Applied Intelligence (2022) 52:1289–1305
https://doi.org/10.1007/s10489-021-02371-w

A novel approach to generating high-resolution adversarial examples

Xianjin Fang[1] · Zhiwei Li[1] · Gaoming Yang[1]

Accepted: 18 March 2021 / Published online: 19 May 2021

Abstract
Deep neural networks (DNNs) have improved expressive performance in many artificial intelligence (AI) fields in recent years. However, they can easily induce incorrect behavior due to adversarial examples. The state-of-the-art strategies for generating adversarial examples were established as generative adversarial nets (GAN). Due to a large amount of data and the high computational resources required, previous GAN-based work has only generated adversarial examples for small datasets, resulting in a less favorable visualization of the generated images. To address this problem, we propose a feasible approach, which improves on the AdvGAN framework through data augmentation, combined with PCA and KPCA to map the input instance's main features onto the latent variables. Experimental results indicate that our approach can generate more natural perturbations on high-resolution images while maintaining 96% + of the features of the original input instance. Moreover, we measured 90.30% attack success rates on CIFAR-10 against the target model ResNet152, a small improvement compared to 88.69% for AdvGAN. We applied the same idea to ImageNet and LSUN, and the results showed that it not only achieves a high attack success rate,but can generate strongly semantically adversarial examples with better transferability on prevailing DNNs classification models. We also show that our approach yields competitive results compared to sensitivity analysis-based or optimization-based attacks notable in the literature.

Keywords Adversarial examples · Generative adversarial nets · Semiwhite-box attack · Computer vision

1 Introduction

Recent tremendous achievements in deep neural networks (DNNs) have led to significant breakthroughs in computer vision [1, 2], speech recognition [3], natural language processing [4], and web mining [5]. However, Szegedy et al. [6] proposed that by adding minor perturbations that are difficult for humans to perceive on the pixels of the original benign instances, deep learning models can be induced to misclassify the input instance into other categories, resulting in a remarkable reduction in performance. These adversarial examples [6] can also be understood simply as a class of synthetic samples that make the deep learning models erroneous. Nguyen et al. [7] proposed that deep learning models can classify with high confidence some instances that are not recognizable to humans, which implies that deep learning models are incredibly vulnerable. There is a theoretical possibility of passing the classification system by generating garbage or malicious examples. An adversary can interfere with an AI service's reasoning process to accomplish an attack effect, such as evading detection by constructing adversarial examples. By studying adversarial examples, we hope to discover blind spots in the deep learning model and improve the overall model's robustness.

With the introduction of adversarial examples, algorithms around generating adversarial examples began to proliferate, and various schools of thought emerged. This phenomenon has significant practical implications for both attack and defense aspects of deep learning. The attack strategies can be classified into sensitivity analysis-based algorithms, optimization-based algorithms, and generative-based algorithms [8].

Sensitivity analysis Adversarial attacks use sensitivity analysis—a class of algorithms used to determine the

✉ Gaoming Yang
gmyang@aust.edu.cn

Xianjin Fang
xjfang@aust.edu.cn

Zhiwei Li
lzwhappiness@gmail.com

1 Anhui University of Science and Technology, Huainan 232000, China

contribution of each input feature to the output—to find sensitive features and perturb them. There are more representative algorithms in this category, such as FGSM [9], JSMA [10], and PGD [11]. These attacks are, in general, faster than optimization-based attacks. Therefore, they are better suited to be incorporated into the training process of deep learning models and improve their robustness. They are also more straightforward than optimization-based attacks.

Optimization Adversaries use optimization algorithms to search for solutions, alternative forms, or constraints. Some of the more representative algorithms are L-BFGS [6], Deep Fool [12], C&W [13], etc. Although sensitivity analysis-based attacks are more common, these require full knowledge of the system under attack. In contrast to sensitivity analysis-based methods, optimization-based attacks are used more in black-box scenarios.

Generative Here the probability distribution of adversarial perturbations is learned using generative models and used to sample new adversarial examples. A representative generative model is the generative adversarial network (GAN) [14], which differs from the optimization-based and the sensitivity analysis-based methods above. Once the trained generator learns the input instances' distribution, it can generate large numbers of adversarial perturbations in a short period. The more typical algorithms here are AdvGAN [15], Natural GAN [16], Rob-GAN [17], etc.

Notably, the GAN-based algorithms are trained with small and simple datasets, such as MNIST, CIFAR-10, and Tiny-ImageNet (64 px). They suffer from poor visualization of the generated adversarial examples. Furthermore, by increasing the number of iterations or adjusting the relevant hyperparameters, the training model may be overfitted on such a small dataset. It resulted in better performance under a white-box attack but worse when the generated adversarial examples are transferred to other models for testing. In this paper, we make improvements to the AdvGAN [15] structure. First, we perform data augmentation [18] to eliminate overfitting of the training process. Then we train on CIFAR-10 to verify the feasibility of our proposed approach. Finally, the high-resolution adversarial examples generated on ImageNet and LSUN as shown in Fig. 1.

Our approach does not require any access to the target model during the test stage, as we only query the target model during training. Hence, our method is a semiwhite-box attack and is more potent than previous white-box attacks.

1.1 Contributions and organization

We summarize our main contributions as follows:

(1) We first apply PCA and KPCA to the training of GAN-based attacks for generating high-resolution adversarial perturbations. Experimental results show that these two methods can improve the attack success rate by nearly 3.3% on ImageNet and 2.4% on LSUN over AdvGAN [15]. And the proposed approach can reduce the time consumption of generating adversarial examples.
(2) Our approach fills the gap of using GAN to generate high-resolution adversarial examples on large-scale datasets with powerful computational power. And the generated adversarial examples have a strong semantic correlation with the original input by some image evaluation metrics.
(3) We report the results (of our approach and other attack algorithms) in an untargeted attack. Our approach performs well in attack success rate, time-consumed to

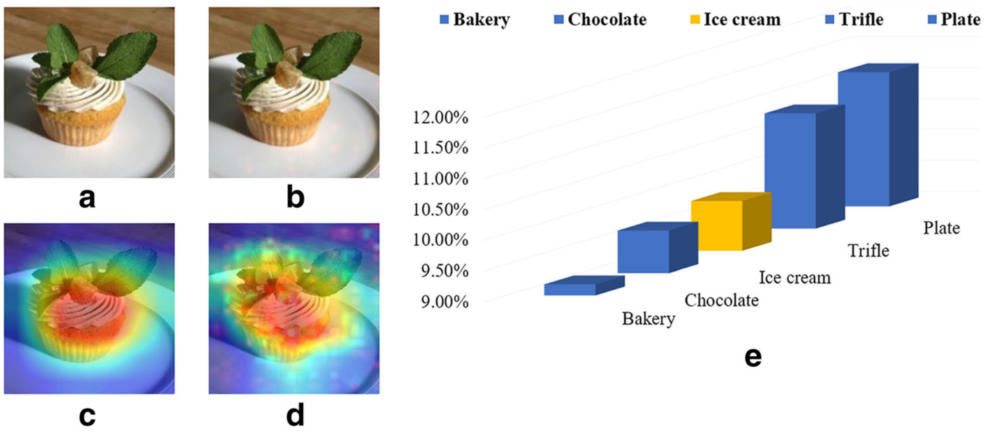

Fig. 1 Adversarial examples generated by our approach (targeted model: ResNet152) (**a**). Original image on ImageNet with the ground truth '*Ice cream.*' (**b**). Adversarial image generated by our approach with the predict label *'Plate.'* (**c**). Crad-CAM of original image. (**d**) Crad-CAM of the adversarial image. (**e**) Top 5 predict classification confidence of adversarial image (The figure-colored yellow is ground truth)

generate an adversarial example, image visualization, semantic correlation, and transferability.

The rest of this paper is organized as follows. Section 2 gives the related concept, such as data augmentation, generative adversarial nets, etc. In section 3, we illustrate our proposed approach in data processing, network structure, objective function, and models. Section 4 experimentally evaluates the proposed method. Finally, the conclusion and future work directions were given in section 5.

2 Related work

2.1 Data augmentation

Deep learning algorithms are more dependent on massive data compared to traditional machine learning algorithms. Extensive work and studies have shown that the training set's size and quality directly affect the performance of deep learning algorithms [19]. When the number and quality of images are higher, the further and deeper image features that the deep learning model can learn, the better the network model's classification and recognition capability will be. A series of picture-enhancing processes, such as expansion and transformation of the amount of data used, are collectively named data augmentation [18]. Therefore, we should perform data augmentation on some datasets with relatively small sample sizes and poor quality. For example, the ImageNet has a small number of images in each subclass. When a certain class samples available are very limited, it is inevitable to be data unbalanced [41]. Some studies have given solutions to this problem [40, 41]. In this paper data augmentation is needed to reduce the GAN overfitting in the learning data distribution.

What makes data augmentation so crucial for training neural networks is that overfitting is easily caused when the dataset's size does not match the deep network. An efficient way to mitigate the overfitting phenomenon is to adjust the dataset's size and quality through data augmentation. Image data augmentation is divided into offline augmentation and online augmentation. Offline augmentation means that when the dataset size is small, the dataset is directly augmented before training, and the number of data points can be increased by the artificial augmentation factor or multiplied. Online augmentation refers to the way in which the network model, after accessing the dataset, implements data augmentation for each object in the dataset using predefined augmentation module [20–22], such as rotation, translation, mirror flip, resize and padding transformation, noise transformation, etc.

