TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 10/23 pp430–446
DOI: 10.26599/TST.2023.9010004
Volume 29, Number 2, April 2024

DCVAE-adv: A Universal Adversarial Example Generation Method for White and Black Box Attacks

Lei Xu and Junhai Zhai*

Abstract: Deep neural network (DNN) has strong representation learning ability, but it is vulnerable and easy to be fooled by adversarial examples. In order to handle the vulnerability of DNN, many methods have been proposed. The general idea of existing methods is to reduce the chance of DNN models being fooled by observing some designed adversarial examples, which are generated by adding perturbations to the original images. In this paper, we propose a novel adversarial example generation method, called DCVAE-adv. Different from the existing methods, DCVAE-adv constructs adversarial examples by mixing both explicit and implicit perturbations without using original images. Furthermore, the proposed method can be applied to both white box and black box attacks. In addition, in the inference stage, the adversarial examples can be generated without loading the original images into memory, which greatly reduces the memory overhead. We compared DCVAE-adv with three most advanced adversarial attack algorithms: FGSM, AdvGAN, and AdvGAN++. The experimental results demonstrate that DCVAE-adv is superior to these state-of-the-art methods in terms of attack success rate and transfer ability for targeted attack. Our code is available at https://github.com/xzforeverlove/DCVAE-adv.

Key words: deep neural network; adversarial examples; white box attack; black box attack; robustness

1 Introduction

Deep learning[1] has made incredible progress since AlexNet[2] was first proposed and won the ImageNet Large Scale Visual Recognition Competition (ILSVRC) by a big margin in 2012. Since then, many deep neural networks are proposed and some of them have become benchmark backbones in computer vision, such as VGG[3], ResNet[4], ZFNet[5], SENet[6], GoogleNet[7], Xception[8], and EfficientNet[9]. Although deep neural networks (DNNs) have shown great performances in many computer vision tasks, it is found that DNNs are vulnerable to adversarial examples[10], which are designed elaborately by attackers to fool deep learning models. This finding[10] sparked a hot research topic: adversarial machine learning[11–15].

In adversarial machine learning, attackers use adversarial examples to attack deep learning models. An adversarial example is a sample of input data which has been modified very slightly in a way that is intended to cause a deep learning model to misclassify it. These modifications are so subtle that it is hard for a human observer to find the difference between an adversarial example and the original images. Based on attacker's knowledge, attacks can be categorized into white box attacks and black box attacks[11]. For white box attacks, the attackers know everything related to trained DNNs, including training data, model architectures, hyper-parameters, model weights, etc. Whereas for black box attacks, the attackers only know the outputs and do not have any other information of the DNNs. Based on attack targets, attacks can be categorized into targeted attacks and non-targeted attacks[11]. Targeted attacks aim

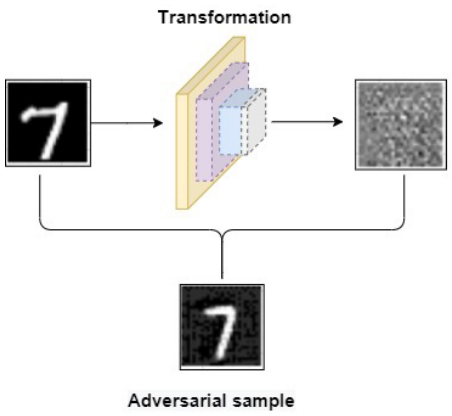
• Lei Xu and Junhai Zhai are with College of Mathematics and Information Science, Hebei University, Baoding 071002, China. E-mail: 392336077@qq.com; mczjh@126.com.
∗ To whom correspondence should be addressed.
Manuscript received: 2022-09-27; revised: 2022-12-26; accepted: 2023-01-29

at misguiding DNNs to classify adversarial examples to a target class. Whereas non-targeted attacks have no target class, the adversarial examples can be incorrectly classified into arbitrary classes except for the correct class. Obviously, black box and targeted attacks are more difficult than white box and non-targeted attacks, respectively.

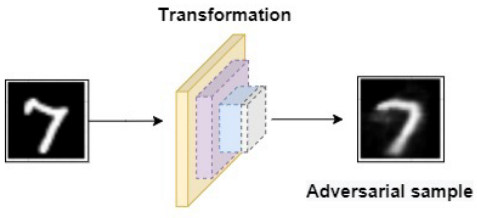
Most existing methods[16–20] generate adversarial examples by elaborately designing perturbations and adding them to the input examples, and the imperceptible perturbations are usually obtained by solving the corresponding optimization problems. Moreover, solving the optimization problems requires repeated iterations, resulting in high computational time complexity and difficulty in generating adversarial examples quickly and effectively. Inspired by the idea of generative models (e.g., generative adversarial network (GAN)[21] and variational autoencoder (VAE)[22]), some generative models based adversarial example generation methods have been proposed in recent years, such as Rob-GAN[23], MAG-GAN[24], CGAN-Adv[25], AdvGAN[26], and AdvGAN++[27]. Among these generation based models, there are relatively few VAE based adversarial example generation methods compared with the GAN based ones. To the best of our knowledge, there are only two prior works that are VAE based[28, 29].

Different from prior iterative approaches, generation based methods can take advantage of generative models and generate perturbations or even the adversarial examples directly. There are two types of representative adversarial example generation methods[26, 27]. One type of approaches first constructs the adversarial perturbations (also called explicit perturbations), then adds the constructed adversarial perturbations to the original images to achieve the adversarial examples. The other type of approaches directly converts the original examples into the corresponding adversarial examples through the generator. This type of approaches does not construct adversarial perturbations and thus is also called implicit disturbances. The differences between adversarial examples generated with explicit perturbations and implicit disturbances are given in Fig. 1. We find that by adding explicit perturbations to generate adversarial examples, the success rate of attack mainly depends on the perturbation amplitude. Larger scale perturbations usually bring higher success rate of attack, but it is relatively easier to be detected by human visual system. If the amplitude of perturbations is too small, the effect of attack cannot be achieved. In the case of transfer attacks, this kind of methods performs well on the dataset with simple distribution. However, the adversarial example generation methods based on implicit perturbation perform well on more complex datasets, but the results are often difficult to control.

In order to deal with the above problems, this paper proposes an adversarial sample generation method based on improved VAE named DCVAE-adv. Different from previous methods, instead of modeling the mapping between original examples and corresponding perturbations or adversarial examples, DCVAE-adv generates adversarial examples from scratch with the help of the powerful generation ability of variational autoencoder, further expanding the search domain of adversarial samples. More specifically, DCVAE-adv applies both explicit and implicit methods by generating perturbation $\boldsymbol{x}_{\text{noise}}$ and adversarial sample $\boldsymbol{x}_{\text{adv}}$ simultaneously and the final adversarial sample is constructed by adding them together. Using the combination of explicit and implicit perturbations not



(a) Diagram of the explicit generation method



(b) Diagram of the implicit generation method

Fig. 1 Difference between explicit and implicit adversarial example generation methods.

only reduces the difficulty of adversarial example generation but also improves the success rate of transfer attack. Moreover, class information is added in the training process, which can alleviate the mode collapse problems and ensure high diversity of the generated adversarial examples. It can also help achieve the goal of generating adversarial examples of the specified class. Another advantage of DCVAE-adv is that during the adversarial example generation stage, adversarial examples can be generated from random noises and do not need to load original examples into memory, which can greatly reduce the memory and time cost of adversarial example generation and significantly improve the efficiency of the proposed algorithm.

To sum up, the main contributions of the proposed method contain the following three aspects:

(1) We propose a novel method DCVAE-adv to generate adversarial examples, which can significantly improve the success rate of attacks and transfer attacks by combining both explicit and implicit perturbations. Compared with prior methods which require iterative updates or original examples as inputs, DCVAE-adv is more time efficient and memory efficient.

(2) DCVAE-adv is based on an improved VAE. The improvement for VAE includes three aspects: (a) A better similarity measurement criterion is introduced to VAE, which not only brings better generation results compared with norm measurement criterion but also reduces the query times of target network and improves training efficiency in semi-black box attack. (b) Class information is introduced to VAE, which not only achieves the purpose of generating adversarial examples of the specified class but also enables the adversarial example generation when the original examples cannot be obtained, thus reducing the memory overhead. (c) A discriminator is introduced to VAE to constrain the quality of the generated adversarial examples.

(3) Extensive experiments are conducted, and the proposed method is compared with three state-of-the-art approaches. The experimental results showed that the adversarial examples generated by DCVAE-adv are of high quality, have better transfer ability, and can be adapted to more difficult black box attack scenario.

The rest of this paper is organized as follows. In Section 2, we review the related work of adversarial example generation. In Section 3, we describe the details of the proposed methods. In Section 4, extensive experiments compared DCVAE-adv with three most advanced adversarial attack algorithms (FGSM, AdvGAN, and AdvGAN++) are carried out to verify the effectiveness of the proposed approaches. At last, we conclude our work in Section 5.

2 Related Work

Since the concept of adversarial attack was proposed by Szegedy et al.[10], many adversarial example generation methods have been proposed in the literature, which can be roughly classified into two categories: optimization-based methods and generation-based methods.

The optimization-based approaches solve an optimization problem to find the minimum perturbations to be added to the input image, and the added perturbations are so subtle that it is hard for a human observer to notice the modification to the input image, while the deep learning model would classify the modified input image incorrectly. The pioneering work of this category is L-BFGS[10], which generates adversarial examples $\boldsymbol{x}'$ by solving the optimization problem $\min_{\boldsymbol{\delta}} \|\boldsymbol{x}-\boldsymbol{x}'\|_2$, where $\boldsymbol{x}$ is the input image. Goodfellow et al.[30] improved L-BFGS by constraining the perturbations in gradient direction, replaced $\|\cdot\|_2$ with $\|\cdot\|_\infty$, and proposed fast gradient sign method (FGSM). The corresponding optimization problem becomes $\min_{\boldsymbol{\delta}} \|\boldsymbol{x}-\boldsymbol{x}'\|_\infty$, where $\boldsymbol{\delta} = \varepsilon \cdot \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{x}, y))$, y is the class label of $\boldsymbol{x}$, and ε is a hyper-parameter. Based on the state-of-the-art algorithm FGSM, many improved algorithms have been proposed by researchers, such as IGSM[16], JSMA[17], and DeepFool[18]. Some other representative algorithms based on optimization models are reported in Refs. [31–35]. Athalye et al.[31] introduced a concept named expectation over transformation (EOT) and proposed an adversarial example generation method based on the EOT. Deng and Zeng[32] introduced the attention mechanism based on gradient-weighted class activation mapping to generate adversarial examples. Concretely, the attention mechanism is used to find the meaningful attack area for generating high quality adversarial examples. Similar to Ref. [32], Xu et al.[33] introduced the self-attention mechanism to deep learning model and proposed a self-attention network to enhance robustness of the deep neural networks for hyperspectral image classification. Vidnerová and Neruda[34] proposed an evolutionary algorithm to generate adversarial examples, which is generic and can be used for any machine learning model in the black box attack scenario.

