Pattern Recognition 144 (2023) 109849

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

# Few pixels attacks with generative model

Yang Li [a,*], Quan Pan [a], Zhaowen Feng [a], Erik Cambria [b]

[a] Northwestern Polytechnical University, China
[b] Nanyang Technological University, Singapore

ARTICLE INFO

*Keywords:*
Neural network vulnerability
Adversarial attack
Few pixels attacks
Generative attack

ABSTRACT

Adversarial attacks have attracted much attention in recent years, and a number of works have demonstrated the effectiveness of attacks on the entire image at perturbation generation. However, in practice, specially designed perturbation of the entire image is impractical. Some work has crafted adversarial samples with a few scrambled pixels by advanced search, e.g., SparseFool, OnePixel, etc., but they take more time to find such pixels that can be perturbed. Therefore, to construct the adversarial samples with few pixels perturbed in an end-to-end way, we propose a new framework, in which a dual-decoder VAE for perturbations finding is designed. To make adversarial learning more effective, we proposed a new version of the adversarial loss by considering the generalization. To evaluate the sophistication of the proposed framework, we compared it with more than a dozen existing related attack methods. The effectiveness and efficiency of the proposed framework are verified from the extensive experimental results. The validity of the model structure is also validated by the ablation study.

## 1. Introduction

Neural Networks have demonstrated great success in many fields, e.g., natural language processing [1], image analysis [2], speech recognition [3], recommender system [4], etc. However, recent studies have indicated that neural networks are vulnerable to adversarial attacks [5]. The adversarial attack usually is deployed with carefully crafted perturbations and added to the image which misleads the model in the meantime to be imperceptible to human eyes.

Different methods exist to construct perturbations, according to the stage at which the adversarial attack is performed, there are two types of attacks: poisoning attacks [6] and evasion attacks [7]. The former takes place during the training phase and creates backdoors in the machine learning model by adding tainted examples to the training set. The latter happens at the test stage, by adding deliberately designed tiny perturbations to benign test samples to mislead the neural network. There are two main categories of attacks based on the attacker's level of knowledge about the target model: white-box attacks and black-box attacks. White-box attacks assume that the attacker has complete knowledge of the model, including its architecture and training data, while black-box attacks assume that the attacker has no knowledge of the model beyond its input–output behavior. There are also two main categories of attacks based on the outcome of the attack: targeted attacks and non-targeted (indiscriminate) attacks. Targeted attacks aim to cause a specific, desired outcome for the attacker, such as misclassifying a particular input as a certain class. Non-targeted attacks, on the other hand, do not aim to achieve any specific outcome, but rather seek to degrade the overall performance of the model. The various combinations of these attack types can result in a wide range of attack scenarios

Adversarial examples are typically generated using a gradient-based method in the global space, with the magnitude of perturbation being equal to the size of the image. For example, methods like Fast Gradient Sign Method (FGSM) [8], Project Gradient Descent (PGD) [9], etc. are optimized on target loss directly across the entire image. There are also models that construct the perturbation over a few pixels, for example, Modas et al. [10] introduced SparseFool, a method that uses a linear approximation to perturb images in order to execute a few successful pixel attacks. However, the number of disturbing pixels in this approach is not practical to manage. Later, Su et al. [11] proposed the OnePixel model, which uses differential evolution to find local optima, but this process takes a significant amount of time. Therefore, can we efficiently identify adversarial points with a controlled number of perturbed pixels, even when the model's architecture and parameters are unknown? By limiting the modifications to a small number of pixels, the adversarial perturbations become less noticeable to human observers, making them highly effective in real-world scenarios. This

* Corresponding author.
*E-mail addresses:* liyangnpu@nwpu.edu.cn (Y. Li), quanpan@nwpu.edu.cn (Q. Pan), fengzhaowen@mail.nwpu.edu.cn (Z. Feng), cambria@ntu.edu.sg (E. Cambria).

https://doi.org/10.1016/j.patcog.2023.109849
Received 20 September 2022; Received in revised form 22 May 2023; Accepted 26 July 2023
Available online 29 July 2023

stealthy nature enables adversaries to launch attacks that can bypass security systems, such as image classifiers or object recognition algorithms, while remaining visually indistinguishable from the original image. Additionally, minimizing the number of modified pixels reduces computational complexity and generation time for adversarial examples, facilitating faster and more efficient attack strategies. Furthermore, it is important to note that in real-world applications, we typically lack detailed information about the victim model. Hence, this question emphasizes the urgency of swiftly identifying these critical points while maintaining control over the manipulated pixels.

To cope with this problem, we try to generate perturbations in an end-to-end manner using a neural network-based model, where a shadow model is used as a substitute to guide the generation of perturbations. In general, the difference between the adversarial sample and the corresponding original sample should be as small as possible, which is the stealthiness usually guaranteed by adversarial attack methods. We believe that the point of attack will be more stealthy when it occurs on the contour of the objective. For example, all the attack points that are in the outline of the image would be more difficult to identify. Therefore, to guide the attack point to be located in the silhouette, we apply a generative model (e.g., generative adversarial network (GAN) [12], variational auto-encoder (VAE) [13]) to generate perturbations. Also, to make the attack effective, inspired by the work of CW [14], adversarial samples should be misclassified with a high probability of misclassification. The problem is that it is difficult to optimize the target category to the adversarial one as it only concerns the maximum and the target one. To cope with this problem, a newly designed adversarial loss is proposed to let the model craft the adversarial samples more smoothly. Another significant challenge is to determine the precise location of a limited number of pixels that can be manipulated. To deal with this problem, in this work, a dual-decoder VAE is used, where the output of one decoder is used to obtain image salient silhouettes, and the output of the other decoder is used to generate perturbations. Then the adversarial sample is constructed by choosing places in the silhouette that have a larger value than the top-$K$. In summary, the contributions of this paper are as below:

- We design a new framework for adversarial attacks by introducing a shadow attack approach.
- We design an adversarial sample generation model that is able to control the number of pixels attacked.
- The empirical results show the effectiveness of the model in a black-box setting.

## 2. Related works

Since Szegedy et al. [8] first proposed the existence of adversarial examples, many adversarial attack algorithms have been proposed to discuss the vulnerability of deep neural networks. Existing adversarial attack models can be roughly divided into white-box attacks and black-box attacks based on whether knowing the target model.

### 2.1. White-box attacks

In recent years, there has been growing concern about the vulnerability of neural networks to adversarial attacks, where small perturbations to images can mislead the network. Szegedy et al. [8] first demonstrated this vulnerability with the L-BFGS attack, which highlights the nonlinear representation and over-fitting of neural networks. In response, Goodfellow et al. [15] then introduced the FGSM, which generates adversarial perturbations through a one-step computation on the non-targeted loss function. Papernot et al. [16] also developed the Jacobian-based Saliency Map Attack (JSMA), which relies on knowledge of the network to create adversarial samples using saliency maps derived from forwarding derivations. There have been several variations on FGSM proposed, such as R-FGSM [17], which adds random perturbations for enhanced attacks, MI-FGSM [18], which integrates momentum terms into iterations to eliminate bad local maxima, and F-FGSM [19], which combines FGSM with random initialization and has been shown to achieve the same effect as PGD [9]. Xie et al. [20] also proposed the DI-FGSM attack, which adjusts the input image randomly in each iteration before generating adversarial perturbations with FGSM.

Kurakin et al. [21] proposed iterative methods, such as the basic iterative method (BIM) and the iterative least-likely class method (ILLC), for generating adversarial examples. These methods differ from single-step attacks in that they involve multiple iterations to generate adversarial examples. The PGD attack, introduced by Madry et al. in their work [9], is essentially a variant of the Fast Gradient Sign Method (FGSM) that involves taking multiple small-step iterations instead of a single large step. In addition, the PGD attack clips the perturbations to a certain range. $\text{PGDL}_2$ and $\text{PGDL}_\infty$ also introduced in [9], use different norms for regulating the perturbations – $L_2$ norm in the case of $\text{PGDL}_2$ and $\infty$ norm in the case of $\text{PGDL}_\infty$. The Targeted Projected Gradient Descent (TPGD) attack with a random start, also from [9], is a more complex method that tends to perform better than the other baseline methods.