2.2 Generative adversarial nets

Generative Adversarial Nets [14] (GAN) consists of a generator G, and a discriminator D. The generator G bears an adversarial relationship with the discriminator D. The GAN simultaneously trains both G and D: G obtains the true data distribution, and D evaluates the probability that the instance is derived from the training data rather than from G. The training process of G is designed to maximize the probability that D evaluates the error. GAN is also referred to as a maximal, minimal game. The optimization to be solved becomes:

$$\min_G \max_D V(D,G) = E_x[\log D(x)] + E_z[\log(1-D(G(z)))] \tag{1}$$

In Eq. (1), z is the prior distribution, such as Gaussian distribution and Uniform distribution; E is the computed expectation. The objective of GAN is to learn the distribution p_g of the training data, to learn this distribution, the input noise variable $p_z(z)$ is defined, which is mapped to the data space $G(z;\theta_g)$, where G is a generative model consisting of a multilayer perceptual network with θ_g as a parameter. In addition, the discriminator $D(x;\theta_d\ )$ is defined to determine whether the input data are from the generator or the training data, and the output of $D(x)$ is the probability that x is from the training data rather than G. Training D aims to maximize the assignment of the correct label to either the training data or the G-generated data.

The primary problem that hindered the original GAN is the non-convergence problem, where the purpose of the GAN is to reach a Nash equilibrium. The behaviors of G and D may counterbalance each other, as an optimizing algorithm that descends along a gradient on one side may lead to an increase in error on the other side. In practice, GAN typically suffer from oscillations, which means that the network vacillates between samples of various generative modes, thus failing to reach some sort of equilibrium. A common phenomenon is that GAN map multiple different inputs to the same output, which is a nonconvergent situation known as mode collapse [14, 23]. Next, the original GAN can only generate continuous data rather than discrete data (e.g., natural language). Because after each update of the generator, the output is a gradient between the previous output and the discriminator's return, and its output must be continuously differentiable. Theoretically, due to the original GAN that uses the Jensen-Shannon (JS) divergence as a metric for generating examples. Generated discrete data cannot be achieved even using a continuous approach such as word distribution or embedding. Finally, GAN's evaluation issue is more complicated than with other generative models; i.e., there is no uniform measure.

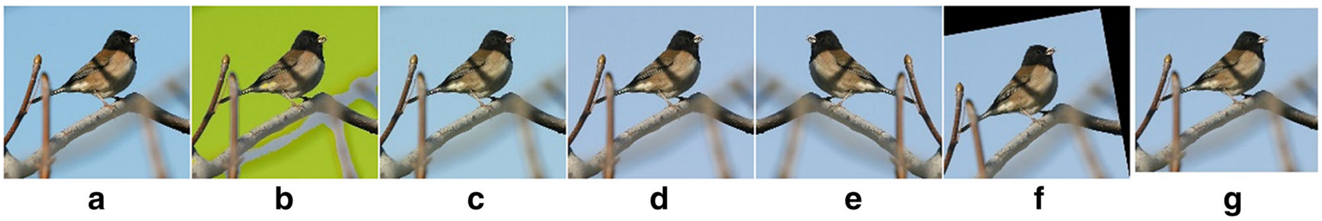

Fig. 2 Image data augmentation. **a** Original image, **b** Hue Shift, **c** Saturation Shift, **d** Brightness Shift, **e** Mirror Flip [35], **f** Rotation, **g** Random Padding [36]

Faced with the deficiencies of the original GAN, the researcher improved the GAN's generative mechanisms by utilizing deep learning achievements on supervised learning tasks. The improvements mainly include the following.

(1) **New network structures**, such as LGGAN [24] consider learning the scene generation in a local context, and correspondingly design a local class-specific generative network with semantic maps as a guide to provide more scene details. CycleGAN [25] present an approach for learning to translate an image from a source domain X to a target domain Y in the absence of paired examples. Which goal is to learn a mapping $G: X \rightarrow Y$ such that the distribution of images from $G(X)$ is indistinguishable from the distribution Y using an adversarial losses.

(2) **Adding regularization constrains**, such as SN-GAN [26], performs corresponding operations on each layer of D, which can significantly improve the generation of GAN and is one of the few GAN structures that can generate all 1000 classes of ImageNet objects using a single network. WGAN-GP [27] further improves the stability of the network by adding gradient penalties, etc.

(3) **Integrating multiple models**, such as the AdaGAN [28], T generator models are trained in sequence by an algorithm similar to AdaBoost. During the t-step training step, the models that failed the previous time are given increased weights. The generators in Stack GAN [29] consist of multiple submodels in series.

(4) **Changing optimization algorithms**, such as Seq GAN [30], treat the sequence generation problem as a sequential decision problem and use RNN as the generation network, Mask GAN [31] uses seq2seq [32] as the generation network to make GAN capable of filling in words, among other improvements.

3 Methodology

For a given classifier f, which can classify $x \in X$ into ground truth $y^{\text{true}}=f_{\theta}(x)$. The purpose of the adversarial attacks is to train the model to generate adversarial examples $x^{*}=x+\delta_x$ that will be identified by f as some other label y^{pred}, where δ_x is a tiny and human imperceptible perturbation added to the original image. In this paper, $\delta_x = G(z|x)$, where G is the generator. The objectives to be optimized are as follows:

$$f_{\theta}(x+\delta_x) = y^{pred} \neq y^{true} \tag{2}$$

$$\text{s.t.} \quad L_p = \left\|x - x^{*}\right\|_p = \left(\sum_{i=1}^{n} \left|x_i - x_i^{*}\right|^p\right)^{\frac{1}{p}} \leq \epsilon \tag{3}$$

In Eq. (3), we wish adversarial example x^{*} to be as similar to x as possible, and ϵ is the maximum magnitude $\|\cdot\|_p$ perturbation allowed. We consider the most commonly used L_0, L_2 and L_∞ distances, each of which has a significant physical interpretation.

We will then discuss our proposed methodology in four aspects: dataset preprocessing, network structure, objective function, and models.

3.1 Data preprocessing

We work with the datasets ImageNet,LSUN andCIFAR-10, where ImageNet contains 1,386,167 images of 1000 classes. The dataset is partitioned into 1,281,167 training images, 50,000 validation images and 100,000 test images. Faced with such a large scale of data, we are unable to use the entire dataset. In this experiment, we selected 140 classes of data in the training set. From total 182,000 images (with each class in the dataset consisting of 1300 images), 7000 images for the validation set, and 14,000 images for the test set. LSUN contains around one million labeled images for each of 10 scene categories and 20 object categories. In this experiment, we selected 'bus', 'cow', and 'sheep' from the 20 object categories. Totally 210,000 images (70,000 images for each category) are used as the training set, 15,000 images for the validation set, and 30,000 images for the test set. For CIFAR-10, in the training phase, our training set was 50,000 images. We further split the test set (10,000 images) into 5000 validation data and 5000 test data.

Algorithm 1: **Data augmentation of original images**

Input: Original image x_i from the training set, Length of the training set Len_{dst}, Mirror flip function $f_{Mir}()$,

Rotation function $f_{Rot}()$,Padding border function $f_{Bor}()$, Random function $R()$.

Output: Transformed Images: x_{Hue}, x_{Sat}, x_{Bri}, x_{Mir}, x_{Rot}, x_{Pad}

```
1 While $i \leq Len_{dst}$ do:
2     Samples the data and normalizes the image;
3     Get the shape of the images, $x_{i\_row}$ and $x_{i\_col}$;
4     for the number of training sets do:
5         #Shift in the attributes of the image itself (Hue shift: each pixel of an image is randomly added or subtracted at (-3,3). Saturation shift: the value of each pixel of an image is randomly multiplied by(0.95,1.05). Brightness shift: Similar to saturation shift.)
6         $x_{Hue} \leftarrow x_i[:,:,0] + R(-3,3)$
7         $x_{Sat} \leftarrow x_i[:,:,1] * R(0.95,1.05)$
8         $x_{Bri} \leftarrow x_i[:,:,2] * R(0.95,1.05)$
9         #Shift in the presentation of the image (Mirror flip: mirror flip of the image. Rotation: random rotation within an angle of (-15,15).)
10        $x_{Mir} \leftarrow f_{Mir}(x_i)$
11        $x_{Rot} \leftarrow f_{Rot}(x_i, R(-15,15))$
12        #Shift in the shape of the image (Padding: randomly pads the vertical and horizontal parts of the image with white space.)
13        $Pad_{vert} \leftarrow [R(x_{i\_row}/10)]$
14        $Pad_{hori} \leftarrow [R(x_{i\_col}/10)]$
15        $x_{i\_row} \leftarrow x_{i\_row} - Pad_{vert} + R(Pad_{vert})$
16        $x_{i\_col} \leftarrow x_{i\_col} - Pad_{hori} + R(Pad_{hori})$
17        $x_{Pad} \leftarrow f_{Bor}(x_{i\_row}, x_{i\_col}, Pad_{vert}, Pad_{hori})$
18     Resize images to the same size
19 End
```

In our experiment, an effective way to prevent overfitting of the model is data augmentation, as shown in Fig. 2. The specific steps of the transformation are illustrated in Algorithm 1. It is worth noting that all the techniques we refer to in this paper for processing images after Algorithm 1 are called data augmentation. During the training process, the images in the training set above are transformed by Algorithm 1 so that the images' number in each original subclass is expanded by seven times. By such image data augmentation, a more sophisticated data distribution will be obtained. Since our experiments are GAN-based to generate adversarial examples, the merit of more complex data distributions enables a more robust trained model with good generalization performance.