Since generative models, such as generative

adversarial network (GAN)[21] and its variants[23, 35–38] as well as variational autoencoder[22, 38–41] and its variants, can generate samples according to some probability distribution, it is a natural idea to use generative models to generate adversarial examples. The pioneering work of this category is AdvGAN[26], which generates adversarial examples using GANs. The advantage of AdvGAN is that it can learn and approximate the distribution of original instances using the approximation capability of GAN. Jandial et al.[27] improved AdvGAN and proposed AdvGAN++ by employing latent features as priors for generating adversarial examples. Chen et al.[24] proposed a novel GAN model with a massive attack generator that is used for generating adversarial examples, and the advantage of their method is that the generated examples have strong attack ability with small perturbations. Zhao et al.[42] observed that common malicious perturbations are often unnatural, not semantically meaningful, and not applicable to complicated domains. To this end, they proposed a framework to generate natural and legible adversarial examples that lie on the data manifold. Song et al.[43] proposed an approach to synthesize unrestricted adversarial examples. Different from most existing methods that generate small norm-bounded perturbations, the proposed approach constructs adversarial examples from scratch using generative adversarial network with an auxiliary classifier. Sharif et al.[44] proposed a general framework, adversarial generative nets (AGNs), to generate adversarial examples satisfying desired objectives by training a generator in AGNs. Liu et al.[45] proposed a two-stage generative adversarial networks with semantic content constraints to generate adversarial examples satisfying predefined semantic constraints. Tang et al.[46] proposed a distance constrained adversarial imitation network (AIN). AIN can generate both targeted and non-targeted examples with an explicit distance constraint. Hu et al.[47] proposed an elastic-net regularized boundary equilibrium generative adversarial network (ERBEGAN) for generating adversarial examples. ERBEGAN improves the diversity and robustness of adversarial examples by introducing both L_2-norm and L_1-norm of perturbation as regularizations to the objective function of ERBEGAN. Zhang[48] viewed the problem of generating adversarial examples as an image-to-image translation problem and proposed an adversarial example generation method in one shot with image-to-image translation GAN. To overcome the drawback that the existing GAN based adversarial sample generation methods can not be effectively adapted to the black box scenario, Yu et al.[28] proposed an adversarial examples generation method based on conditional variational autoencoder. Based on β-VAE, Upadhyay and Mukherjee[29] proposed a novel adversarial example generation method, in which different from the common perturbation-based methods, the actual manipulation takes place in latent space. The latest research progress and challenges on adversarial examples and attacks can be found in Refs. [12–15, 49].

3 Proposed Method

3.1 Problem formulation

The goal of the proposed method DCVAE-adv is to generate adversarial examples of a given class. The encoder network is used to extract the distribution information of the original samples, and then $\boldsymbol{z}_{\text{noise}}$ is sampled from a learned distribution. The decoder receives the noise vector $\boldsymbol{z}_{\text{noise}}$ and the class information and generates the adversative example. The problem can be formulated as follows:

$$\boldsymbol{x}_{\text{adv}} = \text{Decoder}(\boldsymbol{z}_{\text{noise}}, T(\boldsymbol{x})) \tag{1}$$

$$T(\boldsymbol{x}) \neq T(\boldsymbol{x}_{\text{adv}}) \tag{2}$$

$$\text{sim}(\boldsymbol{x}, \boldsymbol{x}_{\text{adv}}) > \lambda \tag{3}$$

In Eq. (1), $\boldsymbol{z}_{\text{noise}} = \boldsymbol{\mu} \times z + \boldsymbol{\sigma}$, where $z \sim N(\boldsymbol{0}, \boldsymbol{I})$, $(\boldsymbol{\mu}, \boldsymbol{\sigma}) = \text{Encoder}(\boldsymbol{x})$, and $T(\boldsymbol{x})$ is the attack network which predicts the class of $\boldsymbol{x}$. Formula (2) indicates that the classes of original sample $\boldsymbol{x}$ and the generated adversarial sample $\boldsymbol{x}_{\text{adv}}$ are different. The $\text{sim}(\cdot,\cdot)$ is a similar measure, and Formula (3) indicates that $\boldsymbol{x}$ and $\boldsymbol{x}_{\text{adv}}$ are very similar.

3.2 DCVAE-adv

VAE is a type of deep generative models (see Fig. 2) optimised via variational inference, which allows for the approximation of intractable distributions. Specifically, VAE approximates the distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ with distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ by minimizing the Kullback–Leibler (KL)-divergence between them:

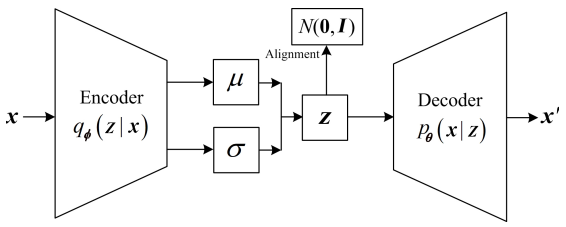


Fig. 2 Architecture of VAE.

$$\mathrm{KL}\left(q_{\boldsymbol{\phi}}(z|\boldsymbol{x})||p_{\boldsymbol{\theta}}(z|\boldsymbol{x})\right)=\sum_{z}q_{\boldsymbol{\phi}}(z|\boldsymbol{x})\log\left(\frac{q_{\boldsymbol{\phi}}(z|\boldsymbol{x})}{p_{\boldsymbol{\theta}}(z|\boldsymbol{x})}\right)=\log p_{\boldsymbol{\theta}}(\boldsymbol{x})-\left(E\left(\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|z)\right)-\mathrm{KL}\left(q_{\boldsymbol{\phi}}(z|\boldsymbol{x})||p_{\boldsymbol{\theta}}(z)\right)\right) \tag{4}$$

Since $\mathrm{KL}(\cdot||\cdot)\geqslant 0$, we have

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x})\geqslant E\left(\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|z)\right)-\mathrm{KL}\left(q_{\boldsymbol{\phi}}(z|\boldsymbol{x})||p_{\boldsymbol{\theta}}(z)\right) \tag{5}$$

The left term of Formula (5) is known as the evidence lower bound (ELBO), which is the objective function that will be optimized by VAE. The first item $E\left(\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|z)\right)$ is the reconstruction loss, and the second item $\mathrm{KL}\left(q_{\boldsymbol{\phi}}(z|\boldsymbol{x})||p_{\boldsymbol{\theta}}(z)\right)$ measures the similarity between the distribution of the latent space and the target distribution $p_{\boldsymbol{\theta}}(z)$. Accordingly, the objective of VAE is to minimize

$$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}=E_{q_{\boldsymbol{\phi}}(z|\boldsymbol{x})}\left(\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|z)\right)-\mathrm{KL}\left(q_{\boldsymbol{\phi}}(z|\boldsymbol{x})||p_{\boldsymbol{\theta}}(z)\right) \tag{6}$$

If the sample size of $\boldsymbol{z}$ is L, then we have

$$L_{\mathrm{VAE}}(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\phi})=L_{\mathrm{reg}}+L_{\mathrm{dis}}=\frac{1}{L}\sum_{l=1}^{L}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|z^{l})-\mathrm{KL}(q_{\boldsymbol{\phi}}(z|\boldsymbol{x})||p_{\boldsymbol{\theta}}(z)) \tag{7}$$

Suppose that $p_{\boldsymbol{\theta}}(z|\boldsymbol{x})\sim N(\boldsymbol{\mu}_{\boldsymbol{x}},\boldsymbol{\sigma}_{\boldsymbol{x}}^{2})$, and $p(z)\sim N(\mathbf{0},\boldsymbol{I})$, then $p_{\boldsymbol{\theta}}(z|\boldsymbol{x})\sim N(G_{\boldsymbol{\theta}}(z),\boldsymbol{\sigma}^{2})$, and the first and second terms on the right side of Eq. (7) will become Eqs. (8) and (9), respectively.

$$L_{\mathrm{reg}}=\frac{1}{L}\sum_{l=1}^{L}\frac{1}{2\sigma^{2}}||G_{\boldsymbol{\theta}}(z)-\boldsymbol{x}||^{2}+\log(\sqrt{2\pi}\sigma^{2}) \tag{8}$$

$$L_{\mathrm{dis}}=-\mathrm{KL}(q_{\boldsymbol{\phi}}(z|\boldsymbol{x})||p_{\boldsymbol{\theta}}(z))=\frac{1}{2}\sum_{i=1}^{d}(\mu_{i}^{2}+\sigma_{i}^{2}-\log\sigma_{i}^{2}-1) \tag{9}$$

where d is the dimension of the latent variable space.