Carlini and Wagner [14] introduced the CW attack, which treats the problem of generating adversarial examples as an optimization problem and aims to minimize the difference between the original and adversarial samples while maximizing the model's wrong classification probability. The CW loss function includes adjustable hyperparameters and parameters that control the confidence of the generated adversarial samples. Although the CW attack is one of the strongest white-box attacks, it is slow due to the line search and may require thousands of iterations.

While the effectiveness of $L_p$ distance as a perceptual quality metric is still being researched, Xiao et al. [22] focused on spatial transformation rather than direct manipulation of pixel values for perturbation generation. Modas et al. [10] proposed SparseFool, which uses the low mean curvature of the decision boundary to control the sparsity of perturbations, resulting in more efficient attacks on high-dimensional data. In general, various methods for generating perturbations have been proposed and tested for their effectiveness in perceptual tasks. However, few works have been able to control the quantity and quality of perturbation generation. In this paper, based on the above work, we propose an adversarial attack model based on a generative model that can quickly and effectively generate adversarial samples with a fixed number of pixel perturbations.

### 2.2. Black-box attacks

Black-box attacks can be classified into three types based on the way they are deployed. The first type involves training a substitute model to approximate the target model and use it to create adversarial samples. For instance, Papernot et al. [23] trained a local model to replace the target DNN and generate adversarial samples. Zhang et al. [24] proposed a perturbation generation framework that used a neural approximation function to find the best camouflage to conceal the attack. An alternative approach is to use higher-order approximation to eliminate the need for a substitute model. For example, the zero-order optimization (ZOO) attack proposed by Chen et al. [25] generates adversarial samples by directly estimating the gradient of the target DNN.

The second type of attack involves finding the decision boundary, which can be approximated directly. For example, the decision-based boundary attack proposed by Brendel et al. [26] starts with a large adversarial perturbation and then iteratively reduces the perturbation while maintaining its adversarial nature. This approach is relatively easy to implement as it requires minimal hyperparameter tuning and does not depend on the use of auxiliary models.

The third type of attack involves finding a universal perturbation. For example, Moosavi et al. [27] proposed a universal adversarial perturbation attack (UAP) by accumulating perturbations on a single input and adding the generated perturbations to each data sample, repeating the process until the sample is misclassified.

Dues to lacking information of target model, it takes time to estimate the gradient or target model. Several researchers have proposed methods to improve the efficiency of adversarial perturbation attacks, which aim to fool machine learning models by making small changes to input data. Du et al. [28] proposed a meta-learning-based black-box attack method that uses prior information from successful attacks to optimize the perturbation. Ru et al. [29] used Bayesian optimization to find successful perturbations with high query efficiency. Su et al. [11] proposed a semi-black box one-pixel attack that modifies a feature of a given image using differential evolution to find the perturbations. Wang et al. [30] introduced an aggregation gradient to obtain the importance of features and destroy important object perception features that influence the decision-making of the target model. Wang et al. [31] created transferable adversarial samples by mixing the gradient of the input image with a small part of each plug-in image while maintaining the original label of the input. Different from those works, we mainly focus on the generative model that can locate the adversarial points in an efficient way.

## 3. Models

We mentioned that the training process involves two steps: *shadow model training* and *perturbation generation*. During *perturbation generation*, we further divide it into two parts: first, we train a well-trained auto-encoder, and then we identify the points to perturb. From our observations, it is challenging to determine the exact locations and values that can cause havoc. As mentioned before, it is more effective to add the perturbation $\sigma$ to the outline of the image $x_o \in x$, as this ensures stealth. The first step in this process is to find the outline $x_o$ on the image. Typically, this is done using the Jacobean to find the first-order features, but in this paper, we use a generated-based model to actively learn the adversarial sample in order to generate more effective perturbations. Specifically, in this paper, we use a VAE as the generative model, although in some other cases, it could also be a GAN. To make the generated adversarial example located in the silhouette, a dual-decoder is adopted. For clarity, we depict the proposed structure in Fig. 1.

There are three parts in the model: $x$ is the input image, $D$ denotes the encoder (a three-convolutional layer with ReLU as the activation function, where the orange blocks represent the convolution layers and the blue blocks represent the batch normalization layers that are specified with the output of the convolution layer), $z_p$ and $z_s$ (shown as yellow blocks) are the latent variables for the dual-decoder, and $\mathcal{H}_p$ and $\mathcal{H}_s$ are the decoders where $\mathcal{H}_p$ generates the perturbation $\varepsilon_p$ and $\mathcal{H}_s$ generates the silhouette $\varepsilon_s$. Both of these decoders consist of three deconvolutional layers with ReLU as the activation function, shown as green blocks. $M$ and $\hat{M}$ are the target model and the shadow model, respectively. The target model is represented by the gray box in the image, while the shadow model is a three-layer convolutional layers model. However, if the victim model is more complex, a correspondingly more intricate model can be used as the shadow model. The output of these two models is connected to a fully-connected layer, represented by the rectangles in the image.

In order to create a successful shadow attack, a training process is followed which includes three main steps: model imitation (depicted in the blue box in Fig. 1), shadow attack points selection (depicted in the green box in Fig. 1), and perturbation optimization (depicted in the yellow box in Fig. 1). Specifically, The first two steps pertain to the *shadow model training*, while the optimization of perturbations belongs to the *perturbation generation* process. During the model imitation step, the shadow model is optimized through the use of knowledge distillation with the original model. The shadow attack points selection step involves identifying the most effective attack points on the skeleton of an object in an image using a dual-decoder. Finally, the perturbation optimization step involves using a specially designed adversarial loss to optimize the perturbation based on the performance of the well-trained shadow model and VAE. The use of a VAE helps to effectively incorporate task-specific features from various models, making it a crucial element in the generation of perturbations. Through these three steps, the objective function is able to accurately reflect the training process, resulting in a successful shadow attack.

### 3.1. Model imitation

To begin with, we will construct a substitute model, also known as a shadow model, to mimic the behavior of the target model through adversarial training. This imitation process is typically done through reinforcement learning, which involves training the shadow model to replicate the output of the target model, even though we have limited knowledge about how the target model works. The purpose of creating a shadow model is to identify vulnerabilities or points of attack in the target model. In order to do this effectively, the shadow model must be able to locate universal optimal adversarial points within an image. However, it can be time-consuming to obtain a comprehensive policy network through reinforcement learning.

In order to create a more effective shadow model, we can use a process called knowledge distillation to transfer knowledge from the teacher model (the target model) to the student model (the shadow model). One common method for doing this is training with KL-divergence, which distills information from the teacher model. However, in this case, the target model is complex and task-specific, making it difficult to transfer knowledge from scratch without knowing the details of the model. Additionally, it is difficult to determine the optimal size of the shadow model for efficient knowledge transfer. In order to achieve effective transfer in this work, we use dual-teacher knowledge distillation, only knowing the logit output of the target model, as described in Eq. (1).

$$L_{im} = y \log \hat{M}(x) + (\hat{M}(x) - M(x))^2 \tag{1}$$

In the equation, $M$ denotes the classification model as mentioned before (i.e., the teacher model), and $\hat{M}$ represents the shadow model (i.e., the student model), $x$ is the input, and the $y$ is ground truth. The first part of the right-hand side (RHS) is the distillation from the ground truth which guides the model in a direct way and ensures the student model obtains the basic knowledge of the data, then the second part of the RHS distills the knowledge from the target model directly. Through these two parts, the shadow model largely distills the knowledge from the target model.