3.2 Network structure

The overall structure of our proposed approach for generating adversarial examples is shown in Fig. 3. It consists of a feature extraction function F_x, a generator G, a discriminator D, and a target function F_t. For a given original input set X, our goal is to generate an adversarial perturbation $\delta_{x'}$ through G, which is limited to a certain order of magnitude. This perturbation is spliced with x' to generate an adversarial image x^*. This image can spoof discriminator D and be misclassified by the target function F_t in an untargeted attack.

During the training phase, to make the generated adversarial example meet the above requirements, we perform data augmentation on the original input set X to obtain X'As described in Algorithm 1. Then we use VGG19 as the feature extraction function to perform image feature extraction on the input x', which eliminates the need to follow the encoder-decoder infrastructure, thereby reducing the training and inference overhead. Before generating the adversarial perturbations, x' is mapped to the corresponding noise vector z with the linear dimensionality reduction method of PCA and the non-linear dimensionality reduction method of KPCA, which can generate a more natural image with strong semantic correlations to the original input. Finally, G receives feature $F_x(x')$ of image x' and a noise vector z (as a concatenated vector) and generates $\delta_{x'}$. The corresponding adversarial example is $x^*=x'+\delta_{x'}$.

3.3 Objective function

Data augmentation loss, the distribution of images obtained after data augmented by Algorithm 1 is somewhat different from the original images. In this experiment, we use the Cross Entropy of the model on the distribution of transformed and original images as the data augmentation loss. The Cross Entropy loss describes the distance between two probability distributions, and when the Cross Entropy is smaller it means that the two are closer to each other. When the Cross Entropy between the original image distribution and the transformed image distribution, i.e., the data augmentation loss, reaches convergence in the training process, then we can use the transformed images for generating adversarial perturbations. This process can be demonstrated in Algorithm 2, where the whole process is divided into two main steps: (1) calculating the Cross Entropy loss of the two distributions on the feature extraction model; (2) calculating the loss on the target model. Moreover, when the whole training process reaches convergence, we can also determine the relevant parameters (in Table 2) of the training process by the variation of the loss function (in Fig. 4) on different data sets.

Algorithm 2: Calculate the data augmentation loss

Input: Training epoch N, the iterations of each epoch n, batch size m, data augmentation function T, feature extract model F_x , target model F_t, loss function L, the ground-truth l

Output: Data augmentation loss: L_{DA}

```
1 Custom weights initialization called on the feature extract model $F_x$ and the target model $F_t$
2 While $L_{DA}$ has not converged do:
3    for $i = 0 \to N$ do
4       for $j = 0 \to n$ do
5          Load a batch of original images $\{x_i\}_{i=1\sim m}$ and ground truth $l$
6          $x' \leftarrow T(x)$ #Image data augmentation, includes rotation, hue shift, mirror flip, etc.
7          #train the feature extract model
8          $F_{x\_model} \leftarrow F_x(x_i')$
9          $L_{feature} \leftarrow$CrossEntropy($F_{x\_model}, l$)
10         $F_x() \leftarrow F_{x\_model}$ #Update the feature extract model
11         #train the target model
12         $F_{t\_model} \leftarrow F_t(x_i')$
13         $L_{target} \leftarrow$CrossEntropy($F_{x\_model}, l$)
14         $F_t() \leftarrow F_{t\_model}$ #Update the target model
15      end for
16   save the feature extract model $F_x$ and the target model $F_t$
17   $L_{DA} \leftarrow L_{feature} + L_{target}$
18   end for
19 End
```

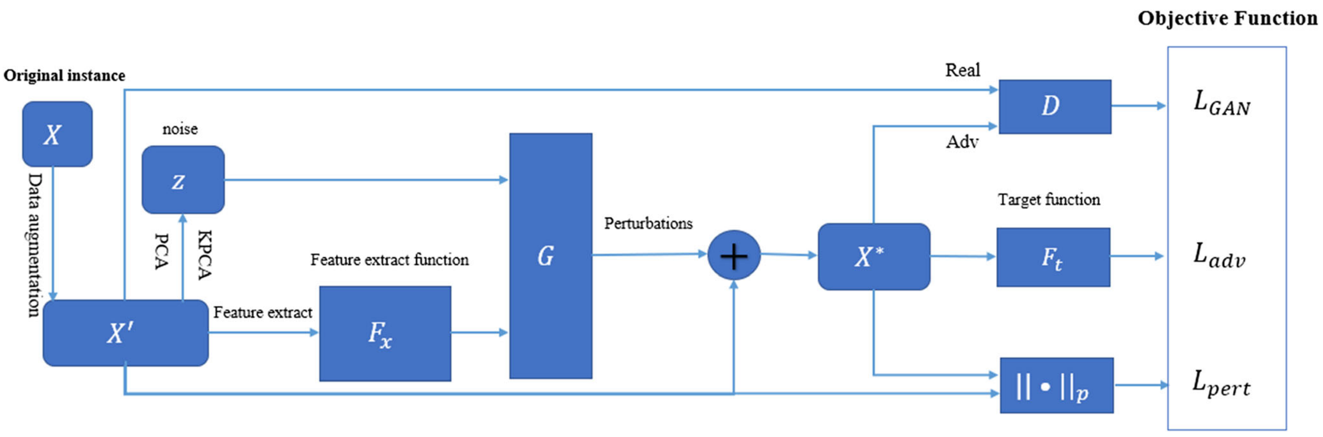

Fig. 3 The overall structure of our proposed approach

GAN loss, our work proposes using Mean Square Error (MSE) loss to detect deviations between predicted and ground-truth labels. We divide GAN training into two processes, training discriminator D and training generator G, as shown in Algorithm 3. For the discriminator, we expect D to maximize the probability to distinguish whether the input instance is from the original image or the generated image, so the training process should minimize the loss from clean images and maximize the loss from generated images. In optimizing the loss function, we set the false sample label to '0' and the true sample label to '1'. Mathematically, we train the discriminator D to maximize:

$$L_{D\_fake} = E_z\left[\left(D\left(G\left(z|F_x\left(x'\right)\right)+x'\right)-0\right)^2\right] \tag{4}$$

And minimize:

$$L_{D\_real} = E_{x'}\left[\left(D\left(x'\right)-1\right)^2\right] \tag{5}$$

For the generator, we expect the generated samples to spoof the discriminator to the extent possible, so during the training of G, we minimize its loss function:

$$L_G = E_z\left[D\left(\left(G\left(z|F_x\left(x'\right)\right)+x'\right)-1\right)^2\right] \tag{6}$$

The total GAN loss is as follows:

$$L_{GAN} = L_{D\_real} - L_{D\_fake} + L_G \tag{7}$$

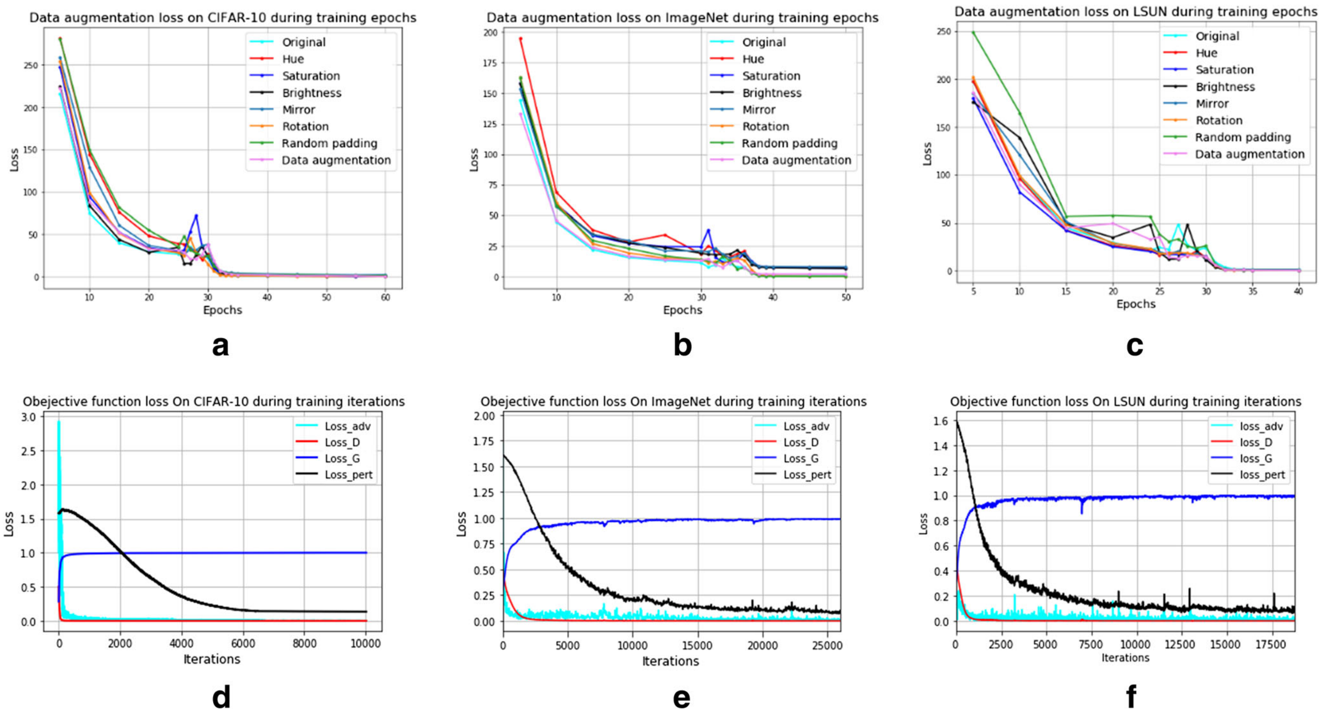

Fig. 4 Some losses during training

Adversarial loss, the loss for spoof the target model F_t in an untargeted attack is:

$$L_{adv} = E_{x',z}\left[F_t\left(G\left(z|F_x\left(x'\right)\right) + x', l'\right)\right] \tag{8}$$

where l' is any class that is different from the ground-truth label l of x'.

