Computers, Materials & Continua CMC, vol.62, no.3, pp.1365-1386, 2020

Defend Against Adversarial Samples by Using Perceptual Hash

Changrui Liu[1], Dengpan Ye[1, *], Yueyun Shang[2], Shunzhi Jiang[1], Shiyu Li[1], Yuan Mei[1] and Liqiang Wang[3]

Abstract: Image classifiers that based on Deep Neural Networks (DNNs) have been proved to be easily fooled by well-designed perturbations. Previous defense methods have the limitations of requiring expensive computation or reducing the accuracy of the image classifiers. In this paper, we propose a novel defense method which based on perceptual hash. Our main goal is to destroy the process of perturbations generation by comparing the similarities of images thus achieve the purpose of defense. To verify our idea, we defended against two main attack methods (a white-box attack and a black-box attack) in different DNN-based image classifiers and show that, after using our defense method, the attack-success-rate for all DNN-based image classifiers decreases significantly. More specifically, for the white-box attack, the attack-success-rate is reduced by an average of 36.3%. For the black-box attack, the average attack-success-rate of targeted attack and non-targeted attack has been reduced by 72.8% and 76.7% respectively. The proposed method is a simple and effective defense method and provides a new way to defend against adversarial samples.

Keywords: Image classifiers, deep neural networks, adversarial samples, attack defense, perceptual hash, image similarity.

1 Introduction

With the development of artificial intelligence, the DNN-based image classification models constantly refresh the record in the field of image recognition [Akhtar, Liu and Mian (2018); Huang, Liu, Van Der Maaten et al. (2017); Krizhevsky, Sutskever and Hinton (2012); Simonyan and Zisserman (2014); Szegedy, Liu, Jia et al. (2015)], which even surpass human beings. However, it does not mean that it has reached a real level of human beings, for that no matter how good the image classifier is, it will have serious errors in the face of a negligible attack. Image samples added with perturbations that cause errors in classification results are called "adversarial samples" [Szegedy, Zaremba, Sutskever et al. (2013)]. Most of the existing DNN models have little or no consideration for the existence of attackers when they are designed and implemented. This leads to the fact that although these models can perform perfectly in the face of natural input, they

[1] Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan, 430072, China.

[2] School of Mathematics and Statistics, South Central University for Nationalities, Wuhan, 430074, China.

[3] University of Central Florida, 4000 Central Florida Blvd. Orlando, Florida, 32816, USA.

[*] Corresponding Author: Dengpan Ye. Email: cherry_liucr@whu.edu.cn.

will make mistakes when encountering a large number of malicious users and even attackers in the real environment. Therefore, how to defend against adversarial samples is the key to ensure the security of the DNN models.

Up to now, there are many defenses methods have been proposed, e.g., Lu et al. [Lu, Issaranon and Forsyth (2017); Xie, Wang, Zhang et al. (2017)]. Although the current methods have achieved some good results, unfortunately, these defense methods have some limitations and can be easily bypassed by the evolutionary and variant adversarial samples. 1) For modifying training/input defense method, it requires a large number of normal samples and adversarial samples, which greatly increases the cost of time and space. On the other hand, it can only defend against specific attack methods, so it is not suitable for use in situations where the amount of data is large; 2) For modifying networks defense methods, most of them use a "gradient masking" method. The reason why they use gradient masking is that most white-box attacks operate by calculating the gradient of the DNN model, so if the effective gradient cannot be calculated, then the attack will fail, and gradient masking's main goal is making the gradient useless. But because the decision boundaries of models trained with the same distribution dataset will not be much different, the gradient masking-based defense methods are easily broken by black-box migration attack, for that the attacker can get a gradient in the "substitute"; 3) For network add-on defense method, the complete defense strategy generally adds other networks that need training, which leads to the reduction of classification accuracy to a certain extent. However, the detection-only strategy cannot solve the unknown risks such as 0-Day attacks.

To address the aforementioned challenges, we propose a new defense strategy based on image perceptual hashing [Lin and Chang (2001)], some results are shown in Fig. 1. Our approach is based upon the observation that in order to generate adversarial samples, existing attack methods adopt an iterative method (see Fig. 2 for an example). Specifically, most attack methods generate perturbations according to generation rules, then add the perturbations to the original image to generate a perturbed image, which is then inputted into the DNN to evaluate the perturbation according to the cost function. Repeat the process until the perturbed image causes the specified model to be misclassified, that is, attack success, in other words, an adversarial sample is generated. Based on this iterative update perturbation rule, we realize that by destroying the process of perturbations' generation, we can effectively prevent attacks on DNNs, which is also our core idea. Specific to the method (see Section 3.3 for more details), we added 1) "Judgment layer" on the DNN, whose main goal is determining whether the two images are similar by comparing the perceptual hash function and the difference of pixel values of two images. 2) "Time layer", this layer mainly compares the difference between the time cost before and after our defense method, and ensures that the use of defense will not make a big change in time, thus preventing the enemy from guessing the defense strategy through time difference.

This work proposes a new strategy using the perceptual hashing to defend against adversarial samples. The major contributions of this paper are as follows:

Effectiveness-Our key contribution is introducing and evaluating perceptual hashing as a new technique to defend against adversarial samples. We can destroy the process of generating adversarial samples. Furthermore, even if the attack algorithms have generated adversarial samples, we can still output the label of clean images to avoid the misclassification of DNNs. Using a white-box attack named Jacobian-based Saliency Map Attack (JSMA) [Papernot, McDaniel, Jha et al. (2016)] and a black-box attack named One Pixel Attack [Su, Vargas and Sakurai (2019)] methods as the defense targets, our defense strategy is shown definite results, whether it is a targeted attack or a non-targeted attack.