### 3.2. Shadow attack points selection

After obtaining the shadow model, we need to identify the points that we want to attack. Previously, the attack models treat all the images as the attack points with the gradient descent, and the learned perturbation $\varepsilon \leq \sigma$ is well crafted in the small range. However, it is impractical to apply detailed perturbations over the entire image. Therefore, we need to determine which features are the most important for image processing and select specific points from these important features to generate the adversarial example. To find the important features, a specially designed VAE which includes a two-head decoder is applied and ensures the attack points are on the skeletons of the object with the reconstruction procedure. The encoder $D$ encodes the input into the latent space for feature extraction. It produces two pairs of outputs: one for perturbation localization, $z_s$, which is passed as input to the decoder $\mathcal{H}s$, and one for interference generation, $z_p$, which is passed as input to the decoder $\mathcal{H}_p$. As described earlier, the

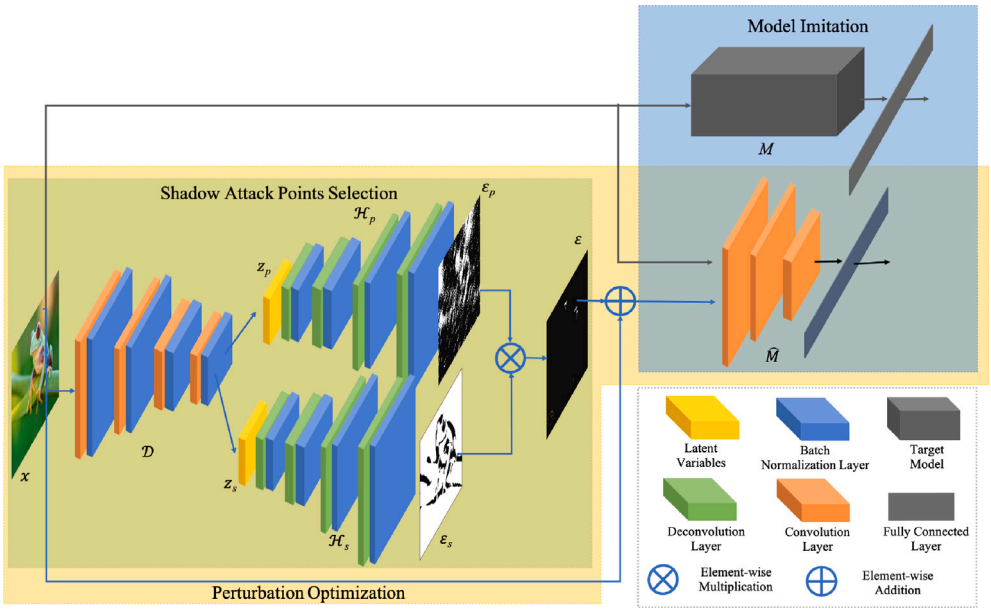


**Fig. 1.** The structure of the proposed model, the blue box denotes the model imitation, the yellow box denotes the shadow attack points selection and the green box denotes the perturbation optimization. The function of each block is described in the corner below. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

perturbation submerged in the structure of the image can assist in the concealment of the attack, especially in the SAR-based image, i.e., the high-contrast image. Therefore, the perturbation is selected with top-$k$ values from $\varepsilon_s$ (i.e., the output of $\mathcal{H}_s$) where we assume that top-$k$ values are the most important features that can be utilized, i.e., $p$ = top-$k(\varepsilon_s)$, and this value is acted as the mask to control the place that needs to be selected, and $k$ denotes the number of the perturbed pixels. The value of the perturbation is then determined by $\varepsilon_p$ (i.e., the output of $\mathcal{H}_p$), which is optimized using the reconstruction loss (i.e., the expectation value of the expected negative log-likelihood for all data points) and the regularizer (i.e., the Kullback–Leibler ($\mathbb{KL}$) divergence between the posterior $\mathcal{H}_s(z_s)$ and prior distributions of the encoder $\mathcal{D}(z_s|x) \sim \mathcal{N}(z_s|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ , where $\mathcal{N}$ is the normal distribution, $\boldsymbol{\mu}_0$ is the mean of the distribution, and $\boldsymbol{\Sigma}_0$ is the covariance matrix of the distribution), and this is described in Eq. (2).

$$L_{re} = \mathbb{E}_{z_s \sim \mathcal{D}(z_s|x)}[\log \mathcal{H}_s(x|z_s)] - \mathbb{KL}(\mathcal{D}(z_s|x)||\mathcal{H}_s(z_s)) \tag{2}$$

The perturbation is calculated as follows:

$$\varepsilon = \mathbb{1}(\varepsilon_s \leq p) \cdot \varepsilon_p \tag{3}$$

where $\mathbb{1}$ denotes the indicator function when the value is true, it will be 1 in that place. The variable of $\varepsilon_p$ is optimized with the adversarial loss described in Eq. (4). After the perturbation is selected, we add it to the image directly to form the adversarial sample.

### 3.3. Perturbation optimization

The first two steps involve constructing the adversarial attack. To guide the entire procedure, a carefully designed adversarial loss is employed. And guarantees the disturbances generated by the model can be antagonistic. We also applied the optimized-based attack to construct our adversarial examples. Generally, a usual way to construct the adversarial examples is to find an adversarial perturbation $\varepsilon$ that $M(x+\varepsilon) \neq y, s.t. \varepsilon \leq \sigma$. And this is optimized with the function $\min c \cdot \|\varepsilon\|_2^2 - \text{loss}_{M,y}(x+\varepsilon)$, where loss is a common loss function which usually is cross-entropy, and $c$ is a positive constant member to yield the stealthy adversarial examples. However, it is difficult to find a proper $\varepsilon$ as $M(x+\varepsilon) \neq y$ is highly non-linear. In works [14], they solve this problem by changing it to an appropriate optimization instance, e.g., $\varepsilon_i = \frac{1}{2}(\tanh(w_i)+1) - x_i$ in an optimization iteration. This approach provides the opportunity for perturbation learning, however, it still needs multiple iterations to find the optimized approximate value of the perturbation. To cope with this problem, and reduce the learning time, we construct a neural-based model to learn the perturbation. The neural network here is the dual-decoder VAE mentioned in Section 3.2. It can also be another generative model e.g., generative adversarial networks (GAN). After obtaining the most sensible place with top-$k$ values from $\varepsilon$, we need to ensure the distance between the original image and its adversarial is close, which can be described in $\min \|\varepsilon\|_p$, which can be the different $p$-norm (e.g., $p=1$, $p=2$ and $p=\infty$, etc.). Another term which is to limit the example to be adversarial, the objective function can be described as $\min(\max(\text{Avg}(\hat{M}(x+\varepsilon)_i : i \neq t) - \hat{M}(x+\varepsilon), -\tau))$, where $\text{Avg}(\hat{M}(x+\varepsilon)_i : i \neq t)$ is the average value of the output when $i \neq t$, $\tau$ is the super-parameters controls the degree of the attack, $t$ is target category, and $i$ is current predicted category. Therefore, the loss in this work is described as in Eq. (4).

$$L_{adv} = \|\varepsilon\|_p + \max(\text{Avg}(\hat{M}(x+\varepsilon)_i : i \neq t) - \hat{M}(x+\varepsilon), -\tau) \tag{4}$$

The first term indicates that the adversarial sample and the original image should be close, which is to ensure concealment. The second term denotes the adversarial loss that guides the perturbation generation. There are different ways to find the optimized points for the perturbation generation. In this work, the gradient information that propagates from the shadow model described in Section 3.1 is adopted in the optimization. Therefore, the objective function of this framework is described as followed:

$$\begin{aligned} L &= \alpha L_{im} + (1-\alpha)(L_{adv} + \gamma L_{re}) \\ \alpha &= \begin{cases} 1 & \text{IF } T < N \\ 0 & \text{ELSE} \end{cases} \end{aligned} \tag{5}$$

Where $\alpha$ is the indication marker that determines the model training procedure, $T$ denotes the number of training iterations, which means that in the first $T$ epochs, we need to train the shadow model well and then optimize the perturbations after the $T$ iteration training, and in the second term, $\gamma$ is the hyperparameter that balances the different losses in the same optimization space.

**Table 1**
The results of the target model on three datasets, as determined by the oracle, are presented in the first two columns. The remaining three columns provide details about each dataset.

| | VGG16 | ResNet | Train | Test | Category |
|---|---|---|---|---|---|
| Fashion MNIST | 92.18% | 85.85% | 60,000 | 10,000 | 10 |
| CIFAR10 | 82.67% | 83.72% | 50,000 | 10,000 | 10 |
| CIFAR100 | 61.40% | 56.11% | 50,000 | 10,000 | 100 |

### 3.4. Different with previous works

There are plenty of models that apply the substitute method to conduct black box attacks, but it is unclear why these substitute models are effective. It can be difficult to determine which specific parts of the model need to be modified, but even small changes can have significant consequences. In this paper, we propose a new method for generating adversarial samples that focuses on a small number of targeted points with high efficiency. To do this, we use a substitute model called a shadow model, which is trained using knowledge distillation. We also employ a VAE to create subtle perturbations in specific locations and values in the original image, making the perturbations virtually invisible. This method allows us to attack a multi-layered neural network model with minimal effort, and the perturbations can be directed toward an approximate target model.