The proposed DCVAE-adv is based on an improved VAE. The improvements over VAE include three aspects:

(1) A better similarity measurement criterion is introduced to VAE, which not only brings better generation results compared with norm measurement criterion but also reduces the query times of target network and improves training efficiency in semi-black box attack. The introduced similarity criterion is pixel cross entropy (PCE) that measures the similarity between $\boldsymbol{x}$ and $\boldsymbol{x}_{\mathrm{adv}}$, i.e., the $\mathrm{PCE}(\boldsymbol{x},\boldsymbol{x}_{\mathrm{adv}})$ is used to calculate the $\mathrm{sim}(\boldsymbol{x},\boldsymbol{x}_{\mathrm{adv}})$. It should be noted that the smaller the $\mathrm{PCE}(\boldsymbol{x},\boldsymbol{x}_{\mathrm{adv}})$ is, the more similar $\boldsymbol{x}$ and $\boldsymbol{x}_{\mathrm{adv}}$ are; Conversely, the greater the value of $\mathrm{PCE}(\boldsymbol{x},\boldsymbol{x}_{\mathrm{adv}})$ is, the less similar $\boldsymbol{x}$ and $\boldsymbol{x}_{\mathrm{adv}}$ are. The definition of $\mathrm{PCE}(\boldsymbol{x},\boldsymbol{x}_{\mathrm{adv}})$ is given in Eq. (10).

$$\mathrm{sim}(\boldsymbol{x},\boldsymbol{x}_{\mathrm{adv}})=\mathrm{PCE}(\boldsymbol{x},\boldsymbol{x}_{\mathrm{adv}})=\sum_{i=1}^{m}\sum_{j=1}^{n}x^{ij}\cdot[-\log(x_{\mathrm{adv}}^{ij})]+(1-x^{ij})\cdot[\log(1-x_{\mathrm{adv}}^{ij})] \tag{10}$$

where x^{ij} represents the value of a pixel point on the feature map with the size of $m\times n$. In our experiments, the value of each pixel of each input sample is transformed into [0, 1] before training.

(2) Class information is introduced to VAE, which brings two advantages. First, it helps achieve the goal of generating adversarial examples of the specified class. Second, it can generate adversarial examples when the original examples are not provided or cannot be obtained, thus reducing the memory overhead. Concretely, we input the class information $\boldsymbol{y}$ into the encoder and decoder network and generate the samples of the specified class by controlling the class. Accordingly, the L_{reg} becomes Eq. (11):

$$L_{\mathrm{reg}}=\frac{1}{L}\sum_{l=1}^{L}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{y},z^{l}) \tag{11}$$

(3) A discriminator is introduced to VAE to constrain the quality of the generated adversarial examples. When traditional VAEs are used to generate adversarial examples, the generated adversarial examples are often blurry. And the vanilla VAE can not achieve the purpose of generating the specified class against the sample. The ACGAN proposed by Odena et al.[50] combines and reconstructs the class information, and it can not only generate forged examples of specified classes but also greatly improve the generation quality, as a result avoiding the phenomenon of mode collapse effectively. Inspired by ACGAN, DCVAE-adv takes the class information $\boldsymbol{y}$ as auxiliary input to the decoder network, and the adversarial examples of the specified class can be generated by controlling the class information. Also, a discriminator is introduced to guarantee that (a) the class label information can be reconstructed from real examples and forged adversarial examples, and (b) adversarial examples are distinguished from the real samples.

The structure of the proposed DCVAE-adv is shown in Fig. 3. DCVAE-adv consists of four components: encoder, decoder, attack network, and discriminator. In the training phase, the encoder learns the distribution $p(\boldsymbol{z}_{\mathrm{noise}}|\boldsymbol{x})$ and outputs $\boldsymbol{z}_{\mathrm{noise}}$ which is the feature representation of the original sample in the latent space.

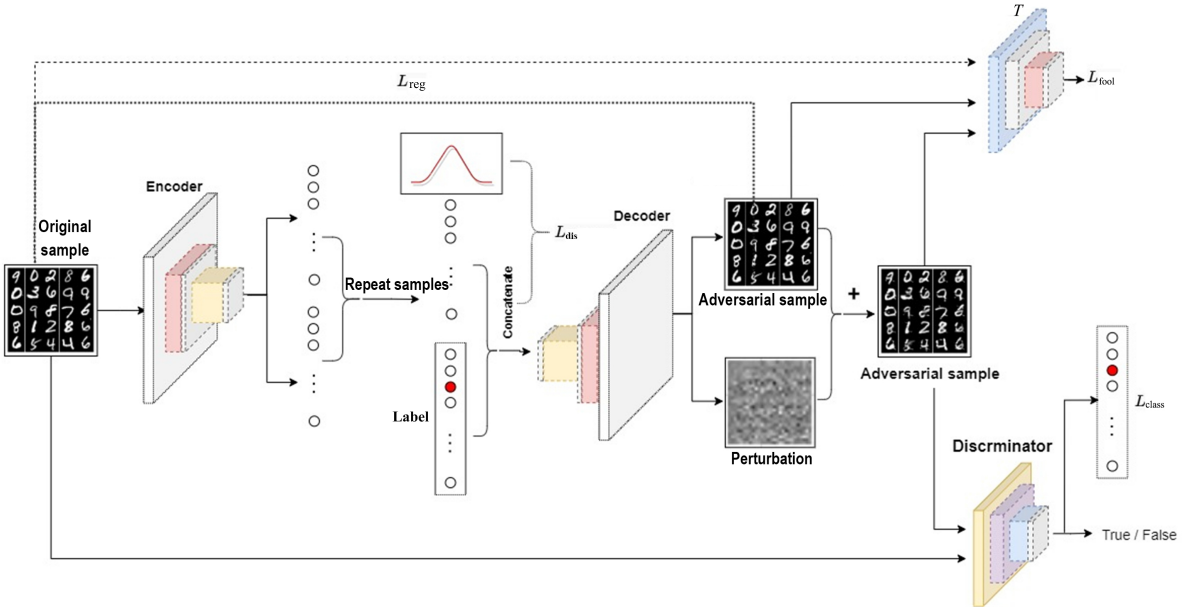


Fig. 3 Structure of DCVAE-adv.

The decoder generates adversarial example $\boldsymbol{x}_{\text{adv}}$ and adversarial perturbation $\boldsymbol{x}_{\text{noise}}$ by sampling from the learned distribution $p(\boldsymbol{x}|\boldsymbol{z}_{\text{noise}}, c)$ and attacks the target network T with adversarial example to ensure the attack ability of the adversarial example, where c is the class of the samples. This step constructs the adversarial samples based on implicit perturbations. Finally, $\boldsymbol{x}_{\text{noise}}$ is added to $\boldsymbol{x}_{\text{adv}}$ to construct the final counter sample $\boldsymbol{x}_{\text{adv}}$, which further improves the attack and migration ability of the adversarial samples. We clip the perturbation $\boldsymbol{x}_{\text{noise}}$ to a small range, so that the visual difference between the original sample and the adversarial sample can be as low as possible. This step constructs the adversarial samples based on explicit perturbations. The decoder learns the real data distribution using the gradient information provided by the discriminator network which identifies whether the input samples are from the original sample or from the adversarial sample. And the discriminator is also responsible for classifying the inputs and assigning higher penalty loss to the reconstructed samples that are inconsistent with the classes of the original samples. By controlling the class information c, the adversarial samples of the specified class are generated.

Because of the gradient vanishing and training instability phenomena during the training of traditional GANs, DCVAE-adv adopts a loss function different from GANs' called cost sensitivity loss to alleviate the occurrence of the above two phenomena. In the initial stage of training, discriminator can easily classify the original samples and the adversarial samples accurately. Therefore, the discriminator cannot provide effective gradient information to update the network parameters of the decoder, which makes the model training process very unstable. To address the above mentioned problem, the weight of discriminant loss term $L_{E_{\text{dis}}}$ (i.e., the loss of discriminator) is initialized as small as possible and is increased with the increasing number of model iterations. This method can effectively overcome the defect of instability caused by poor model generation ability in the initial training stage. Based on this idea, the proposed method solves the following optimization problem:

$$\arg\min_{E_{\boldsymbol{\theta}}, D_{\boldsymbol{\mu}}} E_{c\sim Y, \boldsymbol{x}\sim X} L(\boldsymbol{x}_{\text{adv}}, \boldsymbol{x}; c) \tag{12}$$

where $L(\boldsymbol{x}_{\text{adv}}, \boldsymbol{x}; c)$ is the loss function of the target model with respect to the original samples and the adversarial samples of specific class, X is the probability distribution of the original samples, Y is the set of classes of samples, and $E_{\boldsymbol{\theta}}$ and $D_{\boldsymbol{\mu}}$ are the parameter sets of encoder and decoder, respectively.

To better understand the improved VAE module of DCVAE-adv, the objective functions of each component to optimize are explained in details here. The encoder learns the distribution $p(\boldsymbol{z}_{\text{noise}}|\boldsymbol{x})$ and outputs $\boldsymbol{z}_{\text{noise}}$ that is the feature representation of the original sample in the potential space. The objective function L_E of encoder consists of two terms: L_{dis} and L_{reg}. The L_{dis} loss regularizes the output distribution of encoder to be close to high-dimensional Gaussian distribution as much as possible, while L_{reg} loss makes encoder retain the key

input information as much as possible. Since the second term of Eq. (8) is a constant, L_{reg} is equivalent to PCE, thus we have $L_{\text{reg}} = \text{PCE}(\boldsymbol{x}, \boldsymbol{x}_{\text{adv}})$. We use a hyperparameter τ to balance the effects of the two terms, namely,

$$(\mu_i, \sigma_i) = E(\boldsymbol{x}_i) \tag{13}$$

$$L_{\text{reg}} = \text{PCE}(\boldsymbol{x}, \boldsymbol{x}_{\text{adv}}) \tag{14}$$

$$L_{\text{dis}} = -\frac{1}{2} \times \sum_{i=1}^{d} (\mu_i^2 + \sigma_i^2 - \log \sigma_i^2 - 1) \tag{15}$$

$$L_E = \tau \times L_{\text{dis}} + (1 - \tau) \times L_{\text{reg}} \tag{16}$$

The decoder takes the randomly sampled noise $\boldsymbol{z}_{\text{noise}}$ and class c as inputs to generate adversarial samples $\boldsymbol{x}_{\text{adv}}$ and adversarial perturbations $\boldsymbol{x}_{\text{noise}}$. At the same time, the parameters are updated according to the effective gradient information provided by the discriminator.