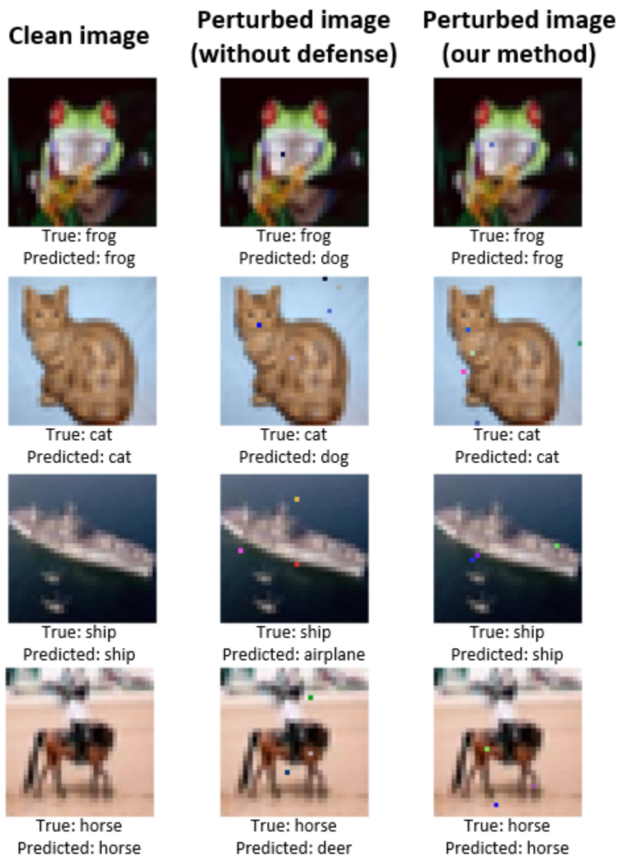

Figure 1: The proposed method that successfully avoided adversarial samples trained on CIFAR-10 dataset. First column: the original images. Second column: perturbed images that can fool DNNs. Third column: DNNs which added our proposed defense method, although there are adversarial samples, the DNNs can still predict the true label, rather than an unexpected label

Flexibility-Our defense strategy can be applied to a variety of DNNs without changing any structure or parameters of the target DNNs and is complementary to other defenses.

Inexpensive-Unlike adversarial training, our proposed defense strategy does not require any additional samples to the training datasets, which reduces the huge computational cost and space cost. In addition, although we do not need to add other additional networks as other defense methods do, for example, GAN [Goodfellow, Pouget-Abadie, Mirza et al. (2014)] and PRN (Perturbation Rectifying Network), our defense method can still detect and fully defend against adversarial samples.

The rest of paper is organized as follows: Section 2 contains the literature review of existing attack methods and defense methods. The realization of our method is described in Section 3. In Section 4, we validate our method by extensive experiments. Section 5 concludes the paper and provides future work of this ongoing research.

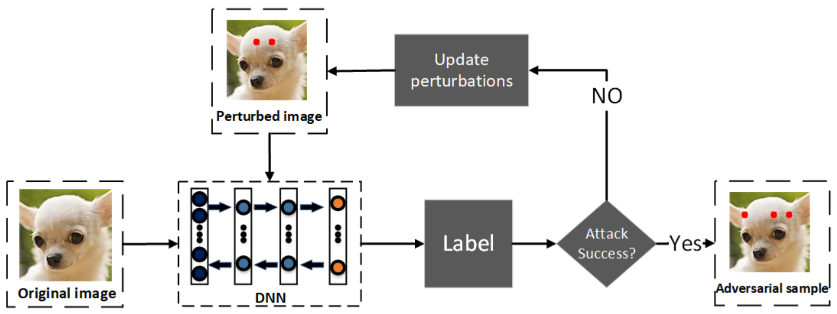

Figure 2: Illustration of adversarial sample generated using for fooling special DNN. This is an iterative strategy, that is, to generate an adversarial sample, adversary need to update perturbations continuously until attack success. So we are trying to destroy this process next. (For better description, we make the perturbations easy to observe.)

2 Related works

2.1 Adversarial attacks

In 2013, Szegedy et al. [Szegedy, Zaremba, Sutskever et al. (2013)] proposed the concept of "Adversarial samples" for the first time in the field of image classification, that is, by adding subtle perturbations to the original image, to change the intrinsic feature information of the image without affecting human's subjective perception of the image, and which will cause errors of image classification models. "Adversarial samples" is the main technology to attack DNNs at present. It has the characteristics of good performance on the attack, easy to operate and diverse implementation methods. It can be divided into white-box attack and black-box attack depending on the application scenario.

2.1.1 White-box attacks

White-box attacks are complete knowledge of the target DNN model, including its parameter values, architecture, training methods, even some training data. The attacker can obtain the algorithms used in machine learning and the parameters used by the algorithms. An attacker can interact with the target model in the process of generating adversarial samples.

Szegedy et al. [Szegedy, Zaremba, Sutskever et al. (2013)] tried to find the smallest loss function addition, which caused the neural network to misclassify and they transform the problem into a convex optimization process. Goodfellow et al. [Goodfellow, Shlens and Szegedy (2014)] developed a method for efficiently calculating adversarial perturbations called FGSM (Fast Gradient Sign Method), which confirms the high dimensional linearity of modern DNNs. Kurakin et al. [Kurakin, Goodfellow and Bengio (2016)] proposed a variant of FGSM's one-step target class. Carlini et al. [Carlini and Wagner (2017)] proposed three methods of adversarial attacks that limit the perturbation approximation by limiting the L_∞, L_2, and L_0 norms. Moosavi-Dezfooli et al. [Moosavi-Dezfooli, Fawzi and Frossard (2016)] generated a minimum normative adversarial perturbation by iterative computation until an error classification occurred. Methods such as FGSM, ILCM, DeepFool can only generate an adversarial sample of a single image. Subsequently, Moosavi-Dezfooli et al. [Moosavi-Dezfooli, Fawzi, Fawzi et al. (2017)] proposed the method of Universal Adversarial Perturbations, which can generate perturbations to all images, Khrulkov et al. [Khrulkov and Oseledets (2018)] also proposed a method of moving the general perturbation into a singular vector of the Jacobian matrix of the feature map of the network, which allows a relatively high fool rate to be achieved using only a small number of images.

2.1.2 Black-box attacks

Black-box attacks are the attack methods that generate adversarial samples for a target DNN model without knowing anything about the DNN model. Attackers do not know the algorithms and parameters used in machine learning, however, they can still interact with machine learning systems, such as judging the output by passing in any input.

Sarkar et al. [Sarkar, Bansal, Mahbub et al. (2017)] proposed two black-box attack algorithms, UPSET (Universal Perturbations for Steering to Exact Targets) and ANGRI (Antagonistic Network for Generating Rogue Images for targeted fooling of deep neural networks). The ability of UPSET comes from the residual gradient network, which can generate adversarial perturbations for a specific target label, so that when the perturbation is added to an image, the image can be classified into target label. Compared with the imperceptible disturbance of UPSET, ANGRI generates image-specific perturbations. They all have high fool rates on MNIST [LeCun, Boser, Denker et al. (1989)] and CIFAR datasets. The Houdini method Cisse et al. [Cisse, Adi, Neverova et al. (2017)] uses the gradient information of the differentiable loss function of the network to generate adversarial perturbations. In addition to image classification networks, the algorithm can also be used to deceive speech recognition networks (Google Voice, etc.). Baluja et al. [Baluja and Fischer (2017)] trained several feedforward neural networks to generate adversarial samples, which can be used to attack one or more target networks. One pixel attack Su et al. [Su, Vargas and Sakurai (2019)] is a black-box attack method, which can be implemented by changing only one pixel's value in the image. The core is to use the differential evolution algorithm to iteratively modify each pixel to generate sub-image, and compares them with the parent image. According to the selection criteria, the sub-image with the best attack effect will be retained.