Perturbation loss, the magnitudes of perturbations are essential for making the output similar to the original images. In Eq. (3), Lp is used to measure the distance (or similarity) between x' and x^*, and the usual choices of p are $[0, 2, \infty]$. L_0 represents the number of the different pixels between the benign image and adversarial example. L_2 measures the standard Euclidean distance between the benign image and adversarial example. L_∞ represents the maximum value of the imperceptible perturbation in the adversarial example. Our approach refers to [33], which combines L_2 and L_∞, the result will lead to better perceptual quality. The perturbation loss is the following:

$$L_{pert} = \lambda_1 {*} E_{x'}\left[\|x^* - x'\|_2\right] + \lambda_2 {*} E_{x'}\left[\|x^* - x'\|_\infty\right] \tag{9}$$

$$\text{s.t.} \quad \lambda_1 + \lambda_2 = 1 \tag{10}$$

In summary, the goal of our approach is to minimize the following objective function:

$$L = L_{GAN} + \alpha {*} L_{adv} + \beta {*} L_{pert} \tag{11}$$

where α and β are the relative importance of each objective.

Algorithm 3: Training algorithm for proposed GAN

Input: Training epoch N, the iterations of each epoch n, batch size m, data augmentation function T, feature extract model F_x , target model F_t, loss function L, the ground-truth l, the predicted label l'.

Output: Objective functions: L_{GAN}; L_{adv}; L_{pert}

```
1 Custom weights initialization called on G and D; Load pretrained model $F_x$, $F_t$
2 While training loss has not converged do:
3   for $i = 0 \to N$ do
4     for $j = 0 \to n$ do
5       Load a batch of original images $\{x_i\}_{i=1\sim m}$ and ground truth $l$
6       Sample noise $\{z_i\}_{i=1\sim m}$ from PCA and KPCA mapping
7         $x' \leftarrow T(x)$ #Image data augmentation
8         $z \leftarrow PCA(z)$ or $z \leftarrow KPCA(z)$ #Mapping
9         $x^* \leftarrow G(z, F_x(x')) + x'$ #Generating adversarial examples
10        $x^* \leftarrow clip(x^*)$
11        #Optimize D
12        if $j \bmod n == 0$
13          $L_{D\_fake} \leftarrow max\, D(x^*, l)$
14        else
15          $L_{D\_fake} \leftarrow max\, D(x^*, l')$
16        end if
17        $L_{D\_real} \leftarrow min\, D(x', l)$
18        #Optimize G
19        $L_G \leftarrow min\, L(x^*, l)$
20        $L_{GAN} \leftarrow L_{D\_fake} - L_{D\_real} + L_G$
21        $L_{adv} \leftarrow L(F_t(x^*, l'))$
22        $L_{pert} \leftarrow \lambda_1 * L_2(x^*, x') + \lambda_2 * L_\infty(x^*, x')$
23    end for
24  end for
25 End
```

3.4 Models

In our work, we refer to the training manner of pix2pix. We utilize the network architecture in [15]. The difference is that we alter the corresponding parameters for the training process to ensure that the network architecture can fit the input images' size. More details about the generator and discriminator architecture can refer to [15].

For the feature extraction model, we use the pretrained VGG19, and for the target model, we use ResNet152. It can be noticed that the model we have chosen here has two characteristics: deeper model depth and a pretraining. The deeper model can extract more input features and improve the whole network's generalization performance. A pretrained model reduces the training overhead by verifying the transferability of the adversarial examples across different models, as described in section 4.

4 Experimental results

All our experiments are run in a CPU: Xeon Gold 6139, RAM: 96G, GPU: Tesla V100 16G environment. In this section, the whole experimental results are presented concerning the following aspects: First, we use Inception-v3 to test the classification accuracy of the original and augmented data from CIFAR-10, ImageNet and LSUN. Second, by optimizing Eq. (11), we ascertain the relevant weights and some hyperparameters in our experiment, such as the learning rate, epochs, optimizer, etc. Next, we demonstrate the visualization effect of our approach on high-resolution images, the semantic correlation between the generated images and the original images, and the similarity between images by some evaluation metrics. Then, we compare our approach with AdvGAN with respect to attack success rate, and further compare it with some sensitivity-based and optimization-based attack strategies with respect to attack success rate and time consumption for generating a single adversarial example. Finally, we measure the attack performance of adversarial examples generated using our approach to test their transferability.

4.1 Dataset accuracy

At the beginning of the experiment, we used Inception-v3 to perform the accuracy test on the original and augmented data from CIFAR-10,ImageNet and LSUN, as shown in Table 1. We calculated top-1 accuracy and top-5 accuracy on ImageNet. Only top-1 accuracy is computed on CIFAR-10 and LSUN due to the relatively small number of classes included in these two datasets (10 classes on CIFAR-10 and 20 object categories on LSUN). It also prevents the corresponding values in Eq. (13) are considerable when the values in Eq. (12) are small. For example, suppose there is a dataset that is only 1% in Eq. (12), which means that even if nothing is manipulated in the experiment, its attack success rate in Eq. (13) is close to 99%, thereby defeating the meaning of generating adversarial examples. Although it is a simple step in the experiment, it is missing in quite a few previous papers on the subject, and they all put too much faith in the performance of the dataset itself and the trained model.

We have defined an accuracy rate in our experiment as follows:

$$Accuracy\ rate = \frac{\sum_{i=1}^{N} \mathbb{I}\left(f_\theta(x_i) = y_i^{true}\right)}{N} \tag{12}$$

where N is the number of test images, $\mathbb{I}()$ is one if condition is true, otherwise zero.

4.2 Corresponding parameters

Our approach identifies some parameters in Table 2 by the loss variation (Fig. 4.) of the dataset in the target model during training. For example, after identifying the optimizer as Adam, we run the CIFAR-10 train set through Algorithm 1 and divide the transformed image into eight subfiles (including a mixed set of images transformed by Algorithm 1, called 'Data augmentation') to determine the epoch. These subfiles as shown in Fig. 4a are used to train the target model. The curve decreases smoothly in the early stages of training, indicating that the model will accelerate learning in the early

Table 1 Accuracy rate on CIFAR-10, ImageNet and LSUN

Dataset		Original	Hue	Saturation	Brightness	Mirror	Rotation	Padding
CIFAR-10	Training set	**99.59%**	99.54%	99.57%	99.55%	99.48%	99.58%	99.49%
	Test set	**98.15%**	95.28%	97.16%	97.68%	82.83%	88.38%	88.76%
ImageNet	Top-1	83.59%	82.69%	84.15%	83.31%	**86.46%**	81.23%	82.92%
	Top-5	96.05%	94.86%	94.85%	96.00%	**97.31%**	94.85%	96.07%
LSUN	Training set	73.91%	67.22%	71.91%	73.24%	**79.93%**	71.81%	73.06%
	Test set	72.24%	66.89%	69.90%	70.91%	**75.92%**	70.47%	71.81%

Table 2 Some relevant parameters on CIFAR-10,ImageNet and LSUN during the training process

	CIFAR-10	ImageNet	LSUN
Optimizer	Adam	Adam	Adam
Learning rate	0.001	0.001	0.001
Batch size	300	128	64
Epoch	60	50	40
Learning rate decay schedule	30	35	30
Weight α	0.1	0.1	0.25
Weight β	0.05	0.02	0.05

stages, making the model already very close to the local or global optimal solution. However, there is a large oscillation before epoch = 30, when the value of the loss function hovers around the minimum value, and it is always difficult to reach the optimum. Therefore, as shown in Table 2, we perform learning rate decay during the training of the CIFAR-10 when epoch = 30, changing from the previous 0.001 to 0.0001. A similar training process can be seen in Fig. 4b and Fig. 4c.

Figure 4d, Fig. 4e and Fig. 4f show the losses of the objective function Eq. (6), (7), (8), (9), and (11) on CIFAR-10,ImageNet and LSUN, respectively, and it can be found that GAN reaches convergence at the later stages of training. The other parameters in Table 2 can also be determined through similar experiments.