To sum up, the loss functions of our improved VAE module include L_{reg}, L_{fool}, and $L_{E_{\text{dis}}}$, and two hyperparameters α and β are used to balance their relative importance.

$$L_D = \alpha L_{\text{reg}} + \beta L_{\text{fool}} + (1 - \alpha - \beta) L_{E_{\text{dis}}} \tag{17}$$

where $\alpha, \beta \in [0, 1]$, $L_{E_{\text{dis}}}$ is given in Eq. (18).

$$L_{E_{\text{dis}}} = E_{z \sim \boldsymbol{z}_{\text{noise}}}[\log(\text{Dis}(D(\boldsymbol{z}, c)) + \text{Loss}(\text{Dis}(D(\boldsymbol{z}, c)), c))] \tag{18}$$

where the first term in Eq. (17) discriminates the real samples from the generated adversarial samples, and the second term in Eq. (17) classifies the generated adversarial samples and penalizes samples with different classes from the real samples. The third term, i.e., Eq. (18), suggests that the adversarial samples generated by the decoder should meet the following two conditions: (1) The discriminator Dis can successfully classify the adversarial samples into class c. (2) The generated adversarial samples can successfully fool the discriminator Dis, and it is hard to distinguish the original samples from the adversarial samples. In other words, the trained model will enable the discriminator Dis to classify the adversarial samples and the original samples with a probability of 0.5.

For non-targeted attacks, it is required that the adversarial samples $\boldsymbol{x}_{\text{adv}}$ can be classified into all other classes except class c, and the loss function for non-targeted attacks is given in Eq. (19).

$$L_{\text{fool}}^{\text{unt}} = \sum_{i=1}^{n} [-\text{Loss}(T(\boldsymbol{x}_{\text{adv}}^i), c) - \text{Loss}(T(\boldsymbol{x}_{\text{adv}}^i + \text{Clip}(\boldsymbol{x}_{\text{noise}}^i, d)), c) + \text{Loss}(T(\boldsymbol{x}^i), c)] \tag{19}$$

where $\text{Clip}(\boldsymbol{x}_{\text{noise}}, d)$ is given in Eq. (20).

$$\text{Clip}(\boldsymbol{x}_{\text{noise}}, d) = d \times \frac{\boldsymbol{x}_{\text{noise}} - \min(\boldsymbol{x}_{\text{noise}})}{\max(\boldsymbol{x}_{\text{noise}}) - \min(\boldsymbol{x}_{\text{noise}})} \tag{20}$$

For the targeted attack, in addition to requiring the target network to incorrectly classify the adversarial samples, the adversarial samples $\boldsymbol{x}_{\text{adv}}$ should be classified into a specified class t, and the loss function for targeted attacks is given in Eq. (21).

$$L_{\text{fool}}^{\text{tar}} = \sum_{i=1}^{n} [\text{Loss}(T(\boldsymbol{x}_{\text{adv}}^i), t) + \text{Loss}(T(\boldsymbol{x}_{\text{adv}}^i + \text{Clip}(\boldsymbol{x}_{\text{noise}}^i, d)), t) + \text{Loss}(T(\boldsymbol{x}^i), c)] \tag{21}$$

where $\text{Loss}(T(\boldsymbol{x}_{\text{adv}}^i), t)$ is given in Eq. (22).

$$\text{Loss}(T(\boldsymbol{x}_{\text{adv}}^i), t) = -\sum_{y=1}^{k} t(\boldsymbol{x}_{\text{adv}}^i, y) \log(T(\boldsymbol{x}_{\text{adv}}^i)) \tag{22}$$

where $t(\boldsymbol{x}_{\text{adv}}^i, y)$ denotes the one-hot encoding of class t. In the proposed model DCVAE-adv, it is required that the adversarial samples constructed in both phases by the decoder should be able to perform the attacks successfully.

The discriminator Dis also classifies the input samples and assigns higher penalty to those adversarial samples whose classes are inconsistent with the those of original samples. The corresponding two loss functions are given in Eqs. (23) and (24).

$$L_{\text{real}} = E_{\boldsymbol{x} \sim X}[\log \text{Dis}(\boldsymbol{x})] + E_{z \sim Z_{\text{noise}}, c \sim Y}[\log(1 - \text{Dis}(D(\boldsymbol{z}, c)))] \tag{23}$$

$$L_{\text{class}} = E_{\boldsymbol{x} \sim X}[\text{Loss}(\text{Dis}(\boldsymbol{x}), c)] + E_{z \sim Z_{\text{noise}}, c \sim Y}[\text{Loss}(\text{Dis}(D(\boldsymbol{z}, c)), c)] \tag{24}$$

The total loss of discriminator Dis is the sum of Eqs. (23) and (24), as shown in Eq. (25).

$$L_{\text{Dis}} = L_{\text{real}} + L_{\text{class}} \tag{25}$$

During the training of DCVAE-adv, to be able to perform more difficult black box attack, we train an approximate model to simulate the network to be attacked. First, the real samples and the corresponding labels are used to train a classifier T. Then DCVAE-adv takes the classifier T as the target network and attacks it, and the target network T would in turn provide gradient information to update the decoder network. Finally, the adversarial samples which successfully attacked the target network T would be used for the black box transfer attack. The pseudo-code of the algorithm for training DCVAE-adv is given in Algorithm 1.

Algorithm 1 Training algorithm for DCVAE-adv

Input:

Training set $S = (\boldsymbol{x}, y)$, size of minibatch b, attack target class t, parameters α, β, τ, and r, and the number of iterations epochs.

Output:

Encoder parameter $E_{\boldsymbol{\theta}}$, decoder parameter $D_{\boldsymbol{\theta}}$, discriminator parameter $\text{Dis}_{\boldsymbol{\rho}}$, and target network T_τ.

Initialize the model parameter $E_{\boldsymbol{\theta}}$, $D_{\boldsymbol{\theta}}$, $\text{Dis}_{\boldsymbol{\rho}}$, and T_τ;
For (numbers of epochs)
  Randomly sample a mini-batch with b original examples $\{\boldsymbol{x}^1_{\text{clean}}, \boldsymbol{x}^2_{\text{clean}}, \ldots, \boldsymbol{x}^b_{\text{clean}}\}$;
  Randomly sample a mini-batch with b labels $\{y^1, y^2, \ldots, y^b\}$;
  $L_T = -\sum_{i=1}^{b}\sum_{y=1}^{k} T_\tau(\boldsymbol{x}^i, y)\log(T_\tau(\boldsymbol{x}^i))$;
  Update the target network parameters T_τ with the stochastic gradient descent algorithm;
  $(\mu_i, \sigma_i) = E(\boldsymbol{x}_i)$;
  $\boldsymbol{z}_i \sim N(\boldsymbol{0}, \boldsymbol{I})$;
  $(\boldsymbol{x}^i_{\text{adv}}, \boldsymbol{x}^i_{\text{noise}}) = D(\mu_i, \sigma_i \cdot \boldsymbol{z}_i)$;
  $L_{\text{reg}} = \text{PCE}(\boldsymbol{x}, \boldsymbol{x}_{\text{adv}})$;
  $L_E = -\frac{1}{2}\tau\sum_{i=1}^{d}(\mu_i^2 + \sigma_i^2 - \log\sigma_i^2 - 1) + (1-\tau)L_{\text{reg}}$;
  Update the target network parameters $E_{\boldsymbol{\theta}}$ with the stochastic gradient descent algorithm;
  $\boldsymbol{x}^i_{\text{adv\_two}} = \boldsymbol{x}^i_{\text{adv}} + \text{Clip}(\boldsymbol{x}^i_{\text{noise}}, d)$;
  $L_{E_{\text{dis}}} = E_{\boldsymbol{x}^i_{\text{adv}}\sim\boldsymbol{x}_{\text{adv}}}[\log(\text{Dis}(\boldsymbol{x}^i_{\text{adv\_two}})) + \text{Loss}(\text{Dis}(\boldsymbol{x}^i_{\text{adv\_two}}, c))]$;
  $L_{\text{fool}} = \sum_{i=1}^{n}\text{Loss}(T_\tau(\boldsymbol{x}^i_{\text{adv}}), t) + \text{Loss}(T_\tau(\boldsymbol{x}^i_{\text{adv\_two}}), t) + \text{Loss}(T_\tau(\boldsymbol{x}^i), y^i)$;
  If (numbers%10 = 0)
    r =(numbers+0.1)/(epochs+1);
  $L_D = \alpha L_{\text{reg}} + \beta L_{\text{fool}} + (1-\alpha-\beta)rL_{E_{\text{dis}}}$;
  Update the target network parameters D_μ with the stochastic gradient descent algorithm;
  $L_{\text{real}} = E_{\boldsymbol{x}^i\sim\boldsymbol{x}}[\log(\text{Dis}(\boldsymbol{x}^i))] + E_{\boldsymbol{x}^i_{\text{adv}}\sim\boldsymbol{x}_{\text{adv}}}[\log(1-\text{Dis}(\boldsymbol{x}^i_{\text{adv}}))]$;
  $L_{\text{Dis}} = L_{\text{real}} + E_{\boldsymbol{x}^i\sim\boldsymbol{x}, y^i\sim y}[\log(\text{Dis}(\boldsymbol{x}^i, y^i))]$;
  Update the target network parameters $\text{Dis}_{\boldsymbol{\rho}}$ with the stochastic gradient descent algorithm;
Return $E_{\boldsymbol{\theta}}$, $D_{\boldsymbol{\theta}}$, $\text{Dis}_{\boldsymbol{\rho}}$, and T_τ.

4 Experimental Result

To verify the effectiveness and efficiency of the proposed approach, we conduct extensive experiments on three datasets, MNIST[51], Fashion-MNIST[52], and CIFAR10[53]. The performance of the proposed approach is evaluated quantitatively and qualitatively. The quantitative evaluation focuses on the attack success rate of DCVAE-adv on different attack networks, whereas the qualitative evaluation focuses on the generation quality of adversarial samples by DCVAE-adv on different datasets. All experiments are implemented in the same experimental environment. The software environment is Window10+Python3.7+Tensorflow2.1, and the hardware environment is Intel(R) I5-6600K CPU, 16.0 GB memory, and Nvidia GTX3050 GPU.