2.2 Defenses against adversarial samples

The existing defense methods are mainly divided into Akhtar et al. [Akhtar and Mian (2018)] three main directions: 1) Modified training/input; 2) Modified networks; 3) Network add-on.

2.2.1 Modified training/output

This method uses modified training during learning or modified input during testing.

Adversarial training method aims at training data, by adding adversarial samples to the training process to build a model with better robustness, so as to achieve the purpose of defending adversarial samples. According to the different types of adversarial samples, adversarial training can be divided into general adversarial training, PGD adversarial training, and ensemble adversarial training. Generally speaking, the defense methods of adversarial training mostly add adversarial samples in the process of adversarial training, and these adversarial samples are generated by attacking the model specified by themselves. But the defense method of ensemble adversarial training separates the process of adversarial training from that of adversarial samples, which increases the diversity of adversarial samples and improves the robustness of defense. However, Moosavi-Dezfooli et al. [Moosavi-Dezfooli, Fawzi, Fawzi et al. (2017)] points out that no matter how many adversarial samples are added, there are new adversarial samples that can fool the network again.

The main idea of modifying the input method is trying to reduce the possible perturbation in test samples by different transformation methods, and then input the transformed samples directly into the original model for prediction. Dziugaite et al. [Dziugaite, Ghahramani and Roy (2016)] and others use JPG image compression method to reduce the impact of FGSM countermeasure disturbance on accuracy. Experiments show that this method is effective for some adversarial algorithms, but usually only using compression method is not enough, and the compression of images will also reduce the accuracy of normal classification, while small compression cannot remove adversarial perturbations. Luo et al. [Luo, Boix, Roig et al. (2015)] proposed that the foveation mechanism can be used to defend against adversarial samples generated by L-BFGS and FGSM. The hypothesis is that training a large number of data sets based on DNN classifier is robust to image scaling and transformation changes, where the confrontation mode does not have this feature. Pixel Defense Song et al. [Song, Kim, Nowozin et al. (2017)] uses the generation model to convert the confrontation samples into the normal sample space, and then converts the transformed samples into the original model for prediction.

2.2.2 Modified networks

This method mainly implements by modifying networks, e.g., by adding more layers, changing loss/activation function etc.

Since most attack algorithms use the gradient information of the model to generate the adversarial samples, the gradient masking defense method can hide the gradient information in the model training, making it difficult for the attack algorithm to attack the model through the gradient solution method, thus leading to the invalidation of the attack method. Nguyen et al. [Nguyen, Wang and Sinha (2018)] introduced a mask-based defense against C&W attacks [Carlini and Wagner (2017)] by adding noise to the logic output of the network. Gu et al. [Gu and Rigazio (2014)] introduced Deep Contractive Networks, which uses smoothness penalty terms similar to Contractive Auto Encoders, so that the output change of the model is less sensitive to the input, thus achieving the purpose of hiding gradient information.

The distillation defense Papernot et al. [Papernot, McDaniel, Wu et al. (2016)] method migrates knowledge of complex networks to simple networks. This knowledge is extracted in the form of a class probability vector of training data and fed back to the training original model, which can withstand small-scale perturbations against the attack.

DeepCloak Inci et al. [Inci, Eisenbarth and Sunar (2018)] adds a layer of mask specifically designed to adversarial sample training in front of the classification layer (usually the output layer). The added layer is explicitly trained by passing clean and adversarial image pairs forward, which encodes the difference between the output characteristics of the previous layers for these image pairs. The theory behind it is that the most important weights in the added layer correspond to the most sensitive characteristics of the network (in terms of combat manipulation). Therefore, when classifying, these features are forced to change the dominant weight of the added layer to zero.

2.2.3 Network add-on

This method mainly implements by using external models when classifying unseen examples.

Akhtar et al. [Akhtar, Liu and Mian (2018)] proposed a defense framework, which attaches additional pre-input layers to the target network and trains them to correct the adversarial samples. The separation detector is trained by extracting the characteristics of input and output differences of training images. A single training network is added to the original model to achieve the method that does not need to adjust the coefficients and immune adversarial samples. Lee et al. [Lee, Han and Lee (2017)] trained a robust network against FGSM [Goodfellow, Shlens and Szegedy (2014)] attacks using the popular framework of GAN. During the training process, classifiers constantly try to classify clean and adversarial images correctly.

The methods mentioned above are complete defense methods. Although the expectations of complete defense methods are high, many experiments show that the actual effect is not good enough. Therefore, researchers have proposed a variety of detection methods for adversarial samples. The detection-only method is used to judge whether the samples are adversarial samples, without the need to identify the real label of the samples.

The Feature Squeezing Tuncay et al. [Tuncay, Demetriou, Ganju et al. (2018)] method uses two models to find out whether adversarial samples. Subsequent work shows that this method also has acceptable resistance to C&W attack. He et al. [He, Wei, Chen et al. (2017)] also combined feature compression with the integration method proposed in Abbasi et al. [Abbasi and Gagné (2017)] to show that defensive power does not always increase by combining them. Meng et al. [Meng and Chen (2017)] proposed a framework for classifying input images into adversarial or clean images using one or more external detectors. During the training process, the aim of the framework is to learn various kinds of clean images. Training the image's manifold measurements to distinguish whether the image is perturbed or not. Liang et al. [Liang, Li, Su et al. (2017)] trained a model to treat all input pictures as noisy, learning how to smooth them first, and then classifying them. The image perturbation is treated as noise, and these perturbations are detected by scalar quantization and spatial smoothing filtering respectively. The separated binary classifier is trained to use the proposed feature to counter the sample detector. Gebhart et al. [Gebhart and Schrater (2017)] regard the calculation of neural network as the information flow in the graph, and propose a method for detecting adversarial perturbations using persistent homology of the induced map.