## 4. Experiments

In this section, we run all experiments five times and record the average results in order to validate the effectiveness of the proposed model. The evaluation metrics for these experiments include the peak signal-to-noise ratio (PSNR) which measures the quality of the perturbed image, the attack decrease rate (ADR) which evaluates the attacking efficiency of the model, and the dot that changed (DTC) which counts the number of perturbed image dots. Especially, the PSNR is calculated as:

$$\mathrm{PSNR} = 10 \cdot \log_{10}\left(\frac{\max_{\hat{x}}^2}{\mathrm{MSE}}\right) \tag{6}$$

where MSE is the mean squared error which is calculated with

$$\mathrm{MSE} = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[\hat{x}(i,j) - x(i,j)]^2 \tag{7}$$

where $m, n$ denote the scale of the image, and $\max_{\hat{x}}$ is the maximum possible value of the image. Different from previous work, to obtain a robust evaluation of the attack success, ADR is calculated on all of the test datasets with:

$$\mathrm{ADR} = \frac{r_s - r_a}{r_s} \tag{8}$$

$r_s$ is the test accuracy on the normal data, and $r_a$ denotes the test accuracy on the adversarial data. DTC is the number of the perturbation added on. We performed experiments on three datasets: FashionMNIST, CIFAR10, and CIFAR100. The target models utilized in these experiments were VGG16 and ResNet, which were pre-trained on ImageNet and then fine-tuned on the aforementioned datasets. The results of these experiments are presented in Table 1.

In this paper, we consider both white-box and black-box attack scenarios using the state-of-the-art (SOTA) attack model. In the white-box scenario, the attacker has access to the model parameters, while in the black-box scenario, the attacker does not have access to these parameters. Further details on these scenarios will be provided in the following section.

### 4.1. Baselines

In this paper, the models we compared include:

- **FGSM** [15] is one of the classic white-box attack methods which deploy the attack by using chain rules and finding the desired gradients, the maximum perturbation is set as 8/255 to the best performance. While, there are many variants based on FGSM, to make comprehensive comparisons, we also listed them in the compared models.
  - **BIM** [21] applied perturbation to physical world images using a basic iterative method with a maximum perturbation of 8/255, step size of 5/255, and a number of steps of 20.
  - **MIFGSM** [32] improved FGSM with iterative momentum, using a maximum perturbation of 8/255, step size of 5/255, and a number of steps of 20.
  - **RFGSM** [17] used a randomization-step attack to generate perturbations with a maximum perturbation of 8/255, step size of 5/255, and a number of steps of 20.
  - **DIFGSM** [20] used a maximum perturbation of 8/255, step size of 5/255, and a number of steps of 20.
  - **FFGSM** [19] is a fast version of FGSM that speeds up the process, using a maximum perturbation of 8/255 and step size of 5/255.
- **PGD** [33] is another iterative attack model which searches for the adversarial perturbation through one-more steps iteration rather than a single step that is used in FGSM. In the application, the maximum perturbation is 8/255, the step size is 5/255, and the step number is 20.
  - **PGDL$_2$** [33] is a variant of PGD that proposed in [33]. In the application, the maximum perturbation is 8/255 and the step size is 5/255 with the step number 20.
  - **TPGD** [34] is to trade off the balance between robustness and accuracy by considering the theory of classification-calibrated loss. In the application, the maximum perturbation is 8/255, the step size is 5/255, and the step number is 20.
  - **APGD** [35] is another SOTA in the adversarial attack. In the application, the maximum perturbation is 8/255, and the step number is 20.
- **CW** [14] deploys the attack with the distilling method which inspires our work in the adversarial objective function designing, and it has become one of the most effective white-box attack models. In the application, the box constraint is 0.5 and the optimization step is 1000 with a learning rate of 0.05.
- **DeepFool** [36] is to find the smallest perturbation that misleads the classifier. In the application, the step number is 20, and the parameter for enhancing the noise is 0.02.
- **SparseFool** [10] is to mislead the classifier with the fewest perturbed points by using a linear approximation to obtain the decision boundary, the perturbed points are fewer than other models. In the application, the step number is 20, the parameter for scaling noise from DeepFool is 3, and the parameter for enhancing the noise is 0.02.
- **OnePixel** [11] is to generate one-pixel adversarial perturbation by differential evolution, and the goal of this model is most relevant to our work. In the application, the step number is 20, the population size in differential evolution is 400, and the pixel number is selected in the range of $[1, 5]$ as needed.

In this study, we conducted a black box attack on a target model using a shadow model with a three-layer convolutional neural network. The shadow model was trained using model imitation, as described in Section 3.1. The attack was performed using the SOTA techniques

mentioned above, which involved generating perturbations over the shadow model. The effectiveness of the attack was evaluated in a black box setting, where the internal structure of the target model was unknown to the attacker. It is the same case in our attack scenario where the shadow model is a three-layer convolutional neural network with VAE as the perturbation generator, the kernel size of the convolutional layers is [3, 4, 5], and the encoder of VAE is a four-layer convolutional model by setting all kernel size to 4, and the decoder is a four-layer deconvolutional layer with kernel size as 4. The details of the structure are shown in Fig. 1. In our attack scenario, we need to first train the shadow model to have a well-trained alternative model by optimizing the imitation loss $L_{im}$. Then we need to use the adversarial loss $L_{adv}$ together with $L_{re}$ to learn the adversarial samples. The performance of the model is described in the following sections.

### 4.2. Model performance

The performance is evaluated with adversarial attack and adversarial robustness respectively.

#### 4.2.1. Adversarial attack

Specifically, in the black box attack scenario for the SOTA attack models, the target model is a well-trained shadow model used to generate perturbations. These perturbations are added to an image, which is then fed into the target model. The attack results on these three datasets are listed in Tables 2–4. From Table 2, we can see that our model is not the best in terms of PSNR performance, since we assume that perturbations can be added manually in the physical world, there is no precise limitation on perturbations. While in the view of ADR, our model achieves the best performance in all cases. The main reason is that our model explores global perturbations in a general way that guarantees good attack performance in the black-box setting and also good transferability. Another important feature is that a fixed number of pixels that can be controlled are perturbed to construct the adversarial examples. In Tables 3 and 4, it is 15 pixels are perturbed on CIFAR10 and CIFAR100, and 5 pixels are perturbed on Fashion MNIST based on the performance in Table 2. In general, FGSM-based and PGD-based methods compute perturbations in all image spaces. It is difficult to produce such an adversarial example in the physical world, though they have higher PSNRs which indicate better stealthy. Our model generally achieves better attacking results with fewer perturbed pixels compared to SparseFool, with the exception of ResNet on CIFAR100 and VGG16 on Fashion MNIST. In some cases, our model also has better performance in terms of PSNR. This is mainly because SparseFool can efficiently find the optimal place to make the fewest number of perturbations using its gradient-based method. Compared to OnePixel, the situation is almost the same as SparseFool, where our model obtains higher ADR for the same perturbed pixels and also has competitive PSNRs. Surprisingly, $PGDL_2$ has the best PSNRs in almost all cases, but the worst ADRs, due to the enhanced similarity between the adversarial and original examples by the $L_2$-norm.

In Eq. (4), the perturbation $\epsilon$ constraints have different norms available. We list these norms in the last three rows of the tables and compare their performance. The results show that the $L_2$-norm consistently outperforms the $L_1$-norm and $L_\infty$-norm in all cases. Therefore, the $L_2$-norm is more suitable for optimizing the decision boundary with the generative model. As a result, the $L_2$-norm will be adopted in all subsequent experiments.

We also illustrate the image that was attacked by different methods, and the results are listed in Figs. 2 and 3. From the figure, we can see that in the OnePixel attack and SparseFool attack, the attack point can be easily perceived. In attack methods e.g., FGSM and PGD, we can observe that the backgrounds are blurred and all images are actually corrupted. While in the cases of CW and DeepFool, the images are almost the same as the original, but the ADR is their shortcomings.