4.3 Visualization of the generated adversarial images

In visualizing the adversarial examples, our experiments are carried out by (i) mapping random Gaussian noise z into the data space and using AdvGAN to generate adversarial perturbations(Z-perturbations); (ii) using PCA to reduce the dimensionality of the input and mapping it to latent variables to generate adversarial perturbations (PCA-perturbations); and (iii) using KPCA to reduce the dimensionality of the input and mapping it to latent variables to generate adversarial perturbations (KPCA-perturbations). Figure 5 shows the visualization results on CIFAR-10. In Fig. 5a are the randomly selected original images. In Fig. 5b, the first row corresponds to the adversarial images generated using random Gaussian noise z mapped to the data space, i.e., AdvGAN; the second row shows the pixel difference between them. In Fig. 5c, the first row presents the adversarial images generated by mapping the input through PCA to latent variables; the second row shows the pixel difference. In Fig. 5d, the first presents the adversarial images generated by mapping the input

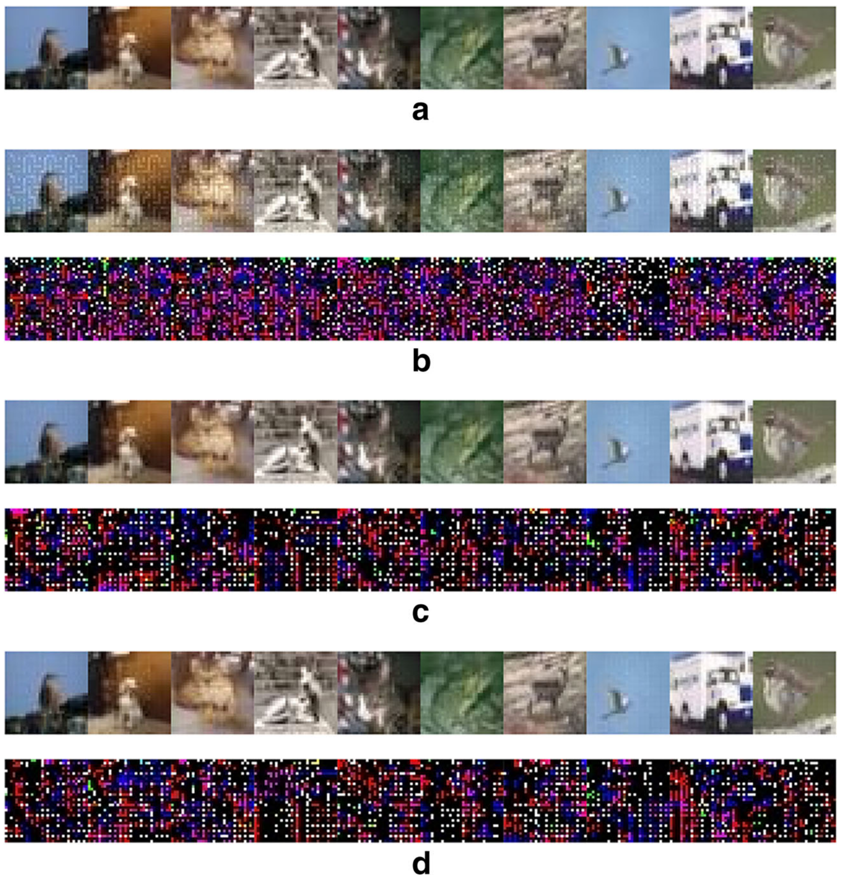

Fig. 5 Visualizations on CIFAR-10. **a** Original images on CIFAR-10, **b** Z-perturbations adversarial images, **c** PCA-perturbations adversarial images, **d** KPCA-perturbations adversarial images

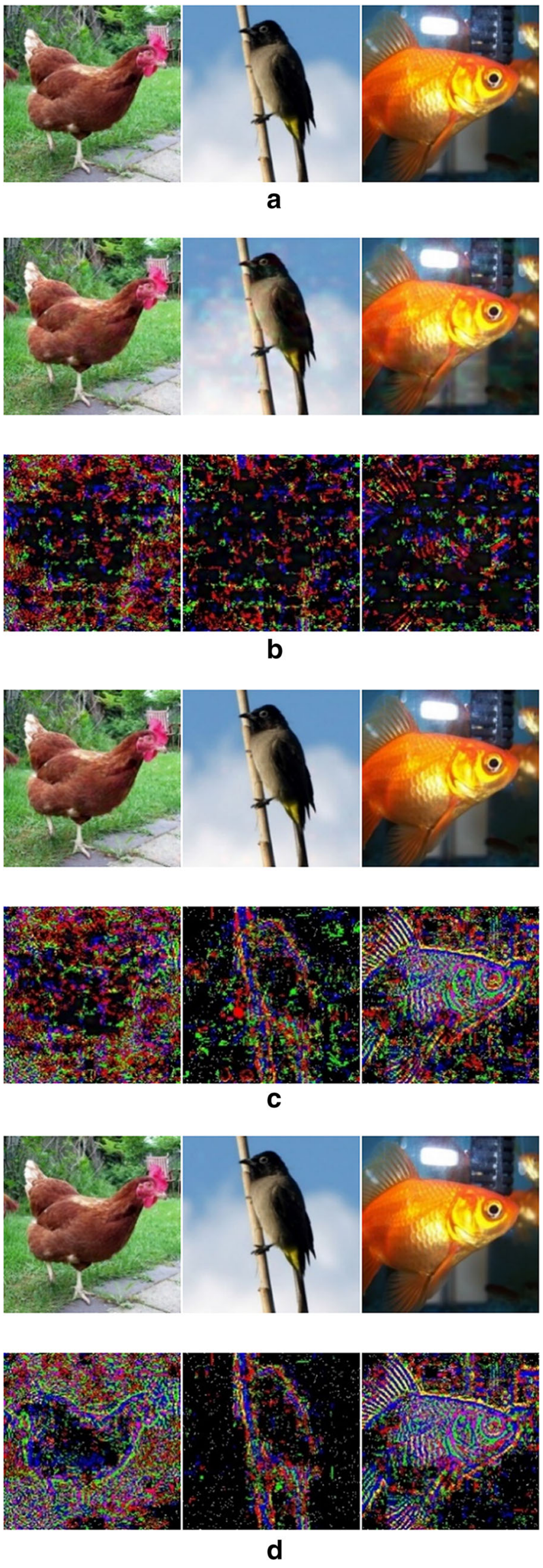

Fig. 6 Visualizations on ImageNet. **a** Original images on ImageNet, **b** Z-perturbations adversarial images, **c** PCA-perturbations adversarial images, (**d**) KPCA-perturbations adversarial images

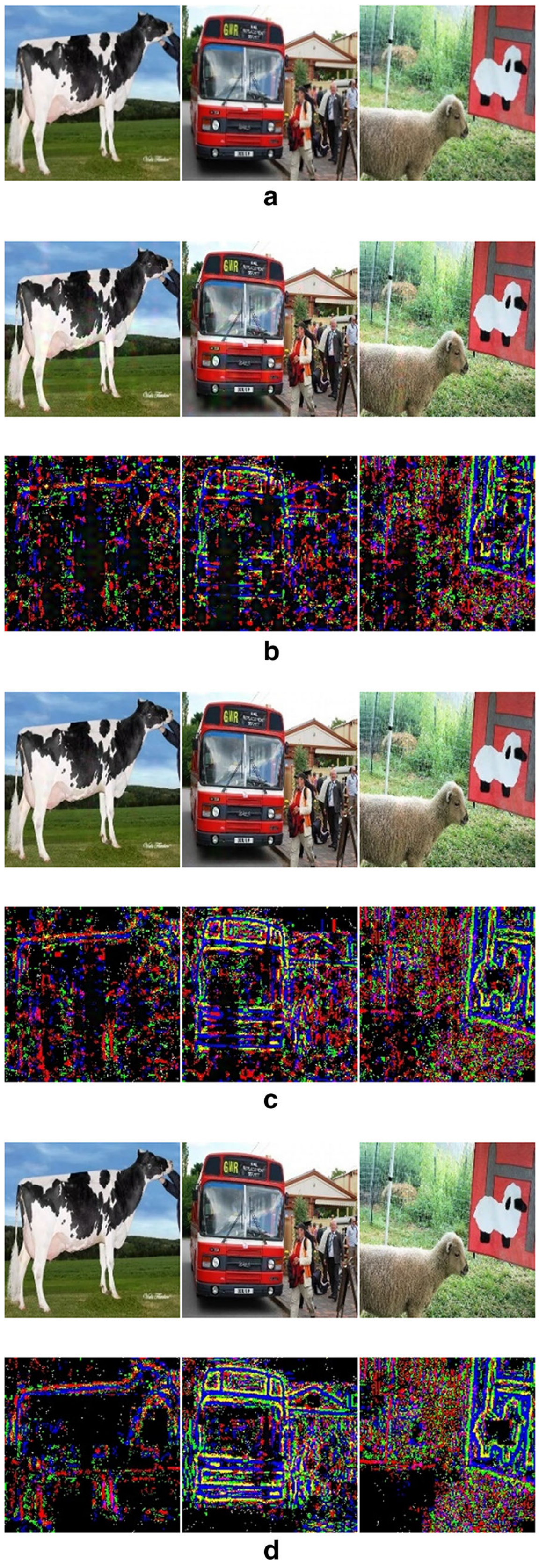

Fig. 7 Visualizations on LSUN. **a** Original images on LSUN, **b** Z-perturbations adversarial images, **c** PCA-perturbations adversarial images, **d** KPCA-perturbations adversarial images

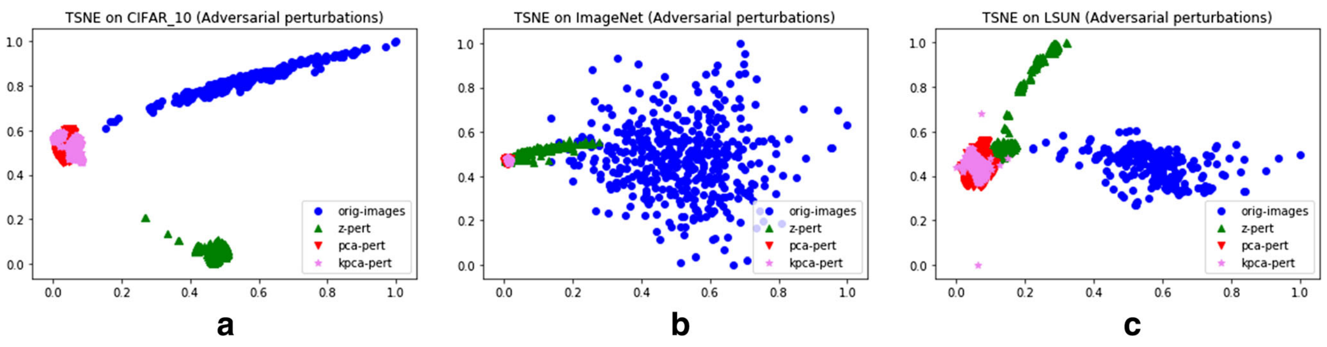

Fig. 8 The adversarial perturbations generated by the different methods were processed and visualized by TSNE

through KPCA to latent variables; the second shows the pixel difference.