3 Methodology

3.1 Problem description

3.1.1 Deep neural networks

A DNN can be written as a function $g: X \rightarrow Y$, where X represents the input space and Y represents the output space. More detailed, for $x \in X$, we have Eq. (1), each f_i represents a layer, the last layer f_L outputs a vector of real numbers in the range [0,1] that add up to 1, which correspond to the class labels' probability distribution.

$$g(x) = f_L(f_{L-1}(...(f_1(x)))) \tag{1}$$

3.1.2 Gradient based attacks

Since most DNN classifiers build the optimization model by maximizing the prediction probability of the classifier on the training set, or by minimizing the prediction difference between the loss and the label. A natural idea is to build a perturbed sample with a perturbation η to legal sample which could change the predictor probability of the classifier or to make the loss as large as possible. Moreover, the perturbation itself is limited to that cannot be perceived by humans, or cannot cause significant damage to the legal sample. Therefore, a norm constraint is usually imposed on η (different norms have different effects).

FGSM Goodfellow et al. [Goodfellow, Shlens and Szegedy (2014)] produces perturbations by maximizing the loss function. Obviously, the perturbation occurs in the direction of the gradient is the most effective perturbation, and a reasonable assumption in FGSM is that the existing image data is stored in a discrete manner, so for a disturbance in the same direction, it will not have a greater impact on vision. Therefore, FGSM obtains the disturbance η by shifting the magnitude of ε in the gradient direction of the legal sample x. Thus, the adversarial sample $x' = x + \eta$, which is just equivalent to applying an infinite norm constraint to the perturbation.

$$\eta = \varepsilon \cdot \mathrm{sign}(\nabla_{\mathrm{x}} \mathrm{J}(\theta, \mathrm{x}, \mathrm{y})) \tag{2}$$

where $\mathrm{sign}(\nabla_{\mathrm{x}} \mathrm{J}(\theta, \mathrm{x}, \mathrm{y}))$ describes the gradient direction of the loss function at point x, and ε is the magnitude of the offset in this direction. η is the pertubration generated by FGSM.

Based on FGSM, many other methods for generating adversarial example are proposed.

I-FGSM is an iterative form of FGSM, which apply FGSM multiple times with a step size ε, and do a clip after every iterative step, see Eq. (3) for an example. Through multiple steps and smaller movements, it is possible to construct a more accurate adversarial sample, but at the same time it also increases the computational complexity of the structure and slows down the construction.

$$x_{N+1}^{adv} = Clip_{x,\varepsilon}(x_N^{adv} + \varepsilon \cdot sign(\nabla_x J(\theta, x_N^{adv}, y))) \tag{3}$$

DeepFool [Moosavi-Dezfooli, Fawzi and Frossard (2016)] is another iterative gradient based attack which solves the problem of choosing the perturbation coefficient ε in

FGSM, and achieves the attack on the general non-linear activation function by multiple linear approximations:

$$r_*(x) \cdot = \arg\min \|r\|_2 = -\frac{f(x)}{(\|w\|_2)^2} w \qquad (4)$$
$$\text{subject } to \ \ \text{sign}(f(x+r)) \neq sign(f(x))$$

Here r is the best perturbation. DeepFool method will produce less perturbations but change label more sufficiently.

Unlike the aforementioned attack methods, the perturbation generated by JSMA [Papernot, McDaniel, Jha et al. (2016)] is the gradient direction of the predicted value of the target class label, which called forward derivative, see Eq. (5) for an example.

$$\nabla F(X) = \frac{\partial F(X)}{\partial (X)} = [\frac{\partial F_i(X)}{\partial x_i}] \qquad (5)$$

It can be seen that the forward derivative is composed of the partial derivative value of the target class for each pixel. Therefore, the authors adopt a heuristic selection method called "saliency map", by checking the perturbation the output of each pixel, to select the most suitable pixel to make the change. That is, the perturbed pixels selected by each iteration can increase the output value of the target class as much as possible, and have a negative effect on the rest of the class labels.

$$S(X,t)[i] = \begin{cases} 0 \ \text{if} \ \frac{\partial F_t(X)}{\partial X_i} < 0, \ \text{or} \sum_{\text{j}\neq\text{t}} \frac{\partial F_j(X)}{\partial X_i} > 0 \\ \frac{\partial F_t(X)}{\partial X_i} \left| \sum_{\text{j}\neq\text{t}} \frac{\partial F_j(X)}{\partial X_i} \right|, \ \text{otherwise} \end{cases} \qquad (6)$$

In fact, it is difficult to find the pixel satisfying the above conditions. Therefore, the authors proposed to find a set of pixel pairs instead of selecting a single pixel.

$$\arg\max_{(p1,p2)} \left(\sum_{i=p1,p2} \frac{\partial F_t(X)}{\partial X_i} \right) \times \left| \sum_{i=p1,p2} \sum_{j\neq t} \frac{\partial F_j(X)}{\partial X_i} \right| \qquad (7)$$

3.1.3 Evolutionary algorithms based attack

Another widely used method to generate adversarial example is Evolutionary Algorithms. Evolutionary Computing is a mature global optimization method with high robustness and wide applicability. Differential evolution (DE) algorithm is an optimization algorithm. It guides the direction of optimization search through the group intelligence generated by the mutual cooperation and competition among individuals within the group. The basic idea of the algorithm is, during each iteration, generate a new individual by summing the vector difference between any two individuals in the population and the third individual from the randomly generated initial population, see Eq. (8) for an example.

$$v_i(g+1) = x_{r1}(g) + F \cdot (x_{r2}(g) - x_{r3}(g)) \qquad (8)$$
$$\text{subject to } \ i \neq r_1 \neq r_2 \neq r_3$$

Then generate the new individual and the corresponding individual in the contemporary population.

$$u_i(g+1)=\begin{cases} v_i(g+1), \text{if } \mathrm{i}=\mathrm{i}_{\text{rand}} \\ x_i(g), \text{otherwise} \end{cases} \tag{9}$$

In comparison, if the fitness of the new individual is better than the fitness of the current individual, the old individual is replaced by the new individual in the next generation, otherwise, the old individual is still preserved. Through continuous evolution, retaining good individuals, eliminating inferior individuals, and guiding search to the optimal solution.

$$x_i(g+1)=\begin{cases} u_i(g+1), & \text{if } \mathrm{f}(u_i(g+1)) \le f(x_i(g)) \\ x_i(g), & \text{otherwise} \end{cases} \tag{10}$$

For using DE to generate adversarial images, we can formulate it as Eq. (11).

$$\begin{aligned} &\max \underset{e(x)}{mize} \ \mathrm{f}_{\text{adv}}(x+e(x)) \\ &subject \text{ to } \|\mathrm{e(X)}\|_0 \le d \end{aligned} \tag{11}$$

where f is the target image classifier, $e(X)=(e_1,...,e_n)$ is an additive adversarial perturbation according to X , and adv means the target class. Finally, d is the limitation of maximum modification.