In our experiments, we train DCVAE-adv to learn the probability distribution of the generated adversarial samples rather than the probability distribution of the original samples, and the trained DCVAE-adv is used to generate the desired adversarial samples. In the black box attack scenario, we train a network (original network, e.g., LeNet5) by the proposed method to generate adversarial samples, and then use the generated adversarial samples to attack another network (transfer attack network, e.g., AlexNet). In other words, the adversarial samples generated by one network are used to attack another different network. The higher the attack success rate is, the better the transferability of the adversarial samples is. The better the transferability of the generated adversarial samples, the stronger the ability of the adversarial samples to carry out black box attack. In the following, we will present the details of our experiments.

4.1 Network structure design for three datasets

We test the proposed model DCVAE-adv on the three datasets mentioned above. For MNIST and Fashion-MNIST datasets, LeNet5[54], AlexNet[2], and VGG[3] are selected as the target networks to be attacked. For training DCVAE-adv, the batch size is 256, and the learning rate is 1×10^{-4}. The Adam optimizer is used to optimize the target loss. For CIFAR10 dataset, we use three ResNet networks as target models to be attacked, which are ResNet20[4], ResNet32[4], and Wide ResNet28 (WResNet28)[55]. For training DCVAE-adv, the batch size is 128, the learning rate is 1×10^{-4}, and the target loss is optimized using the RMSprop optimizer. With respect to the design of network structure, we adopt different network structures for different datasets. In our experiments, we found that it is not true that the more complex the model structure, the higher the quality of the generated adversarial examples[56, 57]. Based on this principle, the network structures we designed for the three datasets, MNIST, Fashion-MNIST, and CIFAR10, are given in Table 1.

In the experiment, we first use the pre-trained network to simulate the target attack network on three datasets, and then use the gradient descent algorithm to solve the optimal parameters of the target network. The classification accuracy of different classifiers on the three test sets is given in Table 2.

Table 1 Design of network structure for three datasets MNIST, Fashion-MNIST, and CIFAR10.

Component	MNIST and Fashion-MNIST	CIFAR10
Encoder	Conv2D (16, 2×2, 2), BN, Tanh Conv2D (32, 2×2, 2), BN, Tanh Conv2D (64, 2×2, 2), BN, Tanh Dense (512), BN, Tanh Dense (10) Dense (10)	Conv2D (16, 2×2, 2), BN, Tanh Conv2D (32, 2×2, 2), BN, Tanh Conv2D (64, 2×2, 2), BN, Tanh Dense (512), BN, Tanh Dense (10) Dense (10)
Decoder	Dense (128), BN, LeakyReLU Dense (256), BN, LeakyReLU Dense (512), BN, LeakyReLU, Reshape (4, 4, 32) Conv2DTranspose (32, 3×3, 1), BN, Tanh Conv2DTranspose (64, 2×2, 1), BN, Tanh Conv2DTranspose (128, 3×3, 2), BN, Tanh Conv2D (1, 5×5, 2), BN, Tanh	Dense (128), BN, LeakyReLU Dense (256), BN, LeakyReLU Dense (512), BN, LeakyReLU, Reshape (4, 4, 32) Conv2DTranspose (32, 3×3, 1), BN, Tanh Conv2DTranspose (64, 2×2, 1), BN, Tanh Conv2DTranspose (128, 3×3, 2), BN, Tanh Conv2D (1, 5×5, 2), BN, Tanh Conv2DTranspose (128, 3×3, 2), BN, Tanh Conv2D (1, 5×5, 2), BN, Tanh
Discriminator	Conv2D (32, 3×3, 2), Relu Conv2D (64, 3×3, 2), Relu, Flatten Dense (10) Dense (1)	Conv2D (32, 3×3, 2), Relu Conv2D (64, 3×3, 2), Relu, Flatten Dense (10) Dense (1)

Table 2 Recognition accuracy of attack networks on different datasets.

Dataset	Attack network	Recognition accuracy (%)
MNIST	LeNet5	99.32
	AlexNet	99.04
Fashion-MNIST	LeNet5	91.08
	AlexNet	91.32
	VGG	93.70
CIFAR10	ResNet20	90.16
	ResNet32	91.15
	WResNet28	95.54

Due to the simple sample distribution of MNIST dataset, even simple LeNet network can achieve nearly 100% classification accuracy. Compared with LeNet5 and AlexNet networks, VGG network is deeper and uses multiple continuous small convolutional kernels instead of large convolutional kernels, thus achieving the highest classification accuracy on the Fashion-MNIST dataset. In the CIFAR10 dataset, due to the high diversity of the overall sample, it is more challenging compared with the first two datasets, so we choose three more complex residual networks. Residual network is mainly composed of multiple residual units, which can effectively alleviate the problem of vanishing gradient caused by too many network layers. Among the three residual networks, WResNet28 achieves the optimal classification accuracy of 95.54% on the test set.

4.2 Attack effect on the original network to be attacked

In this section, we demonstrate the attack effect of the method proposed in this paper on the original network to be attacked without adversarial training. We select LeNet5, ResNet20, ResNet32, and WResNet28 as the networks to be attacked on MNIST, Fashion-MNIST, and CIFAR10 datasets, respectively. For different classes, we construct adversarial samples to attack the corresponding original networks. When the attack network and transfer attack network are the same network, we call it non-transfer attack, which is often less difficult. When the attack network and transfer attack network are different, we call it transfer attack or adversarial sample transfer attack. The transferability of adversarial samples measures the ability of adversarial samples designed for one network successfully attacking other networks. The black box attack is difficult since the network structure and the parameters are not accessible. However, we could transform the black box attack into a white box attack by training an alternative model to approximate the target black box network, and get the adversarial samples which successfully attacked the alternative model to attack the target black box network. Accordingly, the higher the transferability of adversarial sample is, the stronger the black box attack capability of the corresponding attack method is. However, the transfer attack capability of the adversarial samples generated with previous GAN-based methods is weak. In the experiment, we focused on the targeted attack with the adversarial samples generated by DCVAR-adv. Figure 4 demonstrates the experimental results with target class 8. It is observed from Fig. 4 that the adversarial samples generated by DCVAE-adv are clear

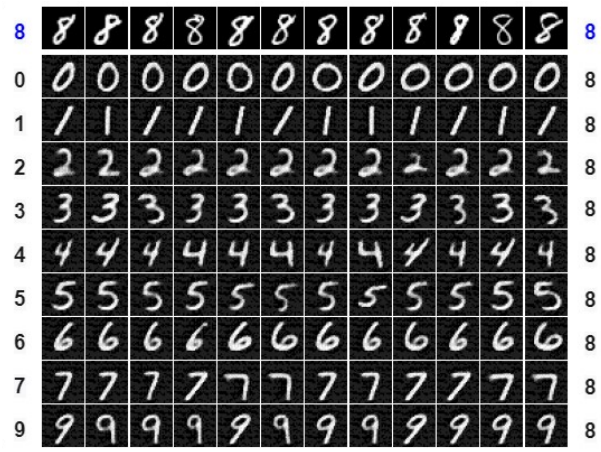


Fig. 4 Experimental results of targeted attack on MNIST (target class 8).

and realistic with less visual difference compared with the original samples.

In Fig. 4, the first row shows the real examples from the target class 8. The rest rows show the adversarial examples generated by DCVAR-adv. And the numbers on the left side show the real classes for each row, and the numbers on the right side show the classes predicted by the target network.

Table 3 shows the success rate of attack against samples constructed for different classes on dataset MNIST. When the attack network is LeNet5, both transfer attack and non-transfer attack have achieved nearly 100% success rate. When the attack network is AlexNet, transfer attack has not achieved desired results. Relatively speaking, slightly successful transfer attacks are mainly concentrated on the samples of class 5, which may be due to the small visual difference between numbers 5 and 8. The model can fool the target network through simple modification of number 5 and transfer to other networks more easily.

The experimental results of targeted attack on the dataset Fashion-MNIST is given in Fig. 5. Because classes 5 and 9 belong to the same superclass as class 7, the success rates of attack and transfer attack of the adversarial samples generated by classes 5 and 9 are higher than that of others, which can be further confirmed by the experimental results in Table 4. For targeted attack, not all classes can produce reliable and

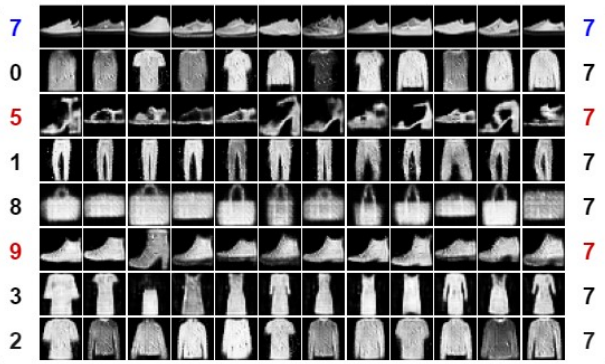


Fig. 5 Experimental results of targeted attack on Fashion-MNIST (target class 7).

Table 3 Experimental results of targeted attacks and transfer attacks on MNIST (target class 8).

Attack network (AN)	Transfer AN	Attack success rate (%)									
		0	1	2	3	4	5	6	7	9	Mean
T_{LeNet5}	LeNet5	100	93.1	97.0	100	100	**100**	100	100	100	98.9
	AlexNet	100	100	100	100	100	**100**	100	100	99.1	99.8
T_{AlexNet}	LeNet5	0	0	1.6	0	0	**12.9**	1.9	0	0.9	1.9
	AlexNet	99.1	100	100	100	99.1	**100**	100	100	99.2	97.7

Note: $\{0, 1, \ldots, 9\}$ are original class labels.

Table 4 Experimental results of targeted attacks and transfer attacks on Fashion-MNIST (target class 7).