**Table 2**
The attack results on Fashion MNIST, the number in bold denotes the best results, the higher the better for the ADR and PSNR and the lower the better for the DTC.

| Attacks | ResNet | | | VGG16 | | |
|---|---|---|---|---|---|---|
| Measurements | PSNR | ADR | DTC | PSNR | ADR | DTC |
| FGSM | 75.54 | 2.82% | 583.9 | 75.53 | 3.28% | 585.0 |
| BIM | 83.62 | 0.26% | 584.8 | 83.62 | 0.76% | 584.3 |
| MIFGSM | 79.89 | 1.48% | 593.4 | 79.89 | 1.55% | 591.9 |
| FFGSM | 80.18 | 1.19% | 585.5 | 80.18 | 1.28% | 585.5 |
| DIFGSM | 80.79 | 0.76% | 636.6 | 80.83 | 1.07% | 635.7 |
| RFGSM | 76.21 | 2.97% | 414.0 | 76.20 | 2.82% | 415.4 |
| PGD | 80.27 | 0.62% | 621.7 | 80.24 | 1.48% | 616.7 |
| $PGDL_2$ | **107.83** | 0.29% | 600.4 | **107.59** | 0.99% | 606.4 |
| TPGD | 91.91 | 0.09% | 49.4 | 91.37 | 0.00% | 50.6 |
| APGD | 80.42 | 0.99% | 622.4 | 80.48 | 1.28% | 622.5 |
| AutoAttack | 88.880 | 0.73% | 88.8 | 89.13 | 0.71% | 84.7 |
| CW | 105.70 | 1.77% | 125.3 | 100.00 | 0.00% | 106.0 |
| DeepFool | 101.09 | 2.11% | 69.0 | 93.98 | 0.34% | 98.4 |
| SparseFool | 73.86 | 3.75% | 7.1 | 82.25 | 0.88% | **1.2** |
| OnePixel | 73.84 | 9.81% | **5.0** | 73.81 | 2.03% | 5.0 |
| Our-$L_1$ | 71.97 | 13.29% | **5.0** | 70.11 | 2.65% | 5.0 |
| Our-$L_2$ | 71.81 | **22.35%** | **5.0** | 71.25 | **3.82%** | 5.0 |
| Our-$L_\infty$ | 70.48 | 4.17% | **5.0** | 72.07 | 2.81% | 5.0 |

**Table 3**
The attack results on CIFAR10, the number in bold denotes the best results, the higher the better for the ADR and PSNR and the lower the better for the DTC.

| Attacks | ResNet | | | VGG16 | | |
|---|---|---|---|---|---|---|
| Measurements | PSNR | ADR | DTC | PSNR | ADR | DTC |
| FGSM | 78.53 | 9.51% | 3053.3 | 78.44 | 10.89% | 2938.2 |
| BIM | 79.06 | 8.92% | 2900.3 | 79.04 | 9.34% | 2927.9 |
| MIFGSM | 78.50 | 9.51% | 2956.5 | 78.46 | 10.41% | 2985.8 |
| FFGSM | 81.44 | 4.32% | 3053.9 | 81.42 | 5.50% | 3056.4 |
| DIFGSM | 81.31 | 4.00% | 1635.3 | 80.04 | 3.68% | 3053.0 |
| RFGSM | 79.20 | 8.55% | 2884.2 | 79.01 | 9.52% | 3046.1 |
| PGD | 79.00 | 8.92% | 3055.1 | 79.00 | 9.31% | 3055.3 |
| $PGDL_2$ | **113.08** | 0.32% | 2876.8 | 83.01 | 1.68% | 3051.4 |
| TPGD | 79.05 | 3.82% | 3054.7 | 79.80 | 4.50% | 3056.0 |
| APGD | 82.45 | 2.54% | 1377.8 | 82.42 | 3.44% | 1382.7 |
| AutoAttack | 89.91 | 8.78% | 266.98 | 89.78 | 1.74% | 274.3 |
| CW | 108.22 | 0.14% | 889.2 | 104.31 | 5.15% | 308.3 |
| DeepFool | 91.30 | 1.19% | 1807.2 | **104.49** | 0.11% | 289.4 |
| SparseFool | 67.29 | 14.45% | 315.1 | 74.37 | 16.43% | 24.4 |
| OnePixel | 76.03 | 8.35% | 15.0 | 77.47 | 12.13% | 15.0 |
| Our-$L_1$ | 76.43 | 11.25% | **14.9** | 78.48 | 8.23% | **14.9** |
| Our-$L_2$ | 76.01 | **14.75%** | **14.9** | 75.91 | **19.45%** | **14.9** |
| Our-$L_\infty$ | 77.48 | 13.20% | **14.9** | 76.53 | 10.22% | **14.9** |

In the scenario of white-box attacks, we directly replace the shadow model with the target model, whose output is directly used to guide the generation of perturbations. We validated the performance on the CIFAR dataset with ResNet as the target model. In this attack scenario, we only compared with SparseFool and OnePixel as they have a limited number of perturbed pixels as we do. For a fair comparison, while, we could not set the number of SparseFool attack points directly, therefore it is necessary to first get the number of perturbed points from it, and then set the perturbed points with the same constant (it is 84 in this setting) in OnePixel and our model, and the results are listed in Table 5.

From the table, we can see that OnePixel has the best performance in both PSNR and ADR, while it spends the most time because it needs time to search for the best points with an evolution-based approach. But in some practical applications, it is unacceptable to wait for this attack. SparseFool comes last in terms of ADR, and it takes more than 2 s to complete an attack on an image. While, from the last row, we can see that our model achieves 75.26% in terms of ADR, which is in 2nd place, but the time it takes on a single image is 5e-4 s, which is largely superior to the other two models. From this point, we can see that our model is able to achieve efficient attacks in a fast way.