For One-pixel-attack, d will be set as 1, and we need to update $e(x)$ iteratively to validate whether the attack is success or not.

3.2 Perceptual hash

In Section 3.1, we can find that whether it is a gradient-based attack or an evolution-based attack, in the process of generating an effective adversarial sample, it adopts an iterative verification method, that is, repeatedly adding the perturbations to original image, and judging whether the attack is successful or not according to the output of the target DNN model. Therefore, our idea is to destroy the iteration process, so that the attack algorithm cannot obtain the prior knowledge from the previous image to update the generation process of adversarial perturbations, therefore the attack falls into a blind state. By using this method, we can ensure DNN classifiers' prediction of the perturbed sample and the original sample are coincident, which leads to the failure of the attack algorithms and the success of defending DNN-based models. Combining with the previous work of our research group, we find that perceptual hash is a suitable choice.

Perceptual hash is a method of generating digital fingerprints for multimedia data, Fig. 3 is an example. In the field of images, its function is to generate a "fingerprint" string for each image and then compare the fingerprints of different images. The closer the result is, the more similar the images are. It is mainly used for similar image searching or target tracking. An image is a two-dimensional signal that contains components of different frequencies. The area with small brightness change is the low-frequency component, which describes a wide range of frame information. The areas with intense brightness changes (such as the edges of objects) are high-frequency components, which describe the specific details of the image. Calculating the perceptual hash process of an image is a process of extracting its low-frequency information. Moreover, it uses a Discrete Cosine Transform (DCT) to obtain the low-frequency components of the image. DCT is an image compression algorithm that transforms an image from the pixel domain to the frequency domain, redundancy and correlation information is removed from the image. Hamming distance is commonly used in the perceptual hash comparison, which represents the number of two words (same length) corresponding to different bits, after counting XOR operation result, we can obtain the Hamming distance.

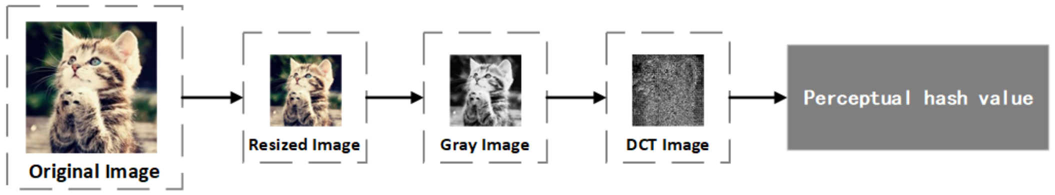

Figure 3: Illustration of perceptual hash generation process. In the image composed of DCT coefficients, the upper left corner of each small block is a DC coefficient, and the other parts of the small block are AC coefficients. There is not only brightness information around the object (cat), but also texture information, so the values of DC and AC are very large

3.3 Method and settings

Fig. 4 is an illustration of our method. It can be divided into judgment layer and time layer.

3.3.1 Judgment layer

Before the input layer of the DNN-based model, we introduce the concept of judgment layer. The main function of the judgment layer is to determine whether two images are similar or not, If the judgment is similar, the previous image's label will output directly, otherwise, the latter image will be inputted to DNN normally. Here we mainly use two methods to judge similarity: 1) perceptual hash; 2) comparison of spatial pixel values.

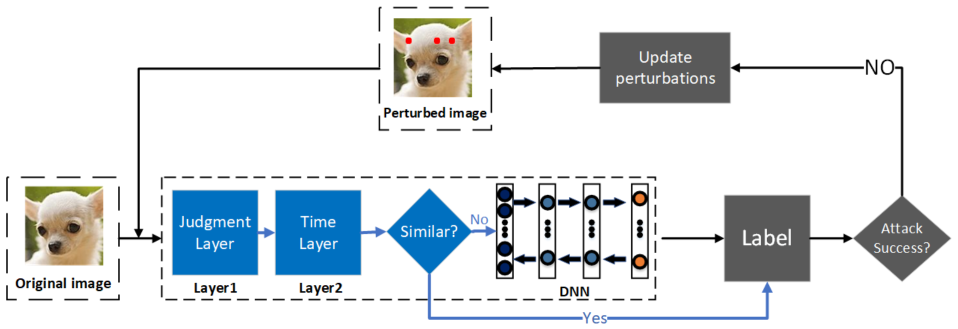

Figure 4: Illustration of our proposed defense method. The main idea (described using blue color) is to destroy the process of perturbations' generation, so that the clean image and the perturbed image have the same label. (For better description, we make the perturbations easy to observe.)

In perceptual hash, firstly, the high-frequency and detail information is removed by size scaling. then convert the image into a grayscale image and calculate the DCT coefficient. According to the DCT coefficient, the average value is calculated in the 16*16 region of the lowest frequency. Then, the value in the matrix is compared with the average value repeatedly. If the value is greater than the average, the bit is set to ‘1’, and if it is smaller, it is set to ‘0’, thus, we can get perceptual hash values. Hamming distance between two pictures can be calculated after the perceptual hash value of two pictures are obtained, which is recorded as N_{hash} .

As for the comparison of spatial pixel values, we cycle to compare the corresponding spatial pixels of the two images and record the different spatial pixels as N_{pixel}. In the aspect of similarity determination, the threshold $N_{threshold}$ is set up by comparing N_{hash} and N_{pixel}. Obviously, there are three cases for N_{hash} and three cases for N_{pixel} compared with the $N_{threshold}$, so there are nine cases in total. Here, for security, we only let the latter image enter the network when N_{hash} and N_{pixel} are both larger than the $N_{threshold}$.

We set up an area to store the RGB values of the previous image (Image 1), and its corresponding label, while the latter image (Image 2) is the same. In the initialization operation, the RGB values and the labels of Image 1 and Image 2 are all None. When the previous image enters the judgment layer, since it must be different from the None value, the label is output by the network normally. At this time, the RGB values of Image 1 are replaced with the previous image, and the label is the previous image’s label. Subsequently, the latter picture is input to the judgment layer, the Image 2 value is updated, and the label is waiting to be output. Then, there will be two cases, described in Algorithm 1 and Fig. 5.