The adversarial images in Fig. 5b have visible perturbations that can be detected by the naked eye and are less sharp than the images in Fig. 5c and Fig. 5d. Furthermore, the pixel difference between the generated adversarial images and the original images in Fig. 5b is more severe and complex than those in Fig. 5c and Fig. 5d, which indicates that our approach is superior to AdvGAN in terms of visualization effect. It is also suggested that our approach is more restrictive on the perturbation magnitude through Eq. (9) and (10).

Next, we would like to concentrate more on the visualization on high-resolution datasets, as shown in Fig. 6. Due to the large size of the images on ImageNet, the adversarial examples generated in Fig. 6b, Fig. 6c, and Fig. 6d are difficult to distinguish by visual inspection. However, by looking at the pixel differences in Fig. 6b, they look like randomly added noise and appear cluttered. The pixel differences in Fig. 6b

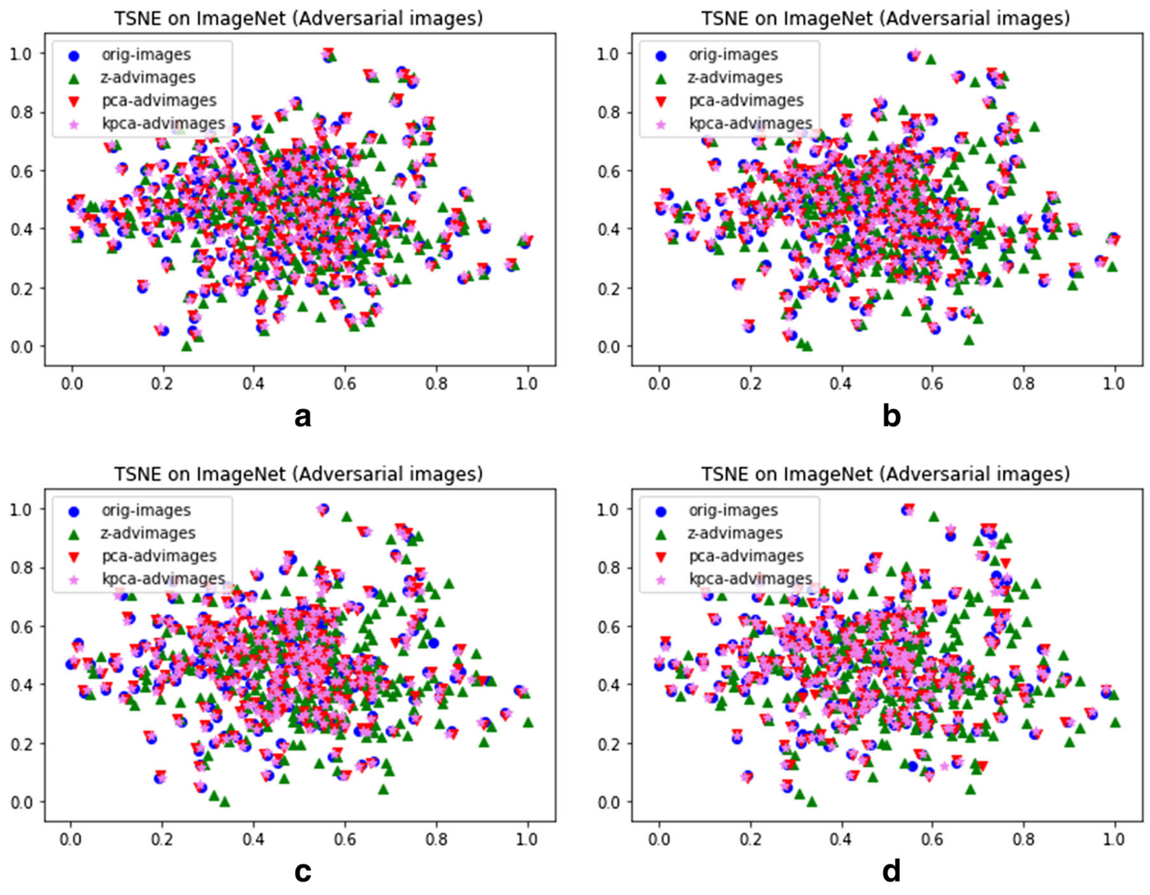

Fig. 9 The similarity of the adversarial image generated by different methods is compared with the original image, where the closer to the original images (blue dots) represents a higher degree of similarity. The AdvGAN generated images (green triangles) are farther away from the original image than our proposed approach, and this phenomenon is more obvious as the image size increases. **a** Image size:32*32*3, **b** Image size:64*64*3, **c** Image size:128*128*3, **d** Image size:224*224*3

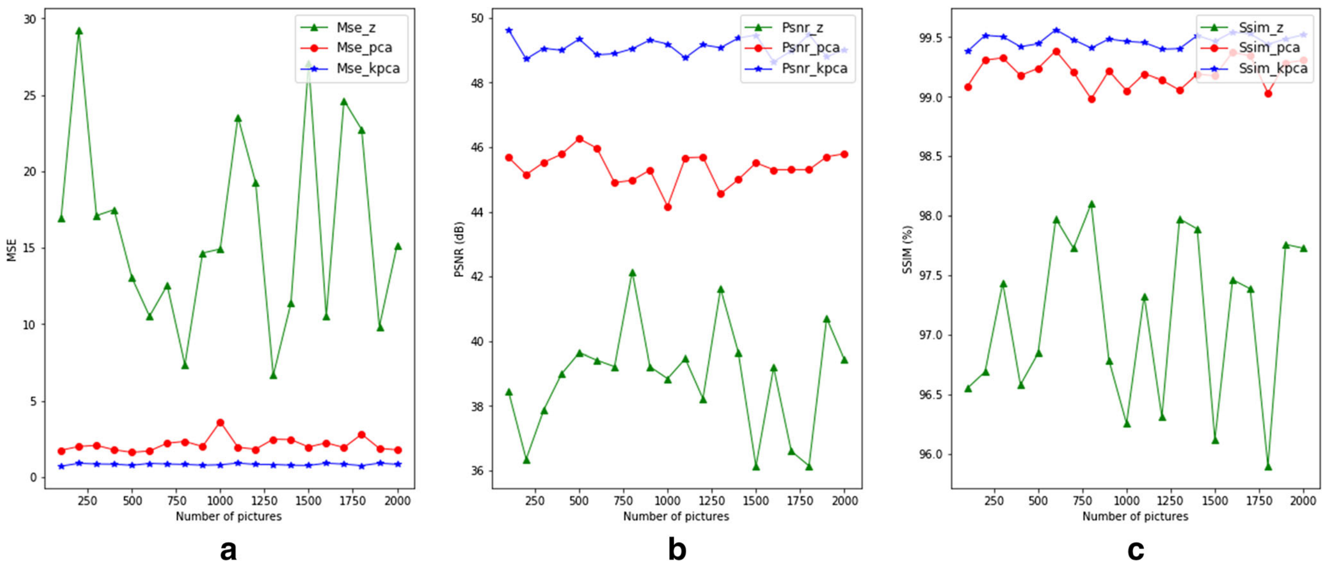

Fig. 10 Some image quality evaluation metrics on ImageNet, (**a**) is used to measure the mean square value of pixels between images, smaller value is better. (**b**) is used to measure the distortion or noise level between images, in dB. Larger value is better. (**c**) is used to measure the structural similarity between images, closer the value to 1 the more similar

and Fig. 6c, in contrast, are semantically more like input instances, which makes the generated adversarial examples look natural. In turn, it argues for the validity of mapping input instances to latent variables via PCA and KPCA, thereby generating strongly semantically related adversarial examples. Likewise, similar visualization effects as described above are also available on LSUN (in Fig. 7).

To further explore the better semantics of the adversarial examples generated by our approach, we randomly select 500 original images from ImageNet; corresponding to the perturbations generated by random Gaussian noise as input; the perturbations generated by mapping PCA to latent variables as input; and the perturbations generated by mapping KPCA to latent variables as input; each with 500 images. We reduced these images into two dimensions and visualized them by TSNE [34], as shown in Fig. 8b. The PCA perturbations and KPCA perturbations are closer after TSNE processing, which explains the pixel differences in Fig. 6c and Fig. 6d that are highly semantically correlated with the original images. Equally, Fig. 8a and Fig. 8c show similar conclusions on CIFAR-10 and LSUN, respectively.