Attack network (AN)	Transfer AN	Attack success rate (%)									
		0	1	2	3	4	5	6	8	9	Mean
T_{LeNet5}	LeNet5	100	98.0	100	99.0	100	98.9	100	99.1	100	99.4
	AlexNet	5.9	2.9	13.8	6.9	20.7	28.7	17.8	62.3	49.5	23.6
	VGG	36.6	41.5	22.7	25.7	41.5	100	45.5	90.1	85.1	54.3
T_{AlexNet}	LeNet5	9.1	0.8	0	0	0	42.3	0.9	0	69.1	13.8
	AlexNet	96.3	94.1	100	94.5	96.1	99.2	100	99.1	100	97.6
	VGG	20.7	36.6	7.9	26.7	7.9	89.1	3.9	37.6	78.2	34.8
T_{VGGNet}	LeNet5	0	0	0	0	0	44.5	4.6	0	0	5.4
	AlexNet	3.2	1.6	0	18.6	0	17.8	1.6	13.8	78.8	15.4
	VGG	97.2	100	99.1	99.2	100	98.0	99.1	100	97.1	98.8

Note: $\{0, 1, \ldots, 9\}$ are original class labels.

effective adversarial samples with good transferability. And the classes more semantically similar to the target class can produce adversarial samples which can successfully attack the target network with a higher probability. For instance, as shown in Fig. 5, sandals (class 5) and ankle boots (class 9) are very similar to the target class sneakers (class 7), thus it would be easier to introduce imperceptible perturbations to these two classes to successfully transform the predicted class into the target class.

In Fig. 5, the first row shows the real examples from the target class 7. The rest rows show the adversarial examples generated by DCVAR-adv. And the numbers on the left side show the real classes for each row, and the numbers on the right side show the classes predicted by the target network.

The experimental results of targeted attack on the dataset CIFAR10 are shown in Fig. 6, and the target class is class 7. The first and third rows are the original samples and their classes, and the second and fourth rows are the corresponding adversarial samples generated by the proposed method. It is observed from Fig. 6 that the background of the generated adversarial sample images is relatively clear, while the instances (e.g., horse and car) are blurry. This is because in order to improve the attack success rate, the instances located in the center of the images are blurred when the adversarial samples are generated. The experimental results of targeted transfer attack on the CIFAR10 dataset (target class is class 7) are shown in Table 5. It is observed from Table 5 that, for different neural networks, ResNet20, ResNet32, and WResNet28, the average attack success rates are 92.8%, 93.7%, and 96.4%, respectively, and the average transfer attack success rates are 16.7%, 24.3%, 26.9%, 26.6%, 17.6%, and 12.2% on this complex dataset, respectively. Overall, the proposed method shows good performance.

Fig. 6 Experimental results of targeted attack on CIFAR10 (target class 7).

4.3 Experimental comparison with related methods

In this section, we experimentally compare DCVAE-adv with FGSM, AdvGAN, and AdvGAN++ in two settings: non-targeted attacks and targeted attacks. For the three datasets, we choose different networks as attack networks. For MNIST dataset, we choose LeNet5 and AlexNet as attack networks; for Fashion-MNIST dataset, we choose LeNet5, AlexNet, and VGG as attack networks; for CIFAR10 dataset, we choose ResNet20, ResNet32, and WResNet28 as the attack networks. The process of adversarial attack includes three steps: (1) training DCVAE-adv model, (2) generating adversarial samples with the trained DCVAE-adv model, and (3) using the generated adversarial samples to attack the corresponding networks and transfer to other attack networks. It should be noted that for the three state-of-the-art methods, FGSM is only applicable to non-targeted attacks, while AdvGAN and AdvGAN++ are applicable to both targeted and non-targeted attacks. The FGSM and AdvGAN are methods constructing adversarial samples by explicitly adding perturbations, and their attack effect depends mainly on the scale of perturbations. Figure 7 shows the attack success

Table 5 Experimental results of targeted attacks and transfer attacks on CIFAR10 (target class 7).

Attack network (AN)	Transfer AN	Attack success rate (%)									
		0	1	2	3	4	5	6	8	9	Mean
T_{ResNet20}	ResNet20	93.9	93.1	94.9	91.0	91.0	91.1	95.9	95.9	88.9	92.8
	ResNet32	9.1	10.9	20.1	25.1	7.1	34.1	3.9	7.0	33.1	16.7
	WResNet28	14.0	11.9	34.1	46.0	23.9	46.9	5.0	15.0	21.9	24.3
T_{ResNet32}	ResNet20	11.9	24.7	29.7	47.5	20.8	29.7	0.9	9.9	45.5	26.9
	ResNet32	92.0	92.1	95.1	93.1	93.1	91.0	96.1	97.1	94.1	93.7
	WResNet28	18.0	11.8	19.8	73.2	3.9	49.5	15.8	18.8	28.7	26.6
$T_{\text{WResNet28}}$	ResNet20	30.1	10.6	10.6	31.7	10.6	34.9	7.3	15.4	7.3	17.6
	ResNet32	28.5	0.8	19.5	8.9	15.4	27.6	1.6	3.2	4.6	12.2
	WResNet28	96.7	91.1	99.2	96.7	99.1	98.1	96.7	96.2	94.3	96.4

Note: $\{0, 1, \ldots, 9\}$ are original class labels.

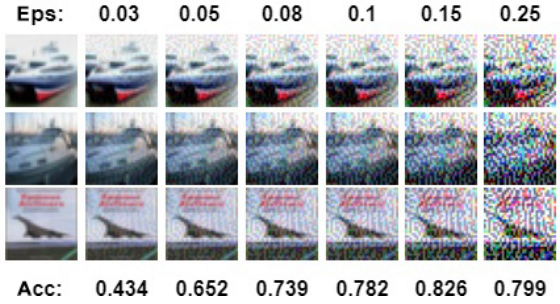


Fig. 7 Comparison of adversarial samples generated by FGSM at different perturbation scales.

rates of FGSM at different perturbation scales. In Fig. 7, Eps and Acc represent perturbation scale and attack success rate, respectively. When the perturbation scale (i.e., Eps) gradually increases from 0.03 to 0.15, the attack success rate (i.e., Acc) gradually increases to 82.6%. However, when the perturbation scale continues to increase, the success rate of attack does not continue to improve but appears to a certain degree of decline. In our experiments, when adversarial samples are generated on MNIST and Fashion-MNIST datasets, the perturbation scale of FGSM is set to 15%, and the perturbation pixel value of AdvGAN is limited to $[-0.2, 0.2]$. When adversarial samples are generated on CIFAR10 dataset, the perturbation scale of FGSM is set to 10%, and the perturbation pixel value of AdvGAN is limited to $[-0.1, 0.1]$. The experimental results of non-targeted and targeted attacks are listed in Tables 6 and 7, respectively. Figure 8 shows an intuitive comparison of the adversarial samples generated by different attack methods.

In Tables 6 and 7, when the attack network and transfer attack network are the same network, the attack is not transfer attack. When the attack network is different from the transfer attack network, the attack is transfer attack. And normally transfer attack is more difficult than non-transfer attack. Table 6 shows that our proposed method provides the best results in the vast majority of cases with three exceptions: (1) When AlexNet was transferred to LeNet5 on MNIST dataset, FGSM achieves the highest success rate of 13.8%, while DCVAE-adv achieved attack success rate of 12.5% with little difference between them. (2) On the Fashion-MNIST dataset, AdvGAN++ achieved the best attack success rate of 97.5% when AlexNet performed non-transfer attack, while the success rate of DCVAE-adv was 95.9%, with only 1.6% difference. (3) On the CIFAR10 dataset, AdvGAN++ achieved the highest success rate of 95.1% when ResNet20 performed non-transfer attack, while DCVAE-adv achieved a success

Table 6 Experimental results of non-targeted attacks.

Dataset	Attack network (AN)	Transfer AN	Attack success rate (%)			
			FGSM	AdvGAN	AdvGAN++	DCVAE-adv
MNIST	LeNet5	LeNet5	66.4	78.1	97.2	**98.1**
		AlexNet	65.1	55.2	47.1	**71.5**
	AlexNet	LeNet5	**13.8**	1.6	10.3	12.5
		AlexNet	88.6	97.5	95.1	**97.1**
Fashion-MNIST	LeNet5	LeNet5	74.7	91.8	96.7	**99.9**
		AlexNet	62.6	51.2	28.9	**74.7**
		VGG	71.5	64.2	45.5	**74.3**
	AlexNet	LeNet5	50.4	47.1	39.2	**55.2**
		AlexNet	63.7	82.9	**97.5**	95.9
		VGG	59.3	46.3	48.7	**65.8**
	VGG	LeNet5	53.6	60.9	35.7	**67.5**
		AlexNet	54.4	52.8	31.7	**66.1**
		VGG	73.9	91.8	86.9	**92.6**
CIFAR10	ResNet20	ResNet20	82.1	90.2	**95.1**	90.2
		ResNet32	71.4	84.5	89.4	**95.7**
		WResNet28	73.2	87.8	85.3	**89.8**
	ResNet32	ResNet20	81.9	78.2	92.6	**93.6**
		ResNet32	83.9	80.4	93.4	**95.7**
		WResNet28	69.9	75.6	85.3	**85.8**
	WResNet28	ResNet20	78.5	78.8	86.6	**91.3**
		ResNet32	73.1	64.2	87.9	**88.5**
		WResNet28	89.3	87.4	94.2	**97.9**

Table 7 Experimental results of targeted attacks.

Dataset	Attack Network (AN)	Transfer AN	Attack success rate (%)		
			AdvGAN	AdvGAN++	DCVAE-adv
MNIST	LeNet5	LeNet5	70.7	93.4	**98.9**
		AlexNet	56.1	12.1	**99.8**
	AlexNet	LeNet5	**3.3**	4.1	1.9
		AlexNet	86.2	93.4	**97.7**
Fashion-MNIST	LeNet5	LeNet5	95.9	94.3	**99.4**
		AlexNet	20.3	4.8	**23.6**
		VGG	51.2	2.4	**54.3**
	AlexNet	LeNet5	11.4	12.1	**13.8**
		AlexNet	94.3	95.4	**97.6**
		VGG	**38.2**	8.9	34.8
	VGG	LeNet5	0.8	**10.5**	5.4
		AlexNet	13.8	4.6	**15.4**
		VGG	97.5	95.9	**98.8**
CIFAR10	ResNet20	ResNet20	50.4	87.8	**92.8**
		ResNet32	2.4	**17.8**	16.7
		WResNet28	0.8	**31.7**	24.3
	ResNet32	ResNet20	12.2	24.1	**26.9**
		ResNet32	92.1	92.5	**93.7**
		WResNet28	13.1	23.1	**26.6**
	WResNet28	ResNet20	0.1	14.3	**17.6**
		ResNet32	0.8	**18.4**	12.2
		WResNet28	89.4	92.5	**96.4**

Fig. 8 Visualizations of the adversarial samples generated by different attack methods.

rate of 90.2%, with 4.9% difference. Table 7 shows the experimental results of targeted attacks. The FGSM is not applicable to the targeted attacks thus is not compared against. Similar to the experimental results of non-targeted attacks listed in Table 6, the proposed method provides the best results in the vast majority of cases with six exceptions.