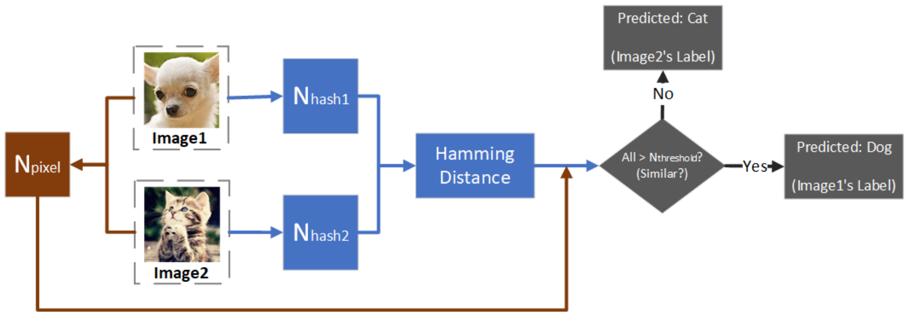

Figure 5: Illustration of the judgment layer. The process of calculating perceptual hash is described using blue color, while the process of calculating spatial pixel values is described using dark red color

1) **The latter image is a perturbed sample**-In this case, the judgment layer calculates N_{hash} and N_{pixel} according to the values of Image 1 and Image 2. Since the change is small, the judgment layer will determine that the two images are similar, thus outputting the label of Image 1 as the output of Image 2 and updating the value of Image 1.

2) **The latter image is a normal sample**-In this case, since both images' N_{hash} and N_{pixel} are larger than the $N_{threshold}$, the judgment layer determines that the two images are not similar, then Image 2 is input to the network and output normally, that its normal label will be obtained, then simultaneously updates the value and the label of Image 1 as Image 2.

Algorithm 1 The Proposed Defense Method

INPUT: Image 1, Image 2-Image 1 is the previous image which sends to DNN-based image classifiers, while Image 2 is the latter image.

OUTPUT: Y_{true}-the true label of the unknown image Image 2.

```
1:  $N_{hash1}$ ← calc_perceptualHash(Image 1)
2:  $N_{hash2}$ ← calc_perceptualHash(Image 2)
3:  $N_{pixel}$ ← calc_pixleDiff(Image 1, Image 1)
4:  $Label_{Image\ 1}$ ← DNN_predictLabel(Image 1)
5:  if HammingDistance($N_{hash1}$,$N_{hash2}$)>$N_{threshold}$ and $N_{pixel}$>$N_{threshold}$ then
6:      $Y_{true}$ ← DNN_predictLabel(Image 1)
7:      Image 1 ← Image 1
8:      $Label_{Image\ 1}$ ← $Y_{true}$
9:  else # two images are similar
10:     $Y_{true}$ ← $Label_{Image\ 1}$
11:     Image 1 ← Image 1
12:     waitTime=True
13: end if
14: return $Y_{true}$
```

3.3.2 Time layer

After the judgment layer, a time layer will be connected. The main function of the time layer is to make it impossible for the opponent to analyze the defense method according to time. According to the previous analysis, it is known that direct output image label will be much faster than normal. In order to reduce the possibility of being analyzed and to confuse the opponent more, the concept of time layer is proposed. Specifically, according to the average time of normal training as a standard, when two images are similar, the time layer will wait for a period of time according to the difference until the time is correct, and then return the label to users. During the experiment, we found that the proposed defense method only takes 0.016 seconds on average compared with the non-defense method. Note that this is a value independent of the sample size of the image, so its time complexity is O(1). On the other hand, compared with the time required to predict an image (e.g., Densenet needs 1.43 seconds), the defense time is insignificant.

4 Evaluation and results

To verify the performance of the proposed method, we reproduce a white-box attack (JSMA) and a black-box attack (One Pixel Attack) respectively as the defense targets. As mentioned before, JSMA is a gradient-based attack which generates perturbations by calculating the gradient of the DNN while One Pixel Attack is an evolutionary-based attack that the attack cannot obtain any information about our model, including the gradients.

4.1 Experiment details

Dataset. We used the CIFAR-10 dataset and loaded the dataset using the Paddle framework and the Keras framework. The dataset contains 60,000 RGB color images, which are 32*32 and divided into 10 categories include: airplane, car, bird, cat, deer, dog, frog, horse, boat, truck.

Models. For JSMA, we used the CIFAR-10 dataset to train 3 DNN models as designated image classifiers to be attacked. For One Pixel Attack, the number of DNN models are set to 6. In the process of model building, we restore the initialization process of parameters in the original model as much as possible to maintain the accuracy of the model.

Equipment. Our experiments were carried out on Intel (R) Core (TM) i7-7700 with 3.6 GHz CPU, GTX1080 Ti with 11 G GPU and 32 G memory using Ubuntu 18.04 as the operating system.

Setup. In general, when calculating the perceptual hash, the image will be cropped to 32*32, but as mentioned in Section 4.1, the image size of CIFAR-10 is 32*32, so we adjust the size of the image that calculates the perceptual hash to 16*16. On the other hand, for One Pixel Attack, we can set the threshold to be very low, such as 1%, but in the actual experiment, this threshold will not have a good effect on other attacks, in order to improve the robustness of the DNN model, we set the threshold to 10% of the total pixels of the cropped image, so that we can defend against a variety of attacks on DNN-based image classifiers.

4.2 Results of JSMA

First, we reproduce JSMA using AdvBox [Dou, Wang and Hao (2019)] and randomly selected 500 nature images in the CIFAR-10 test dataset to attack 3 different DNN models without adding our proposed defense. Then we use JSMA and the same 500 images to attack the same DNN models added with "judgment layer" and "time layer" to verify whether the defense is effective.

In order to ensure the accuracy, we use different attack parameter θ, which represents the amount of the selected features that to perturb. Generally, different θ will cause different attack-success-rate, as shown in Fig. 6, there is a positive correlation between them.

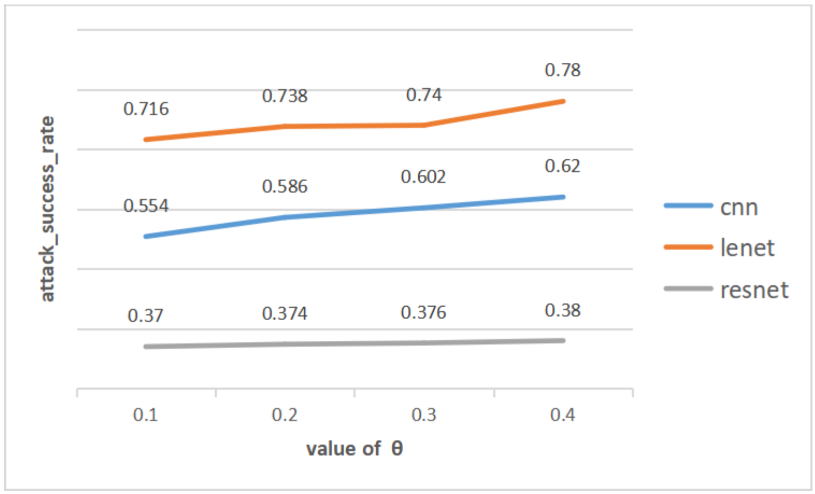

Figure 6: Illustration of the attack-success-rate by changing the attack parameter. The blue "cnn" represents a DNN model which has two convolutional layers and two fully connected layers that we trained. The results of LeNet and ResNet are shown in orange and gray respectively

As shown in Fig. 7, the attack-success-rate of all the DNN models we tested has decreased significantly. Specifically, the average attack-success-rate for cnn dropped from 59% to 37%, while LeNet dropped from 74% to 50%, ResNet dropped from 38% to 23%. The results show that our method is effective. More details are shown in Tab. 1.