4.4 Similarity between images

In this subsection, we demonstrate that the adversarial examples generated by our two methods are closer to the original images than AdvGAN. We randomly select 250 images from ImageNet and use three methods to generate the corresponding adversarial images total 1000 images. To mitigate the effect of different sizes on the similarity between images, we set the image size to 32*32*3,64*64*3128*128*3224*224*3 in turn. These images are also reduced to two dimensions by TSNE [34] and visualized. In Fig. 9, the larger the size, the greater difference in similarity. Even though the images are at different sizes, the similarity to the original images is more pronounced in our approach than in AdvGAN. Specifically, the similarity between the adversarial images generated by PCA and KPCA mapping (represented by red triangles and pink pentagons) and the original image (blue dots) is much closer. The AdvGAN-generated adversarial images (green triangles), on the other hand, is more distant from the original image as the size of the image increases. This indicates that the adversarial images generated by AdvGAN do not perform as well as our proposed approach in similarity.

Table 3 Attack success rate on CIFAR-10

CIFAR-10	Epoch_20	Epoch_40	Epoch_60
Z-advimages	89.95%	88.93%	88.69%
PCA-advimages	**90.08%**	90.15%	**90.30%**
KPCA-advimages	90.05%	**90.32%**	90.24%

Table 4 Attack success rate on ImageNet

ImageNet	Epoch_20	Epoch_40	Epoch_50
Z-advimages	76.89%	79.27%	78.94%
PCA-advimages	**80.19%**	**80.28%**	**80.18%**
KPCA-advimages	79.89%	80.02%	79.97%

Table 5 Attack success rate on LSUN

LSUN	Epoch_20	Epoch_30	Epoch_40
Z-advimages	64.03%	65.39%	66.01%
PCA-advimages	66.28%	66.55%	**67.53%**
KPCA-advimages	**66.76%**	**66.74%**	66.71%

Then, we refer to some image quality evaluation metrics to verify that the images generated by the proposed approach are more similar to the original images [37]. In this paper, in order to evaluate the image quality and avoid the interference of single metrics on the image quality, we use Mean Square Error (MSE) [38], Peak Signal to Noise Ratio (PSNR) [39], and Structural Similarity (SSIM) [39] in Fig. 10 to compare similarity rigorously and objectively. We select 2000 adversarial images generated by each of the three methods from ImageNet, divide them into 20 batches, and calculate the average value of each metric for each batch. In Fig. 10a, MSE first calculates the mean square value of the pixel difference between the generated images and the original images, and then determines the generated distortion by the magnitude of the mean square value. In Fig. 10b, PSNR is the ratio of the maximum pixel value of the image to the noise intensity, which is an objective measure of the distortion or noise level of the image. It is expressed as the greater the value between images, the more similar they are. In Fig. 10c, we use the SSIM to measure the image similarity in terms of brightness, contrast, and structure, which takes values in the range of [0,1]. The value closer to 1 means the similarity between images is stronger. Therefore, it is clear from Fig. 10 that the images generated by the proposed approach perform better in distortion, similarity, and correlation. It also indicates that our method generates images with better visualization than AdvGAN.

4.5 Attack success rate

In this part, we will discuss two aspects: (i) the comparison between the adversarial examples generated by our approach and AdvGAN in terms of attack success rate; and (ii) the comparison of our approach with sensitivity analysis-based, optimization-based attack method in terms of attack success rate and time consumed for single image generation.

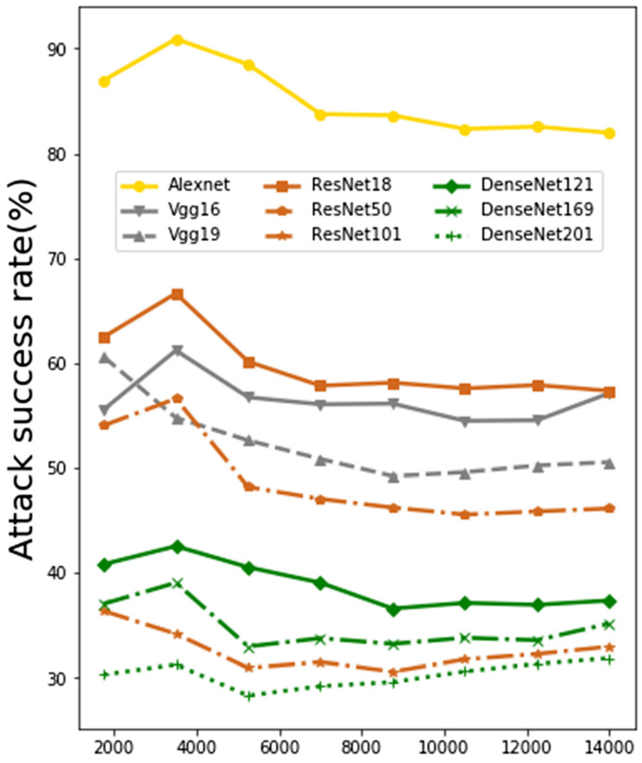

Fig. 11 Transferability of adversarial examples on other models

First, we have defined the attack success rate in our experiment as follows:

$$Attack\ success\ rate = \frac{\sum_{i=1}^{N}\mathbb{I}\left(f_{\theta}(x_i) = y_i^{pred} \neq y_i^{true}\right)}{N} \quad (13)$$

It is an essential measure of the performance of adversarial examples. In the experiment, we use generator G trained from different epochs to generate the corresponding perturbations. The perturbations are spliced with the original images to form adversarial examples. Table 3 presents the attack success rate for the adversarial examples on the CIFAR-10 test set. Our approach improves the attack success rate by approximately 1.5% compared to AdvGAN; Table 4 shows that attack success rate on ImageNet improves by approximately 3.3%. Table 5 shows that attack success rate on LSUN increases by about 2.4%.

Table 6 Performance of various attack algorithms

	Sensitivity Analysis-based		Optimization-based		Generative-based		
Attack algorithms	FGSM	PGD	DeepFool	C&W	AdvGAN	Ours (PCA)	Ours (KPCA)
Attack success rate	61.51%	95.83%	66.89%	**97.54%**	78.94%	80.18%	79.97%
Time consumption	1.16 s	7.09 s	4.63 s	76.68 s	1.15 s	0.95 s	**0.88 s**

Next, we compare our approach with some other attack algorithms. The test images are taken from test set on ImageNet, and the experimental results are all obtained by running on a Tesla V100 GPU. In this compare experiment, all perturbations are applied to the pixel values of images, which normally take values in the range [0,255]. So the maximum perturbation ϵ with respect to the data's dynamic range would be $\lambda/256$, where λ is perturbation strengths. For the sensitivity analysis-based methods, we choose FGSM [9] and PGD [11], where the adversarial examples generated by FGSM with $\epsilon = 8/255$,and for PGD with the number of attack iterations n=40 and attack step size α=1. For the optimization-based methods, C&W [13] and DeepFool [12]. The adversarial images by DeepFool with maximum *epochs*=50,and for C&W: maximum iterations are 1000; learning rate lr=0.01 and initial constant c = 0.0001. The primary algorithm implementation is referenced to Advbox which is a toolkit for generating adversarial examples against the deep learning model. In this experiment, the Alexnet model is pretrained under PyTorch when generating adversarial examples to influence the dropout and BN layer behavior. Each method is computed the required time to generate a single adversarial image.

For being able to evaluate the effectiveness of different attack strategies in a rigorous and fair way, the adversarial images we use for testing satisfy four requirements: (1) the structural similarity between the adversarial image and the original image, i.e., SSIM [39], should be maintained above 95%; (2) the PSNR [39] between the adversarial image and the original image should be over 35 dB (less image distortion); (3) all these attack strategies were bounded attacks according to a predefined maximum perturbation $\epsilon \leq 8/255$ with respect to the L_∞ norm; (4) all adversarial images were generated in a white-box setting and in an untargeted attack way. And the attack success rate is calculated using ResNet50 in the testing phase. As shown in Table 6, our approach has a higher attack success rate than the sensitivity analysis-based algorithm and a lower attack success rate than the optimization-based algorithm. Nevertheless, our time overhead is much less than that of the optimization-based method and better than that of AdvGAN.

4.6 Transferability of adversarial examples

At the end of the experiment, the transferability of the adversarial examples generated by our approach is tested, and we calculate the attack success rate of a total of 14,000 adversarial examples on each of the main deep learning models. Figure 11 shows the trend of transferability during testing. The adversarial examples achieve a higher success rate on simpler models. These findings also lay the foundation for our approach in future black-box attacks.

5 Conclusion

In this paper, we propose an approach for generating high-resolution and more naturalistic adversarial examples from the perspective of exploring the blind spots that exist in DNN. The technique incorporates generative adversarial nets, where the data are augmented to avoid overfitting. We innovatively combine the typical linear dimension reduction method of PCA and the nonlinear dimension reduction method of KPCA for the first time to map the input instances to the latent variables required for the generator training process. Experimental results show that our approach generates adversarial examples with an improved attack success rate and better semantics compared to AdvGAN. We fill the gap of generating high-resolution adversarial examples via GAN on large-scale datasets by demonstrating the feasibility of the proposed approach on CIFAR-10 and by applying the above approach to ImageNet and LSUN. Furthermore, the visualization of the pixel differences between the original images and the adversarial images; the scatter plots obtained by TSNE processing; and some image quality evaluation metrics to further illustrate that the adversarial examples generated by our approach are more natural and have strong semantically relevance.

Next, the comparison of our approach with the sensitivity analysis-based and optimization-based methods in terms of attack success rate and time consumption for generating a single image is presented. Both demonstrate the good performance of our approach and to provide options for future researchers in choosing how to generate adversarial examples.

