.83s	0.036s	0.013s

V. CONCLUSION

An adversarial example attack method with content characteristic constraints TSGAN is proposed in this paper, which based on two-stage training. During the training, the TSGAN pays attention to the semantic information of examples, and no more superimposed perturbations. In the first-stage, the dataset is used for normal training; and semantic content constraints are adopted for second-stage training. Based on the training strategy of the TSGAN, the adversarial examples generated by the model are not limited to the existing examples, while the examples are generated directly by the attack model with learned data distribution. In the experiment part, the MNIST and CIFAR-10 datasets are used to training the TSGAN and other models. Experiment results show that the attack success rate of TSGAN is 23.81% better than FGSM in two datasets; 2.5% batter than PGD and 4.59% better than AdvGAN. As for generation time, the TSGAN is 98.03% faster than FGSM, 99.73% faster than PGD and 63.89% faster than AdvGAN. The attack method proposed in this paper have high attack success rates and good performance in image quality.

In addition, how to further improve the attack success rate of the attack model on the basis of ensuring the quality of generated examples is our future work, and how to generate some adversarial examples with a series of encrypted images might be an interesting work [36]. Meanwhile, how to effectively defend this kind of adversarial example attack will also be our next consideration.

REFERENCES

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Banff, AB, Canada, 2014, pp. 1–10.

[2] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Toulon, France, 2017, pp. 1–14.

[3] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on State-of-the-Art face recognition," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Vienna, Austria, Oct. 2016, pp. 1528–1540.

[4] J. Lu, H. Sibai, and E. Fabry, "Adversarial examples that fool detectors," 2017, *arXiv:1712.02494*. [Online]. Available: http://arxiv.org/abs/1712.02494

[5] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 1625–1634.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. ICONIP*, Montreal, QC, Canada, 2014, pp. 2672–2680.

[7] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, 2015, pp. 1–9.

[8] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, 2015, pp. 1–11.

[9] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 9185–9193.

[10] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Jose, CA, USA, May 2017, pp. 39–57.

[11] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 2574–2582.

[12] S. Baluja and I. Fischer, "Adversarial transformation networks: Learning to generate adversarial examples," 2017, *arXiv:1703.09387*. [Online]. Available: http://arxiv.org/abs/1703.09387

[13] A. J. Bose and P. Aarabi, "Adversarial attacks on face detectors using neural net based constrained optimization," in *Proc. IEEE 20th Int. Workshop Multimedia Signal Process. (MMSP)*, Vancouver, BC, Canada, Aug. 2018, pp. 1–6.

[14] Z. Zhao, D. Dua, and S. Singh, "Generating natural adversarial examples," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Vancouver, BC, Canada, 2018, pp. 1–15.

[15] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song, "Generating adversarial examples with adversarial networks," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Stockholm, Sweden, Jul. 2018, pp. 3905–3911.

[16] S. Jandial, P. Mangla, S. Varshney, and V. Balasubramanian, "AdvGAN++: Harnessing latent layers for adversary generation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Seoul, Republic of Korea, Oct. 2019, pp. 2045–2048.

[17] Y. Song, N. Kushman, R. Shu, and S. Ermon, "Constructing unrestricted adversarial examples with generative models," in *Proc. NeurIPS*, Montreal, QC, Canada, 2018, pp. 8312–8323.

[18] X. Wang, K. He, C. Song, L. Wang, and J. E. Hopcroft, "AT-GAN: An adversarial generator model for non-constrained adversarial examples," 2019, *arXiv:1904.07793*. [Online]. Available: http://arxiv.org/abs/1904.07793

[19] F. Chen, Y. Shang, J. Hu, and B. Xu, "Few features attack to fool machine learning models through mask-based GAN," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Glasgow, U.K., Jul. 2020, pp. 1–7.

[20] G. Zhao, M. Zhang, J. Liu, and J.-R. Wen, "Unsupervised adversarial attacks on deep feature-based retrieval with GAN," 2019, *arXiv:1907.05793*. [Online]. Available: http://arxiv.org/abs/1907.05793

[21] L. Jiang, K. Qiao, R. Qin, L. Wang, W. Yu, J. Chen, H. Bu, and B. Yan, "Cycle-consistent adversarial GAN: The integration of adversarial attack and defense," *Secur. Commun. Netw.*, vol. 2020, pp. 1–9, Feb. 2020.

[22] Z. Lin, Y. Shi, and Z. Xue, "IDSGAN: Generative adversarial networks for attack generation against intrusion detection," 2018, *arXiv:1809.02077*. [Online]. Available: http://arxiv.org/abs/1809.02077

[23] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2414–2423.

[24] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," 2015, *arXiv:1508.06576*. [Online]. Available: http://arxiv.org/abs/1508.06576

[25] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 694–711.

[26] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, 2015, pp. 1–14.

[28] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5767–5777.

[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2323, Nov. 1998.

[30] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep. TR 2009, 2009, pp. 3–16.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778.

[32] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, 2015, pp. 1–15.

[33] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Vancouver, BC, Canada, 2018, pp. 1–28.

[34] D. C. Dowson and B. V. Landau, "The Fréchet distance between multivariate normal distributions," *J. Multivariate Anal.*, vol. 12, no. 3, pp. 450–455, Sep. 1982.

[35] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[36] C. L. Chowdhary, P. V. Patel, K. J. Kathrotia, M. Attique, K. Perumal, and M. F. Ijaz, "Analytical study of hybrid techniques for image encryption and decryption," *Sensors*, vol. 20, no. 18, p. 5162, Sep. 2020.

JIANYI LIU received the Ph.D. degree in signal and information processing from the Beijing University of Posts and Telecommunications, in 2005. He is currently an Associate Professor with the Beijing University of Posts and Telecommunications. His current research interests include information security, natural language processing, and big data analysis.

YU TIAN received the M.S. degree from the Beijing University of Posts and Telecommunications, in 2020. His research interests include watermarking, deep learning, and information security.

RU ZHANG (Member, IEEE) received the Ph.D. degree from the School of Computer Science, Beijing Institute of Technology, in 2003. She is currently a Professor with the Beijing University of Posts and Telecommunications. Her research interests include multimedia security, watermarking, and vulnerability analysis.

YOUQIANG SUN is currently pursuing the Ph.D. degree with the Beijing University of Posts and Telecommunications. His research interests include deep learning and information security.

CHAN WANG employed with the Information and Communication Branch, State Grid Corporation of China. Her research interest includes enterprise network security.