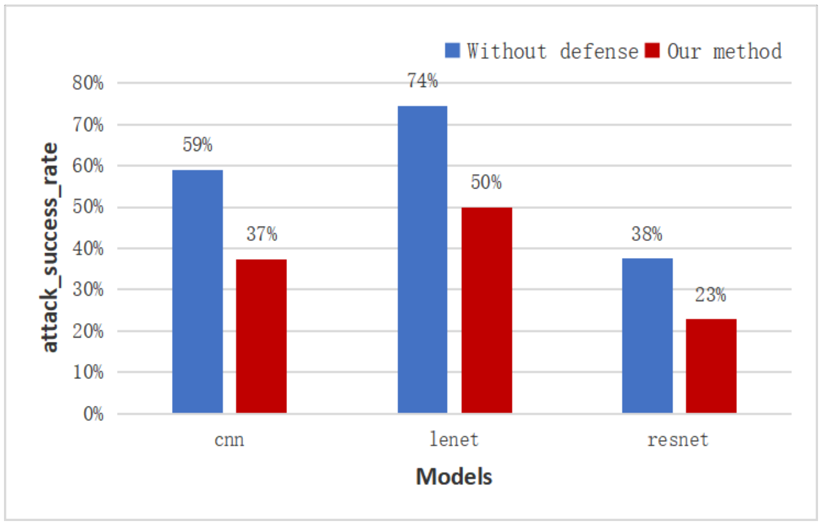

Figure 7: Results of using the proposed method to defend against JSMA. The original average attack-success-rate for DNN models are expressed in blue, while the red color shows average attack-success-rate after using our defense method

Table 1: Detailed results of JSMA

	Model	**Value of θ**	**Attack-success-rate (without defense)**	**Attack-success-rate (Our method)**
0	cnn	0.1	0.554	**0.374**
1	cnn	0.2	0.586	**0.368**
2	cnn	0.3	0.602	**0.374**
3	cnn	0.4	0.62	**0.376**
4	lenet	0.1	0.716	**0.498**
5	lenet	0.2	0.738	**0.49**
6	lenet	0.3	0.74	**0.504**
7	lenet	0.4	0.78	**0.502**
8	resnet	0.1	0.37	**0.232**
9	resnet	0.2	0.374	**0.224**
10	resnet	0.3	0.376	**0.228**
11	resnet	0.4	0.38	**0.228**

4.3 Results of one pixel attack

As for One Pixel Attack, we reproduce the method and attack 6 different DNN models without adding our proposed defenses. The attack methods include targeted and non-targeted attack. For targeted attack, if the predicted label is consistent with the specified label, the attack is successful, and we also randomly selected 500 nature images for testing in the CIFAR-10 test dataset. For non-targeted attack, when any the predicted label is inconsistent with the true label, the attack is successful, and we randomly selected 100 original images trying to perform all labels (excluding real label). Then we use One Pixel Attack to attack the same DNN models added with "judgment layer" and "time layer" to verify whether it is as effective as the white-box attack.

4.3.1 Targeted attack

As shown in Fig. 8, the attack-success-rate of all 6 different DNN models we tested has decreased significantly. The model with the maximum decline was DenseNet, dropped from 19% to 2% (as high as 88%), while LeNet has the minimum decline (65%), dropped from 43% to 15%. Specifically, the average attack success rate dropped from 24% to 7% after using our defense methods. More details are shown in Tab. 2.

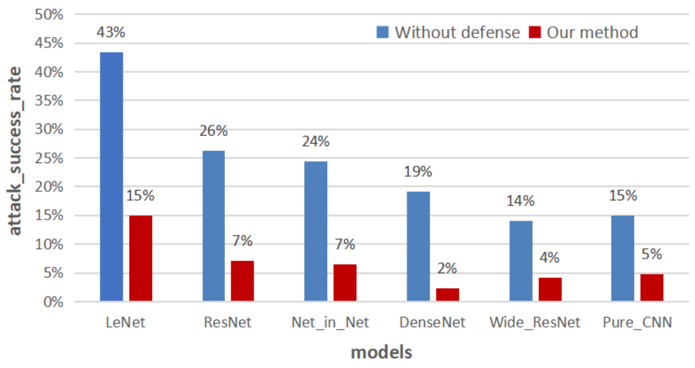

Figure 8: Results of using the proposed method to defend against One Pixel Attack's targeted attack. The original average attack-success-rate for DNN models are expressed in blue, while the red color shows average attack-success-rate after using our defense method

Table 2: Detailed results of One Pixel Attack's targeted attack

	Model	Number of pixels to perturb	Attack-success-rate (without defense)	Attack-success-rate (Our method)
0	lenet	1	0.15	**0.12**
1	lenet	3	0.59	**0.15**
2	lenet	5	0.56	**0.18**
3	pure_cnn	1	0.07	**0.03**
4	pure_cnn	3	0.17	**0.05**
5	pure_cnn	5	0.21	**0.06**
6	net_in_net	1	0.03	**0.04**
7	net_in_net	3	0.32	**0.06**
8	net_in_net	5	0.39	**0.10**
9	resnet	1	0.12	**0.04**
10	resnet	3	0.33	**0.07**
11	resnet	5	0.34	**0.10**
12	densenet	1	0.03	**0.00**
13	densenet	3	0.24	**0.04**
14	densenet	5	0.30	**0.04**
15	wide_resnet	1	0.01	**0.03**
16	wide_resnet	3	0.19	**0.04**
17	wide_resnet	5	0.22	**0.06**

4.3.2 Non-targeted attack

As shown in Fig. 9, our method to defend against One Pixel Attack's non-targeted attack is as effective as targeted attack. The DNN model with the maximum decline was DenseNet, dropped from 57% to 9% (as high as 84%), while LeNet has the minimum decline (61%), dropped from 83% to 32%. Specifically, the average non-targeted attack-success-rate dropped from 59% to 14% after using our defense methods. Notice that whether it is a targeted attack or a non-targeted attack, the best and worst DNN models are DenseNet and LeNet respectively. Through experiments, we found that DenseNet and LeNet represent the highest and lowest accuracy of the 6 DNN models. So we infer that for One Pixel Attack, the effectiveness of our proposed defense method is relevant to the accuracy of the model. More details are shown in Tab. 3.