Because targeted attacks are more difficult than non-targeted attacks, making targeted transfer attacks even more difficult, the success rate of targeted transfer attacks is much lower than that of non-targeted attacks. However, it is observed from Tables 6 and 7 that for MNIST dataset, the transfer attack success rate of targeted attack was higher than that of non-targeted attack when the adversarial samples are transferred from the LeNet5 network to the AlexNet network. And it is observed that DCVAE-adv attempted to construct adversarial samples with pixel values in the interval [0.34, 0.87] to attack the target network. And these adversarial samples can attack the target network effectively with a high attack success rate and transfer to other networks easily. In addition, we also found from Table 6 that on the CIFAR10 dataset, the success rate of non-transfer attack on ResNet20 network generated by DCVAE-adv was 90.2%, while the success rate of transfer attack on the target network ResNet32 was 95.1%. The possible reason could be that the CIFAR10 dataset is relatively complex, contains 10 classes, and has a high diversity of images, which makes it difficult to learn the data distribution. Therefore, it is inevitable that some samples are of poor generation quality, and the poor quality adversarial samples may be correctly identified by the attacking network and incorrectly identified by the transfer attack networks. The visualizations of the experimental results in Tables 6 and 7 are given in Figs. 9 and 10, respectively.

To demonstrate the differences between the adversarial samples generated by different methods more intuitively, Fig. 8 shows adversarial samples

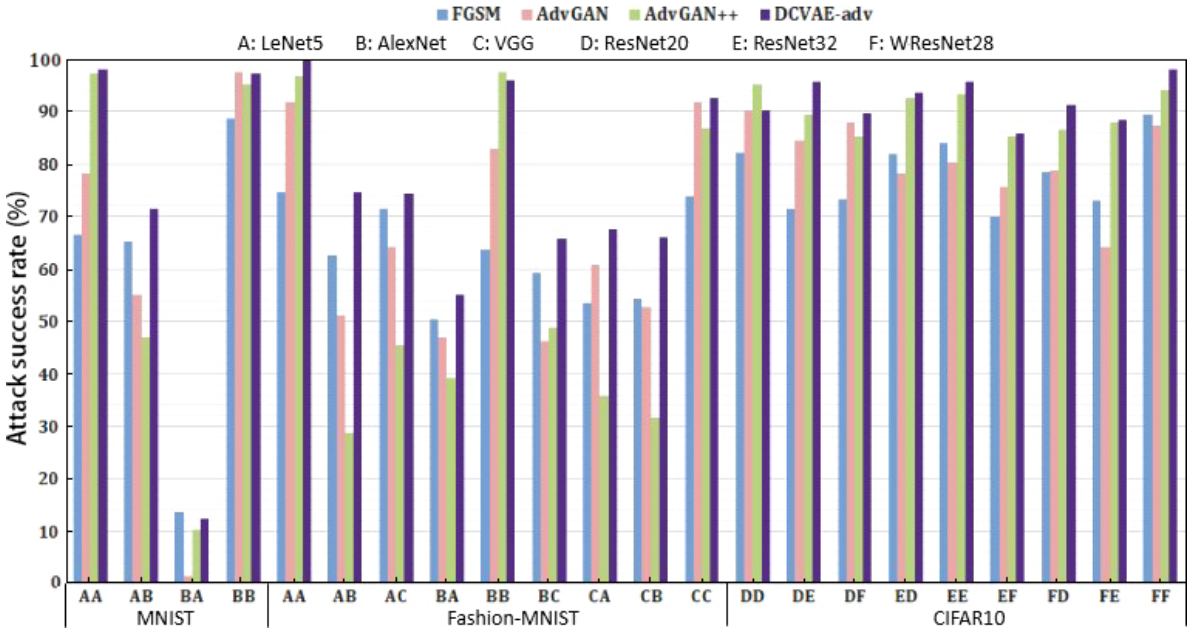


Fig. 9 Visualizations of the experimental results given in Table 6.

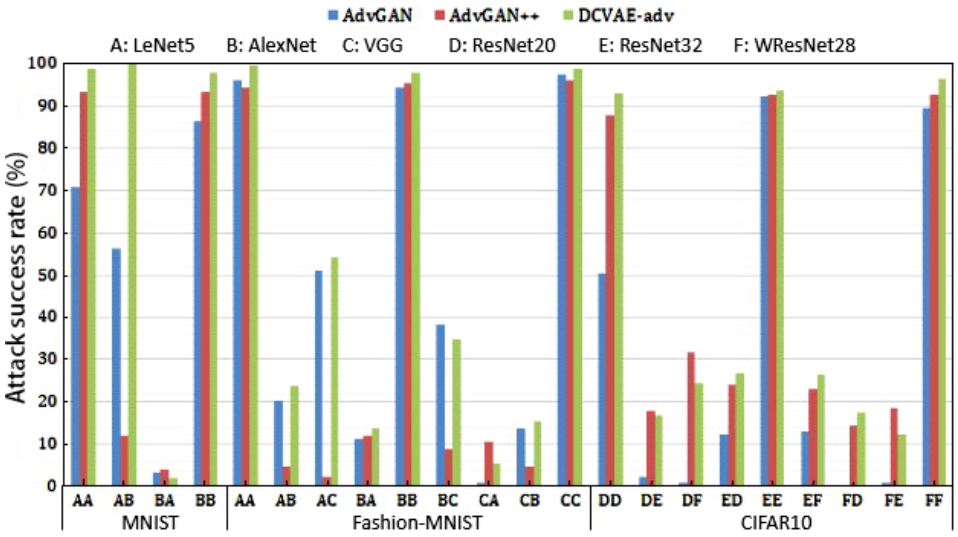


Fig. 10 Visualizations of the experimental results given in Table 7.

generated by different methods on MNIST dataset. It is observed that for FGSM baseline, the adversarial perturbations almost cover the whole generated adversarial samples. The adversarial samples generated by AdvGAN++ are very competitive in most classes, but for some classes the generation quality is poor, such as numbers 2, 3, and 4. The third and fourth rows are adversarial samples generated by AdvGAN at a perturbation scale of 0.1 and 0.2, respectively. The former has better imperceptibility. The adversarial samples generated by DCVAE-adv are shown in the last row. The generated samples are more realistic and sharper compared with the adversarial samples generated by other methods.

4.4 Experimental comparison of generation speed

As for the efficiency, we compared the time required to generate 1000 adversarial samples on the three datasets, MNIST, Fashion-MNIST, and CIFAR10 by the four methods, DCVAE-adv, FGSM, AdvGAN, and AdvGAN++. The experimental results are listed in Table 8.

Table 8 Speed of generating adversarial examples with different methods.

Dataset	Time of generating adversarial examples (s)			
	FGSM	AdvGAN	AdvGAN++	DCVAE-adv
MNIST	10.02	**0.090**	0.128	0.097
Fashion-MNIST	10.12	0.111	0.207	**0.102**
CIFAR10	18.66	0.215	0.302	**0.198**

It is observed from Table 8 that DCVAE-adv is the most efficient approach, as the speed of adversarial sample generation by DCVAE-adv is faster than the other three methods on two datasets, Fashion-MNIST and CIFAR10, whereas on dataset MNIST, the speed of AdvGAN is faster among the four methods, yet the speed of DCVAE-adv is almost the same as that of AdvGAN.

5 Conclusion

A novel adversarial example generation method DCVAE-adv combining variational inference with adversarial learning is proposed in this paper. DCVAE-adv has three advantages: (1) It is suitable not only for white box attacks but also for black box attacks, since it integrates both explicit and implicit perturbations. (2) It is memory efficient and time efficient for adversarial example generation, since no original examples are needed to generate adversarial examples. (3) It generates more realistic and sharper adversarial samples achieving a higher success rate of attacks and transfer attacks compared with prior methods. The promising future works of this study include: (1) Investigate the interpretability of adversarial examples. (2) Is there any essential difference in the quality of adversarial examples generated by different probabilistic generation models? For example, generative adversarial network, variational autoencoder, and diffusion model. Is there a unified defense method against these adversarial examples? (3) The adversarial examples are well designed, and not just any clean original examples can be obtained by adding a subtle perturbation. How to select the clean examples that can generate adversarial examples?

Acknowledgment

This work was supported by the Key R&D Program of Science and Technology Foundation of Hebei Province (No. 19210310D) and the Natural Science Foundation of Hebei Province (No. F2021201020).

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] A. Krizhevsky, I. Sutskeve, and G. E. Hinton, ImageNet classification with deep convolutional neural networks, in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, Lake Tahoe, NV, USA, 2012, pp. 1097–1105.

[3] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv: 1409.1556v2, 2015.

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

[5] M. D. Zeiler and R. Fergus, Visualizing and understanding convolutional networks, in *Proc. 13th Eur. Conf. Comput. Vis.*, Zurich, Switzerland, 2014, pp. 818–833.

[6] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, Squeeze-and-excitation networks, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 8, pp. 2011–2023, 2020.

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, Going deeper with convolutions, in *Proc. 2015 IEEE Conf. Comput. Vis. Pattern Recognit.* (*CVPR*), Boston, MA, USA, 2015, pp. 1–9.

[8] F. Chollet, Xception: Deep learning with depthwise separable convolutions, in *Proc. 2017 IEEE Conf. Comput. Vis. Pattern Recognit.* (*CVPR*), Honolulu, HI, USA, 2017, pp. 1800–1807.

[9] M. Tan and Q. V. Le, EfficientNet: Rethinking model scaling for convolutional neural networks, in *Proc. 36th Int. Conf. Mach. Learn.* (*ICML2019*), Long Beach, CA, USA, 2019, pp. 6105–6114.