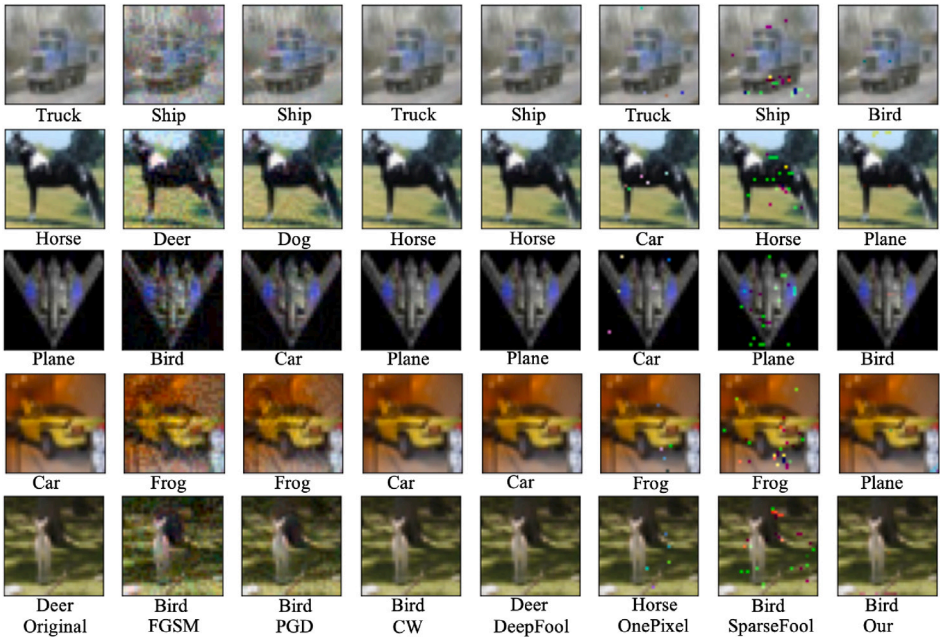


**Fig. 2.** The attack results with different methods on CIFAR10.

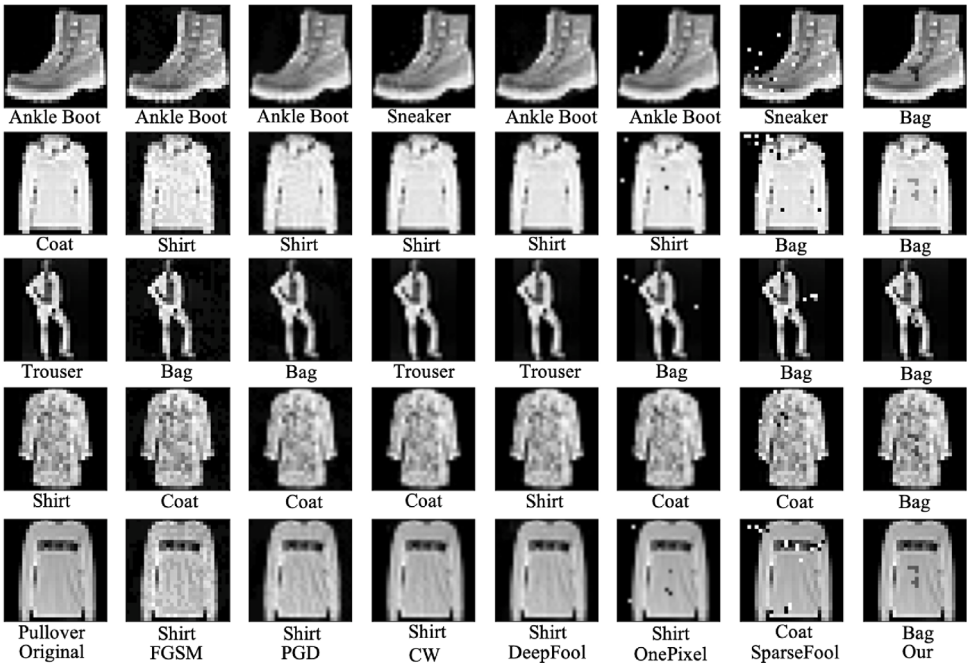


**Fig. 3.** The attack results with different methods on Fashion MNIST.

*4.2.2. Adversarial robustness*

To assess the adversarial robustness of the model, we implemented adversarial training with VGG16 on CIFAR10 using multiple models and recorded the results in Table 6. The first column, labeled "AT", represents the performance of the adversarially trained model when subjected to the attack methods listed in that row. The remaining columns represent the performance of the target model after adversarial training under attack, with the attack methods listed in each column.

Our model achieved the best adversarial training results with an accuracy of 89.52%. Our results show that the same model attack usually performs the best, for example, the FGSM attack performs the best when the adversarial training uses FGSM. From the table, we can see that all adversarial training methods were able to prevent OnePixel attacks, which had an accuracy of around 40% in all cases. This is likely because OnePixel attacks only perturb a few pixels, making it difficult for the model to improve its performance through adversarial training. When using adversarial training with PGD, we found that FGSM also had the best resistance to attack, followed by PGD which improved the model's performance under attack. Surprisingly, adversarial training with CW was not able to resist other attack models, suggesting that the optimization method with CW may have better-attacking performance, but the perturbations it generates do not effectively improve the model's performance. Overall, our model had the best performance under attack in almost all cases, indicating that the perturbations generated by our model make the decision boundary more flexible and improve the model's robustness globally.

**Table 4**
The attack results on CIFAR100, the number in bold denotes the best results, the higher the better for the ADR and PSNR and the lower the better for the DTC.

| Attacks | ResNet | | | VGG16 | | |
|---|---|---|---|---|---|---|
| Measurements | PSNR | ADR | DTC | PSNR | ADR | DTC |
| FGSM | 78.35 | 3.97% | 3024.6 | 78.37 | 14.35% | 3014.2 |
| BIM | 79.52 | 2.16% | 2985.9 | 79.44 | 10.16% | 2987.8 |
| MIFGSM | 78.52 | 3.10% | 3026.5 | 78.52 | 12.85% | 3027.7 |
| FFGSM | 81.44 | 0.68% | 3041.0 | 81.39 | 5.96% | 3039.9 |
| DIFGSM | 80.20 | 2.64% | 3038.7 | 80.09 | 7.30% | 3039.3 |
| RFGSM | 79.54 | 2.44% | 3040.6 | 82.40 | 4.46% | 3034.5 |
| PGD | 79.56 | 2.28% | 3041.3 | 79.46 | 9.43% | 3041.1 |
| $PGDL_2$ | **113.08** | 0.02% | 3024.8 | **113.08** | 0.07% | 3021.1 |
| TPGD | 79.38 | 3.14% | 3040.8 | 79.48 | 12.51% | 3041.3 |
| APGD | 98.54 | 0.02% | 33.5 | 99.14 | 0.15% | 22.4 |
| AutoAttack | 98.60 | 0.29% | 28.1 | 98.76 | 0.03% | 30.0 |
| CW | 102.15 | 0.02% | 975.8 | 106.18 | 0.03% | 986.4 |
| DeepFool | 108.30 | 0.03% | 23.9 | 83.24 | 0.65% | 30.6 |
| SparseFool | 90.30 | 0.71% | **7.9** | 83.18 | 0.65% | 30.4 |
| OnePixel | 78.28 | 1.51% | 15.0 | 78.28 | 10.00% | 15.0 |
| $Our\text{-}L_1$ | 79.49 | 14.83% | 14.9 | 76.88 | 8.66% | **14.9** |
| $Our\text{-}L_2$ | 76.33 | **18.41%** | 14.9 | 76.64 | **23.49%** | **14.9** |
| $Our\text{-}L_\infty$ | 77.22 | 15.38% | 14.9 | 78.95 | 10.96% | **14.9** |

**Table 5**
Attack performance in the white-box setting targeting ResNet on the CIFAR10 dataset, Time denotes the time used to attack a single image, measured in seconds ($s$).

| Attacks | PSNR | ADR | Time |
|---|---|---|---|
| SparseFool | 70.10 | 69.79% | 2.2 |
| OnePixel | 72.25 | 84.03% | 7.0 |
| Our | 69.04 | 75.26% | 5e-4 |

**Table 6**
The results of the robustness improvement after the adversarial attack.

| Method | AT | FGSM | PGD | CW | SparseFool | OnePixel |
|---|---|---|---|---|---|---|
| FGSM | 84.53% | **62.70%** | 2.76% | 5.19% | 7.43% | 38.00% |
| PGD | 84.63% | 48.32% | 26.85% | 18.23% | 14.25% | 41.11% |
| CW | 84.08% | 13.70% | 0.47% | 20.47% | 18.00% | 47.00% |
| SparseFool | 84.06% | 8.34% | 2.45% | 14.09% | 23.57% | 40.30% |
| OnePixel | 84.23% | 8.61% | 2.17% | 13.76% | 29.56% | 51.52% |
| Our | **89.52%** | 59.19% | **36.68%** | **26.81%** | **35.80%** | **52.50%** |

### 4.3. Ablation study

There are many questions that need to be answered regarding the validity of the proposed model. For example, are all the components of the model necessary, would our model perform better if we increased the number of perturbed pixels or increased the degree of perturbation, do shadow models really work, and would a more powerful shadow model help the performance of the model? In this section, to answer these questions, we will validate the effectiveness of the proposed structure and explore the impact of the properties of the proposed model on the attack performance, i.e., the number of pixels, the bounds of perturbation, and the capability of the shadow model.

#### 4.3.1. Structure ablation

validate the proposed structure, we elaborate on whether the components in the structure are in effect. Therefore, in the experiments, we first remove the knowledge distillation, i.e., train a profound imitation model, and use a randomly initialized model as the shadow model (/SM). Then we remove the VAE training which is to process the shadow attack (/NA) generation. And the results are listed in Table 7.

By analyzing the results presented in the table, it is clear that the VAE component significantly improves the performance of the model. This can be observed through the increase in PSNR values, which indicate the stealthiness of the attack. The inclusion of this component is essential to achieving our goal of improving the stealthiness of the attack.

Additionally, the shadow model plays a crucial role in the generation of adversarial examples. Without this component, the model would lack the necessary guidance information and may produce confusing training results. However, when this component is included, we see improved PSNR values compared to other cases. This demonstrates the importance of the shadow model in the proposed model.

**Table 7**
The ablation study results with different components.

| Model | VGG | | ResNet | |
|---|---|---|---|---|
| Measurement | PSNR | ADR | PSNR | ADR |
| ALL | 75.90 | **19.45%** | 76.01 | **14.75%** |
| /NA | 76.42 | 9.19% | 75.90 | 8.71% |
| /SM | **80.45** | 4.03% | **82.16** | 1.90% |

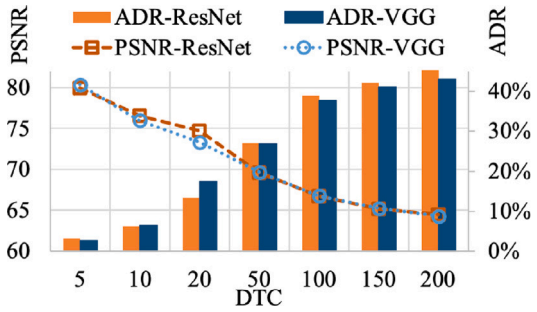


**Fig. 4.** Model performance for a different number of perturbed pixels, i.e., the $x$-axis in the figure. The line is the PSNR and the columns bar indicates the ADR. The orange-brown color indicates the performance of ResNet and the blue color indicates the performance of VGG16. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

#### 4.3.2. Number of pixels

To answer the question of whether the number of pixels affects the performance, we increase the maximum number of pixels that can be perturbed in a linear manner and observe the performance of the model. The results are shown in Fig. 4.