Finally, given that this work is based on semiwhite-box attacks, we study the transferability of the generated adversarial examples across some major deep learning models, and lay the groundwork for applying the method of using GAN to generate high-resolution adversarial examples on large-scale datasets to black-box attacks in the future.

Acknowledgments This work was supported in part by the National Natural Science Foundation of China under Grant 61572034, the Major Science and Technology Projects in Anhui Province under Grant 18030901025, the Anhui Province University Natural Science Fund under Grant KJ2019A0109, the Natural Science Foundation of Anhui Province of China under Grant 2008085MF220, the Science and Technology Project of Wuhu City in 2020 under Grant No.2020yf48, and the School Foundation of Anhui University of Science and Technology under Grant No.2020CX2077.

References

1. Zhang K, Gool LV, Timofte R (2020) Deep unfolding network for image super-resolution. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 3217–3226. https://doi.org/10.1109/CVPR42600.2020.00328

2. McKinney SM, Sieniek M, Godbole V, Godwin J, Antropova N, Ashrafian H, Back T, Chesus M, Corrado GS, Darzi A, Etemadi M, Garcia-Vicente F, Gilbert FJ, Halling-Brown M, Hassabis D, Jansen S, Karthikesalingam A, Kelly CJ, King D, Ledsam JR, Melnick D, Mostofi H, Peng L, Reicher JJ, Romera-Paredes B, Sidebottom R, Suleyman M, Tse D, Young KC, De Fauw J, Shetty S (2020) International evaluation of an AI system for breast cancer screening. Nature 577(7788):89–94. https://doi.org/10.1038/s41586-019-1799-6
3. Huang H, Xue F, Wang H, Wang Y (2020) Deep graph random process for relational-thinking-based speech recognition. In: International Conference on Machine Learning, PMLR, 119: 4531–4541
4. Gao L, Li X, Song J, Shen HT (2020) Hierarchical LSTMs with adaptive attention for visual captioning. IEEE Trans Pattern Anal Mach Intell 42(5):1112–1131. https://doi.org/10.1109/tpami.2019.2894139
5. Delcoucq L, Lecron F, Fortemps P, Aalst WMPvd (2020) Resource-centric process mining: clustering using local process models. Paper presented at the Proceedings of the 35th Annual ACM Symposium on Applied Computing. Brno, Czech Republic, pp 45–52. https://doi.org/10.1145/3341105.3373864
6. Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow IJ, Fergus R (2014) Intriguing properties of neural networks.
7. Nguyen A, Yosinski J, Clune J (2015) Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 427–436. https://doi.org/10.1109/CVPR.2015.7298640
8. Serban A, Poll E, Visser J (2020) Adversarial examples on object recognition: a comprehensive survey. 53 (ACM Comput. Surv.): article 66. https://doi.org/10.1145/3398394
9. Goodfellow IJ, Shlens J, Szegedy CJapa (2014) Explaining and harnessing adversarial examples
10. Papernot N, McDaniel P, Jha S, Fredrikson M, Celik ZB, Swami A (2016) The limitations of deep learning in adversarial settings. In: 2016 IEEE European symposium on security and privacy (EuroSamp;P), IEEE, pp 372–387. https://doi.org/10.1109/EuroSP.2016.36
11. Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A (2018) Towards deep learning models resistant to adversarial attacks.
12. Moosavi-Dezfooli S-M, Fawzi A, Frossard P (2016) Deepfool: a simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2574–2582. https://doi.org/10.1109/CVPR.2016.282
13. Carlini N, Wagner D (2017) Towards evaluating the robustness of neural networks. In: 2017 ieee symposium on security and privacy (sp), IEEE, pp 39–57. https://doi.org/10.1109/SP.2017.49
14. Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville AC, Bengio Y (2014) Generative adversarial nets.
15. Xiao C, Li B, Zhu J-Y, He W, Liu M, Song D (2018) Generating adversarial examples with adversarial networks. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence, pp 3905–3911. https://doi.org/10.24963/ijcai.2018/543
16. Zhao Z, Dua D, Singh S (2018) Generating natural adversarial examples
17. Liu X, Hsieh C-J (2019) Rob-gan: generator, discriminator, and adversarial attacker. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 11234–11243. https://doi.org/10.1109/CVPR.2019.01149
18. Mikołajczyk A, Grochowski M (2018) Data augmentation for improving deep learning in image classification problem. In: 2018 international interdisciplinary PhD workshop (IIPhDW), IEEE, pp 117–122. https://doi.org/10.1109/IIPHDW.2018.8388338
19. Shorten C, TMJJoBD K (2019) A survey on image data augmentation for deep learning. J Big Data 6(1):60
20. Yun S, Han D, Oh SJ, Chun S, Choe J, Yoo Y (2019) Cutmix: regularization strategy to train strong classifiers with localizable features. In: Proceedings of the IEEE International Conference on Computer Vision, pp 6023–6032. https://doi.org/10.1109/ICCV.2019.00612
21. Zhong Z, Zheng L, Kang G, Li S, Yang Y (2020) Random erasing data augmentation. In: AAAI, pp 13001–13008. https://doi.org/10.1609/aaai.v34i07.7000
22. Zhong Z, Zheng L, Zheng ZD, Li SZ, Yang Y (2019) CamStyle: A novel data augmentation method for person re-identification. IEEE Trans Image Process 28 (3):1176–1190. https://doi.org/10.1109/tip.2018.2874313
23. Bang D, Shim H (2018) Improved training of generative adversarial networks using representative features. In: International Conference on Machine Learning, PMLR 80:433–442
24. Tang H, Xu D, Yan Y, Torr PH, Sebe N (2020) Local class-specific and global image-level generative adversarial networks for semantic-guided scene generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 7870–7879. https://doi.org/10.1109/CVPR42600.2020.00789
25. Zhu J-Y, Park T, Isola P, Efros AA (2017) Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of the IEEE international conference on computer vision, pp 2223–2232. https://doi.org/10.1109/ICCV.2017.244
26. Miyato T, Kataoka T, Koyama M, Yoshida Y (2018) Spectral normalization for Generative Adversarial Networks
27. Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville AC (2017) Improved training of wasserstein gans. In: Advances in Neural Information Processing Systems, pp 5767–5777
28. Tolstikhin IO, Gelly S, Bousquet O, Simon-Gabriel C-J, Schölkopf B (2017) Adagan: boosting generative models. In: Advances in Neural Information Processing Systems, pp 5424–5433
29. Huang X, Li Y, Poursaeed O, Hopcroft J, Belongie S (2017) Stacked generative adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5077–5086. https://doi.org/10.1109/CVPR.2017.202
30. Yu L, Zhang W, Wang J, Yu Y (2017) Seqgan: sequence generative adversarial nets with policy gradient. In: Thirty-first AAAI conference on artificial intelligence, pp 2852–2858
31. Fedus W, Goodfellow I, Dai AM (2018) MaskGAN: better text generation via filling in the_. In: International Conference on Learning Representations
32. Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks. In: Advances in neural information processing systems, pp 3104–311
33. Jia X, Wei X, Cao X, Foroosh H (2019) Comdefend: an efficient image compression model to defend adversarial examples. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 6084–6092. https://doi.org/10.1109/CVPR.2019.00624
34. van der Maaten L, Hinton G (2008) Visualizing data using t-SNE. J Mach Learn Res 9:2579–2605
35. . Lin Z, Sun J, Davis A, Snavely N (2020) Visual chirality. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 12295–12303. https://doi.org/10.1109/CVPR42600.2020.01231
36. Xie C, Zhang Z, Zhou Y, Bai S, Wang J, Ren Z, Yuille AL (2019) Improving transferability of adversarial examples with input diversity. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 2730–2739. https://doi.org/10.1109/CVPR.2019.00284

37. Dong Y, Fu Q-A, Yang X, Pang T, Su H, Xiao Z, Zhu J (2020) Benchmarking adversarial robustness on image classification. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 318–328. https://doi.org/10.1109/CVPR42600.2020.00040
38. Yuan J, He Z (2020) Ensemble generative cleaning with feedback loops for defending adversarial attacks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 581–590. https://doi.org/10.1109/CVPR42600.2020.00066
39. Zhu P, Abdal R, Qin Y, Wonka P (2020) Sean: Image synthesis with semantic region-adaptive normalization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 5104–5113. https://doi.org/10.1109/CVPR42600.2020.00515
40. Hayashi T, Fujita H, Hernandez-Matamoros A (2021) Less complexity one-class classification approach using construction error of convolutional image transformation network. Inf Sci 560:217–234. https://doi.org/10.1016/j.ins.2021.01.069
41. Zhou F, Yang S, Fujita H, Chen D, Wen C (2020) Deep learning fault diagnosis method based on global optimization GAN for unbalanced data Knowledge-Based Systems 187. https://doi.org/10.1016/j.knosys.2019.07.008

Xianjin Fang received the Ph.D. degree in computer application technology from Anhui University in 2010. He is a professor and Ph.D. supervisor at Anhui University of Science and Technology. His research interests include information security, data mining, etc.

Zhiwei Li received his B.S. degree in 2017 in Shanghai polytechnic university. He is currently pursuing the master degree at Anhui University of Science and Technology. His research interests include AI security, privacy preserving, etc.

Gaoming Yang received the master's degree in computer application from Guizhou University in 2003; the Ph.D. degree in computer application technology from Harbin Engineering University in 2012. He is an associate professor and M.S. supervisor at Anhui university of Science and Technology. He used to be a visiting scholar at the University of Arkansas for one year. His research interests include machine learning, privacy preserving, etc.