[10] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, and I. Goodfellow, Intriguing properties of neural networks, presented at 2014 2nd Int. Conf. Learn. Represent. (ICLR2014), Banff, Canada, 2014.

[11] X. Yuan, P. He, Q. Zhu, and X. Li, Adversarial examples: Attacks and defenses for deep learning, *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2805–2824, 2019.

[12] J. Zhang and C. Li, Adversarial examples: Opportunities and challenges, *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 7, pp. 2578–2593, 2020.

[13] N. Akhtar, A. Mian, N. Kardan, and M. Shah, Advances in adversarial attacks and defenses in computer vision: A survey, *IEEE Access*, vol. 9, pp. 155161–155196, 2021.

[14] K. Ren, T. Zheng, Z. Qin, and X. Liu, Adversarial attacks and defenses in deep learning, *Engineering*, vol. 6, no. 3, pp. 346–360, 2020.

[15] A. Serban, E. Poll, and J. Visser, Adversarial examples on object recognition: A comprehensive survey, *ACM Comput. Surv.*, vol. 53, no. 3, pp. 1–38, 2020.

[16] A. Kurakin, I. J. Goodfellow, and S. Bengio, Adversarial examples in the physical world, presented at 2017 Int. Conf. Learn. Represent. (ICLR2017), Toulon, France, 2017.

[17] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, The limitations of deep learning in adversarial settings, in *Proc. 2016 IEEE Eur. Symp. Secur. Priv.* (*EuroS&P*), Saarbruecken, Germany, 2016, pp. 372–387.

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

[19] L. Gao, Z. Huang, J. Song, Y. Yang, and H. T. Shen, Push & pull: Transferable adversarial examples with attentive attack, *IEEE Trans. Multimed.*, doi: 10.1109/TMM.2021.3079723.

[20] A. Chaturvedi and U. Garain, Mimic and fool: A task-agnostic adversarial attack, *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 4, pp. 1801–1808, 2021.

[21] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, Generative adversarial nets, in *Proc. 27th Int. Conf. Neural Infor. Process. Syst.*, Montreal, Canada, 2014, pp. 2672–2680.

[22] D. P. Kingma and M. Welling, Auto-encoding variational bayes, presented at 2014 Int. Conf. Learn. Represent., Banff, Canada, 2014.

[23] X. Liu and C. J. Hsieh, Rob-GAN: Generator, discriminator, and adversarial attacker, in *Proc. 2019 IEEE/CVF Conf. Comput. Vis. Pattern Recognit.* (*CVPR*), Long Beach, CA, USA, 2019, pp. 11226–11235.

[24] J. Chen, H. Zheng, H. Xiong, S. Shen, and M. Su, MAG-GAN: Massive attack generator via GAN, *Inf. Sci.*, vol. 536, pp. 67–90, 2020.

[25] P. Yu, K. Song, and J. Lu, Generating adversarial examples with conditional generative adversarial net, in *Proc. 2018 24th Int. Conf. Pattern Recognit.* (*ICPR*), Beijing, China, 2018, pp. 676–681.

[26] 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.* (*IJCAI-18*), Stockholm, Sweden, 2018, pp. 3905–3911.

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

[28] T. Yu, S. Wang, C. Zhang, Z. Wang, Y. Li, and X. Yu, Targeted adversarial examples generating method based on cVAE in black box settings, *Chin. J. Electron.*, vol. 30, no. 5, pp. 866–875, 2021.

[29] U. Upadhyay and P. Mukherjee, Generating out of distribution adversarial attack using latent space poisoning, *IEEE Signal Process. Lett.*, vol. 28, pp. 523–527, 2021.

[30] I. J. Goodfellow, J. Shlens, and C. Szegedy, Explaining and harnessing adversarial examples, presented at 3rd Int. Conf. Learn. Represent. (ICLR2015), San Diego, CA, USA, 2015.

[31] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, Synthesizing robust adversarial examples, in *Proc. 35th Int. Conf. Mach. Learn.*, Stockholm, Sweden, 2018, pp. 284–293.

[32] T. Deng and Z. Zeng, Generate adversarial examples by spatially perturbing on the meaningful area, *Pattern Recognit. Lett.*, vol. 125, pp. 632–638, 2019.

[33] Y. Xu, B. Du, and L. Zhang, Self-attention context network: Addressing the threat of adversarial attacks for hyperspectral image classification, *IEEE Trans. Image Process.*, vol. 30, pp. 8671–8685, 2021.

[34] P. Vidnerová and R. Neruda, Vulnerability of classifiers to evolutionary generated adversarial examples, *Neural Netw.*, vol. 127, pp. 168–181, 2020.

[35] T. Karras, S. Laine, and T. Aila, A style-based generator architecture for generative adversarial networks, in *Proc. 2019 IEEE/CVF Conf. Comput. Vis. Pattern Recognit.* (*CVPR*), Long Beach, CA, USA, 2019, pp. 4396–4405.

[36] M. Y. Zhai, L. Chen, F. Tung, J. He, M. Nawhal, and G. Mori, Lifelong GAN: Continual learning for conditional image generation, in *Proc. 2019 IEEE/CVF Int. Conf. Comput. Vis.* (*ICCV*), Seoul, Republic of Korea, 2019, pp. 2759–2768, 2019.

[37] M. Y. Zhai, L. Chen, J. He, M. Nawhal, F. Tung, and G. Mori, Piggyback GAN: Efficient lifelong learning for image conditioned generation, in *Proc. 16th Eur. Conf. Comput. Vis.*, Glasgow, UK, 2020, pp. 397–413.

[38] M. Y. Zhai, L. Chen, and G. Mori, Hyper-LifelongGAN: Scalable lifelong learning for image conditioned generation, in *Proc. 2021 IEEE/CVF Conf. Comput. Vis. Pattern Recognit.* (*CVPR2021*), Nashville, TN, USA, 2021, pp. 2246–2255.

[39] K. Sohn, X. Yan, and H. Lee, Learning structured output representation using deep conditional generative models, in *Proc. 28th Int. Conf. Neural Infor. Process. Syst.* (*NIPS*), Montreal, Canada, 2015, pp.3483–3491.

[40] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, Beta-VAE: Learning basic visual concepts with a constrained variational framework, presented at 5th Int. Conf. Learn. Represent., Toulon, France, 2017.

[41] H. Shao, Z. Xiao, S. Yao, D. Sun, A. Zhang, S. Liu, T. Wang, J. Li, and T. Abdelzaher, ControlVAE: Tuning, analytical properties, and performance analysis, *IEEE Trans. Pattern Anal. Mach. Intell.*, doi: 10.1109/TPAMI.2021.3127323.

[42] Z. Zhao, D. Dua, and S. Singh, Generating natural adversarial examples, presented at 6th Int. Conf. Learn. Represent., Vancouver, Canada, 2018.

[43] Y. Song. R. Shu, N. Kushman, and S. Ermon, Constructing unrestricted adversarial examples with generative models, in *Proc. 32nd Int. Conf. Neural Infor. Process. Syst.* (*NeurIPS 2018*), Montréal, Canada, 2018, pp. 8322–8333.

[44] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, A general framework for adversarial examples with objectives, *ACM Trans. Priv. Secur.*, vol. 22, no. 3, pp. 1–30, 2019.

[45] J. Liu, Y. Tian, R. Zhang, Y. Sun, and C. Wang, A two-stage generative adversarial networks with semantic content constraints for adversarial example generation, *IEEE Access*, vol. 8, pp. 205766–205777, 2020.

[46] P. Tang, W. Wang, J. Lou, and L. Xiong, Generating adversarial examples with distance constrained adversarial imitation networks, *IEEE Trans. Dependable Secur. Comput.*, doi: 10.1109/TDSC.2021.3123586.

[47] C. Hu, X. Wu, and Z. Li, Generating adversarial examples with elastic-net regularized boundary equilibrium generative adversarial network, *Pattern Recognit. Lett.*, vol. 140, pp. 281–287, 2020.

[48] W. Zhang, Generating adversarial examples in one shot with image-to-image translation GAN, *IEEE Access*, vol. 7, pp. 151103–151119, 2019.

[49] W. Jiang, Z. He, J. Zhan, W. Pan, and D. Adhikari, Research progress and challenges on application-driven adversarial examples: A survey, *ACM Trans. Cyber Phys. Syst.*, vol. 5, no. 4, p. 39, 2021.

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

2651.

[51] L. Deng, The MNIST database of handwritten digit images for machine learning research, *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, 2012.

[52] H. Xiao, K. Rasul, and R. Vollgraf, Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms, arXiv preprint arXiv: 1708.07747, 2017.

[53] A. Torralba, R. Fergus, and W. T. Freeman, 80 million tiny images: A large data set for nonparametric object and scene recognition, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 11, pp. 1958–1970, 2008.

[54] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[55] S. Zagoruyko and N. Komodakis, Wide residual networks, in *Proc. British Mach. Vis. Conf.* (*BMVC 2016*), York, UK, 2016, pp. 1–12.

[56] C. Z. Wang, X. Lv, W. P. Ding, and X. D. Fan, No-reference image quality assessment with multi-scale weighted residuals and channel attention mechanism, *Soft Comput.*, vol. 26, pp. 13449–13465, 2022.

[57] X. Lv, C. Wang, X. Fan, Q. Leng, and X. Jiang, A novel image super-resolution algorithm based on multi-scale dense recursive fusion network, *Neurocomputing*, vol. 489, pp. 98–111, 2022.

Junhai Zhai received the BS degree in mathematics and the MS degree in computing mathematics from Lanzhou University, Lanzhou, China in 1988 and 2000, respectively, and the PhD degree in optical engineering from Hebei University, Baoding, China in 2010. He is currently a professor with the College of Mathematics and Information Science, Hebei University, Baoding, China. His main research interests include machine learning, deep learning, and big data processing.

Lei Xu received the BS degree in software engineering from Hebei University, Baoding, China in 2017. He is currently pursuing the MS degree in the College of Mathematics and Information Science, Hebei University, Baoding, China. His main research interests include machine learning and deep learning.