From the figure, it is evident that both ResNet and VGG16 exhibit similar performance, with an increase in ADR at the expense of a decrease in PSNR. This indicates that increasing the number of perturbed pixels in the adversarial examples can improve the attack decrease rate, but at the cost of reduced stealth. To achieve better stealth, it is advisable to select a lower number of perturbed pixels. Alternatively, one can choose a high number of perturbed pixels, but no more than 100, to achieve a significant performance improvement while maintaining a certain level of stealth.

Then, to evaluate the effectiveness of our model, we compared its performance to that of FGSM and PGD under various levels of perturbation. Specifically, we varied the number of perturbed pixels and plotted the results in Fig. 5. The three nodes on each line represent perturbation levels of 20, 50, and 100 pixels, respectively. Our model was tested on two popular image classification networks, ResNet and VGG16, and compared to the results of FGSM and PGD at each perturbation level.

From the results in Fig. 5, it is clear that our model outperforms both FGSM and PGD on both ResNet and VGG16. In particular, our model achieves the best ADR among the three methods, followed by PGD and FGSM. However, our model has a relatively lower PSNR compared to PGD. Overall, our model demonstrates superior performance in terms of ADR while maintaining a competitive PSNR compared to the other two methods.

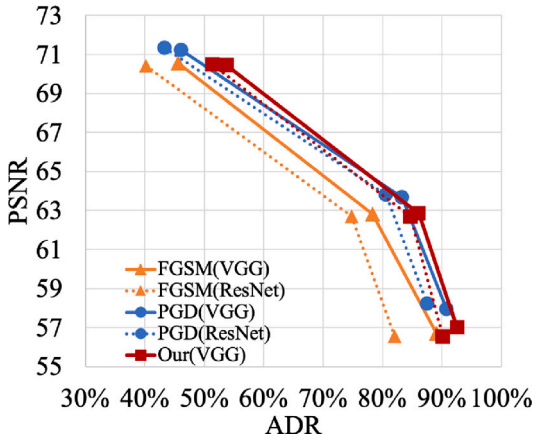


**Fig. 5.** The performance of the model with the different number of the perturbed pixels. The orange-brown lines indicate the performance of the FGSM, the blue lines are the performance of the PGD, and the red lines are the performance of our model. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

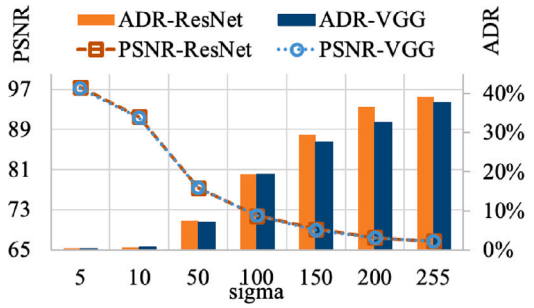


**Fig. 6.** The performance of the model with different boundaries of the perturbation, i.e., the $x$-axis in the figure. The line is the PSNR and the columns bar denotes the ADR. The orange-brown color indicates the performance of ResNet and the blue color indicates the performance of VGG16. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

#### 4.3.3. Boundaries of perturbation

As we described before, we do not have a constraint on the degree of perturbation as in the case of FGSM and PGD, nevertheless, what would happen to the performance of the model if we introduce such a constraint? To verify this question, we set boundaries of the perturbation value linearly and the results are illustrated in Fig. 6.

To further investigate the impact of perturbation on model performance, we implemented a constraint on the degree of perturbation in our testing. We varied the constraint on perturbation linearly and recorded the results, as shown in Fig. 6. This allowed us to determine the effect of imposing a constraint on perturbation on the model's performance.

As seen in the figure, the performance of our model improves as the maximum bound of the perturbation increases. However, this improvement comes at the cost of reduced stealth, as the attack becomes more noticeable. When the maximum bound of the perturbation is less than 10, our model has essentially no performance, but when the bound is larger than 100, the performance improves significantly. However, it should be noted that these results may not be applicable in real-world scenarios where there may not be a restrictive range on the degree of perturbation.

#### 4.3.4. Capability of shadow model

Another important question is whether shadow models affect perturbation learning and whether a more powerful model leads to a more powerful perturbation generator? To verify this question, a linear increment in training epochs was used to train the shadow model. And the results are shown in Fig. 7.

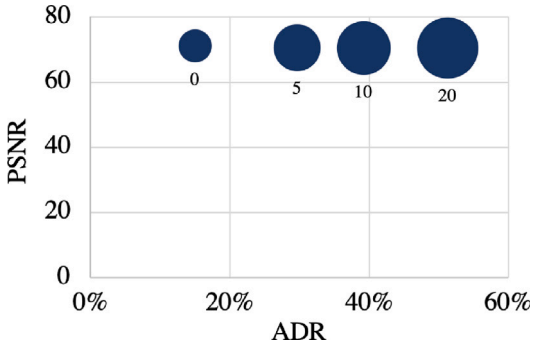


**Fig. 7.** The performance of the model when the shadow model is trained with different epochs. $Y$-axis is the PSNR and the $x$-axis denotes the ADR, the size of the circle represents the different training epochs i.e., the number under the circles.

From the figure, we can see that there is a significant improvement in ADR as the training epoch increases. From Section 4.3.1, we know that the shadow model is the essential part of our proposed model. We further validate here the effectiveness of a more powerful shadow model that distills knowledge from the original model and will nicely improve the attack performance of the model. Evidently, the purpose of the shadow model is to generate perturbations instead of the target model. In contrast to ADR, PSNR decreases modestly with increasing training epochs, and this can also be seen from Figs. 4 and 6. The effect of the model on the PSNR value is not substantial, i.e., our model has a relatively good guarantee in terms of attack concealment.

## 5. Conclusions and future works

In conclusion, our proposed framework for the few-pixels attack showed promising results in both effectiveness and efficiency compared to other existing methods. The dual-decoder VAE and modified adversarial loss helped to improve the effectiveness of adversarial learning, and the extensive experimental results and ablation study validated the validity of our model structure. However, there is still room for improvement, as the training of the generative model can be time-consuming and the perturbations may have negative impacts on the PSNR. In future research, we will aim to develop a more effective method for generating universal perturbations. Overall, this work represents a significant step forward in the field of few-pixel attacks, and we believe it will have important implications for the security and robustness of machine learning systems.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

The dataset we applied is opensource data.

## Acknowledgments

This work is funded by the National Natural Science Foundation of China (No. 62103330, 62233014).

## References


[1] Yang Li, Wei Zhao, Erik Cambria, Suhang Wang, Steffen Eger, Graph routing between capsules, Neural Netw. 143 (2021) 345–354.

[2] Gong Cheng, Ruimin Li, Chunbo Lang, Junwei Han, Task-wise attention guided part complementary learning for few-shot image classification, Sci China Inf Sci 64 (2021) 1–14.

[3] Guanjun Li, Shan Liang, Shuai Nie, Wenju Liu, Zhanlei Yang, Deep neural network-based generalized sidelobe canceller for dual-channel far-field speech recognition, Neural Netw. 141 (2021) 225–237.

[4] Yang Li, Suhang Wang, Quan Pan, Haiyun Peng, Tao Yang, Erik Cambria, Learning binary codes with neural collaborative filtering for efficient recommendation systems, Knowl.-Based Syst. 172 (2019) 64–75.

[5] Yang Bai, Yisen Wang, Yuyuan Zeng, Yong Jiang, Shu-Tao Xia, Query efficient black-box adversarial attack on deep neural networks, Pattern Recognit. 133 (2023) 109037.

[6] Jacob Steinhardt, Pang Wei W. Koh, Percy S. Liang, Certified defenses for data poisoning attacks, in: Proceedings of the Advances in Neural Information Processing Systems, Vol. 30, 2017.

[7] Uday Shankar Shanthamallu, Jayaraman J Thiagarajan, Andreas Spanias, Uncertainty-matching graph neural networks to defend against poisoning attacks, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, No. 11, 2021, pp. 9524–9532.

[8] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, Rob Fergus, Intriguing properties of neural networks, in: Proceedings of the International Conference on Learning Representations, 2014.

[9] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, Adrian Vladu, Towards deep learning models resistant to adversarial attacks, in: Proceedings of the International Conference on Learning Representations, 2018.

[10] Apostolos Modas, Seyed-Mohsen Moosavi-Dezfooli, Pascal Frossard, Sparsefool: a few pixels make a big difference, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 9087–9096.

[11] Jiawei Su, Danilo Vasconcellos Vargas, Kouichi Sakurai, One pixel attack for fooling deep neural networks, IEEE Trans. Evol. Comput. 23 (5) (2019) 828–841.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, Generative adversarial networks, Commun. ACM 63 (11) (2020) 139–144.

[13] Diederik P. Kingma, Max Welling, Stochastic gradient VB and the variational auto-encoder, in: Proceedings of the International Conference on Learning Representations, 2014.

[14] Nicholas Carlini, David Wagner, Towards evaluating the robustness of neural networks, in: Proceedings of the IEEE Symposium on Security and Privacy, IEEE, 2017, pp. 39–57.

[15] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, Pieter Abbeel, Adversarial attacks on neural network policies, in: Proceedings of the International Conference on Learning Representations (Workshop Track), 2017.

[16] Nicolas Papernot, Patrick Mcdaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, Ananthram Swami, The limitations of deep learning in adversarial settings, in: Proceedings of the IEEE European Symposium on Security and Privacy, 2016, pp. 372–387.

[17] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, Patrick McDaniel, Ensemble adversarial training: Attacks and defenses, in: Proceedings of the International Conference on Learning Representations, 2018.

[18] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, Jianguo Li, Boosting adversarial attacks with momentum, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 9185–9193.

[19] Eric Wong, Leslie Rice, J. Zico Kolter, Fast is better than free: Revisiting adversarial training, in: Proceedings of the International Conference on Learning Representations, 2019.

[20] Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, Alan L. Yuille, Improving transferability of adversarial examples with input diversity, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 2730–2739.

[21] Alexey Kurakin, Ian J. Goodfellow, Samy Bengio, Adversarial examples in the physical world, in: Artificial Intelligence Safety and Security, Chapman and Hall, 2018, pp. 99–112.

[22] Chaowei Xiao, Jun Yan Zhu, Bo Li, Warren He, Mingyan Liu, Dawn Song, Spatially transformed adversarial examples, in: Proceedings of the International Conference on Learning Representations, 2018.

[23] Nicolas Papernot, Patrick Mcdaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, Ananthram Swami, Practical black-box attacks against machine learning, in: Proceedings of the ACM on Asia Conference on Computer and Communications Security, 2017.

[24] Yang Zhang, P.D. Hassan Foroosh, Boqing Gong, CAMOU: Learning a vehicle camouflage for physical adversarial attack on object detections in the wild, in: Proceedings of the International Conference on Learning Representations, 2019.

[25] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, Cho-Jui Hsieh, ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models, in: Proceedings of the ACM Workshop on Artificial Intelligence and Security, 2017.

[26] Wieland Brendel, Jonas Rauber, Matthias Bethge, Decision-based adversarial attacks: Reliable attacks against black-box machine learning models, in: Proceedings of the International Conference on Learning Representations, 2018.

[27] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, Pascal Frossard, Universal adversarial perturbations, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 86–94.

[28] Jiawei Du, Hu Zhang, Joey Tianyi Zhou, Yi Yang, Jiashi Feng, Query-efficient meta attack to deep neural networks, in: Proceedings of the International Conference on Learning Representations, 2019.

[29] Binxin Ru, Adam D. Cobb, Arno Blaas, Yarin Gal, BayesOpt adversarial attack, in: Proceedings of the International Conference on Learning Representations, 2020.

[30] Zhibo Wang, Hengchang Guo, Zhifei Zhang, Wenxin Liu, Zhan Qin, Kui Ren, Feature importance-aware transferable adversarial attacks, in: Proceedings of the IEEE International Conference on Computer Vision, 2021, pp. 7639–7648.

[31] Xiaosen Wang, Xuanran He, Jingdong Wang, Kun He, Admix: Enhancing the transferability of adversarial attacks, in: Proceedings of the IEEE International Conference on Computer Vision, 2021, pp. 16158–16167.

[32] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, Jianguo Li, Boosting adversarial attacks with momentum, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 9185–9193.

[33] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, Adrian Vladu, Towards deep learning models resistant to adversarial attacks, in: Proceedings of the International Conference on Learning Representations, 2018.

[34] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, Michael Jordan, Theoretically principled trade-off between robustness and accuracy, in: Proceedings of the International Conference on Machine Learning, PMLR, 2019, pp. 7472–7482.

[35] Francesco Croce, Matthias Hein, Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks, in: Proceedings of the International Conference on Machine Learning, PMLR, 2020, pp. 2206–2216.

[36] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Pascal Frossard, Deepfool: a simple and accurate method to fool deep neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2574–2582.


**Yang Li** is an associate professor with the school of automation at Northwestern Polytechnical University, Xi'an, China. After receiving his bachelor's and doctoral degrees from Northwestern Polytechnical University in 2014 and 2018 respectively, he worked as a research fellow in SenticTeam under Professor Erik Cambria at Nanyang Technological University in Singapore and also was an adjunct research fellow at the A *STAR High-Performance Computing Institute (IHPC). His research interests are in Adversarial Attack & Defense in AI, NLP, Recommender System, Explainable Artificial Intelligence, etc. He has published several papers on these topics at international conferences and peer-reviewed journals. He is an active reviewer of several journals, e.g., INFORM FUSION, IEEE TAFF, NEUCOM, KBS, KAIS, etc. He is also an advisory board member of Socio-Affective Computing and the guest editor of Future Generation Computer Systems.

**Quan Pan** was born in China, in 1961. He received the B.S. degree in automatic control from the Huazhong University of Science and Technology, in 1982, and the M.S. and Ph.D. degrees in control theory and application from Northwestern Polytechnical University. From 1991 to 1993, he was an Associate Professor with Northwestern Polytechnical University, where he has been a Professor with the Automatic Control Department, since 1997. He has authored 11 books, more than 400 articles. His research interests include information fusion, target tracking and recognition, deep network and machine learning, UAV detection navigation and security control, polarization spectral imaging and image processing, industrial control system information security, commercial password applications and modern security technologies. He is an Associate Editor of the journal Information Fusion and Modern Weapons Testing Technology.

**Zhaowen Feng**, is a senior engineer and a tier-one expert of the Aviation Industry Corporation of China. He works as the associate chief engineer in the Information Security Institute of the Aviation Industry Development Center. His research interest includes but not limited in network security attack and defense, systems penetration testing, unmanned aerial vehicle (UAV) information security evaluation and industrial control systems security analysis. He received the bachelor degree and M.S. degree in Electrical Engineer from Beihang University and Chinese Aeronautical Establishment of China in 2010 and 2013, respectively. He currently also works on his Ph.D. program in Cyber Security with Northwestern Polytechnical University (NPU) in China.

**Erik Cambria** is the Founder of *SenticNet*, a Singapore-based company offering B2B sentiment analysis services, and an Associate Professor at *NTU*, where he also holds the appointment of Provost Chair in Computer Science and Engineering. Prior to joining NTU, he worked at Microsoft Research Asia and HP Labs India and earned his Ph.D. through a *joint programme* between the University of Stirling and MIT Media Lab. He is recipient of many awards, e.g., the 2018 *AI's 10 to Watch* and the 2019 *IEEE Outstanding Early Career* award, and is often featured in the news, e.g., Forbes. He is Associate Editor of several journals, e.g., NEUCOM, INFFUS, KBS, IEEE CIM and IEEE Intelligent Systems (where he manages the Department of *Affective Computing and Sentiment Analysis*), and is involved in many international conferences as PC member, program chair, and speaker.