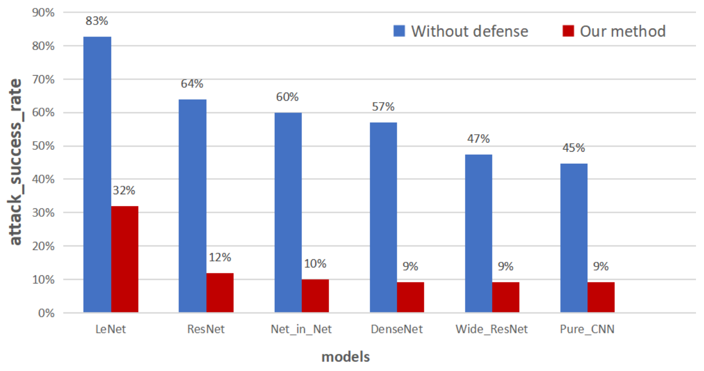

Figure 9: Results of using the proposed method to defend against One Pixel Attack's non-targeted attack. The original average attack-success-rate for DNN models are expressed in blue, while the red color shows average attack-success-rate after using our defense method

Table 3: Detailed results of One Pixel Attack's non-targeted attack

	Model	Number of pixels to perturb	Attack-success-rate (without defense)	Attack-success-rate (Our method)
0	lenet	1	0.63	**0.212**
1	lenet	3	0.92	**0.328**
2	lenet	5	0.93	**0.416**
3	pure_cnn	1	0.13	**0.064**
4	pure_cnn	3	0.58	**0.092**
5	pure_cnn	5	0.63	**0.116**
6	net_in_net	1	0.34	**0.066**
7	net_in_net	3	0.73	**0.102**
8	net_in_net	5	0.73	**0.134**
9	resnet	1	0.34	**0.072**
10	resnet	3	0.79	**0.124**
11	resnet	5	0.79	**0.16**
12	densenet	1	0.31	**0.052**
13	densenet	3	0.71	**0.092**
14	densenet	5	0.69	**0.132**

15	wide_resnet	1	0.19	**0.052**
16	wide_resnet	3	0.58	**0.092**
17	wide_resnet	5	0.65	**0.132**

5 Conclusion and future work

In this paper, we proposed a method to defend against adversarial samples based on perceptual hashing. The main idea is to destroy the process of iterative perturbation generation used by some attack methods, that we can ensure the label outputted by DNN is correct whether it's a perturbed image or a clean image, so as to achieve the purpose of defense. We provide experimental evidence not only on a white-box attack, but also a black-box attack. The results show that for JSMA, in the best case, our method can reduce the attack-success-rate by 40% of 3 different DNN models. For One Pixel Attack with 6 mainstream DNN-based models, the best case can reduce the attack-success-rate of targeted attack by 100% and that of non-targeted attack by 90%. Compared with the existing defense methods, the proposed defense method greatly reduces the computational consumption and does not need to add any other auxiliary networks, so it has the characteristics of effective, flexible and inexpensive. Therefore, our work provides a new way for defending against adversarial samples and can be used as a complement to other defense methods.

The reason why there are adversarial samples is that the training dataset cannot cover all the possibilities, and even only cover a small part, so it is impossible to train a DNN model covering all the features of the samples. On the other hand, when training DNN-based classification models, the goal is how to classify better, so the DNN model will maximize the distance between samples and boundaries, and expand the space of each class area. The advantage of this is that it makes classification easier, but the disadvantage is that it also includes a lot of space in each area that does not belong to this class. So it is necessary to make the perturbation magnitude of model misjudgment larger. in other words, make the model more robust.

In the future, we will make experiment with models trained on other JPEG datasets (such as ILSVRC) and other attack methods, then trying to encrypt and save the perceptual hash values of specific images, and using a more efficient hash function to retrieve the images in the judgment layer. In this way, even if the adversaries trying to bypass our defense methods illegally, it cannot attack the model successfully.

Acknowledgement: The authors would like to thank the anonymous reviewers for their insightful comments on the preliminary version of this paper. The work is supported by the National Key Research Development Program of China (2016QY01W0200), the National Natural Science Foundation of China NSFC (U1636101, U1736211, U1636219).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

Abbasi, M.; Gagné, C. (2017): Robustness to adversarial examples through an ensemble of specialists. arXiv:1702.06856.

Akhtar, N.; Liu, J.; Mian, A. (2018): Defense against universal adversarial perturbations. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3389-3398.

Akhtar, N.; Mian, A. (2018): Threat of adversarial attacks on deep learning in computer vision: a survey. *IEEE Access*, vol. 6, pp. 14410-14430.

Baluja, S.; Fischer, I. (2017): Adversarial transformation networks: learning to generate adversarial examples. arXiv:1703.09387.

Carlini, N.; Wagner, D. (2017): Towards evaluating the robustness of neural networks. *IEEE Symposium on Security and Privacy*, pp. 39-57.

Cisse, M.; Adi, Y.; Neverova, N.; Keshet, J. (2017): Houdini: fooling deep structured prediction models. arXiv:1707.05373.

Dou, G.; Wang, Y.; Hao, X. (2019): Advbox: a toolbox to generate adversarial examples that fool neural networks. https://github.com/baidu/AdvBox.

Dziugaite, G. K.; Ghahramani, Z.; Roy, D. M. (2016): A study of the effect of jpg compression on adversarial images. arXiv:1608.00853.

Gebhart, T.; Schrater, P. (2017). Adversary detection in neural networks via persistent homology. arXiv:1711.10056.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D. et al. (2014): Generative adversarial nets. *Advances in Neural Information Processing Systems*, pp. 2672-2680.

Goodfellow, I.; Shlens, J.; Szegedy, C. (2014): Explaining and harnessing adversarial examples. *Computer Science*.

Gu, S.; Rigazio, L. (2014): Towards deep neural network architectures robust to adversarial examples. arXiv:1412.5068.

He, W.; Wei, J.; Chen, X.; Carlini, N.; Song, D. (2017): Adversarial example defense: ensembles of weak defenses are not strong. *11th {USENIX} Workshop on Offensive Technologies*.

Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K. Q. (2017): Densely connected convolutional networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4700-4708.

Inci, M. S.; Eisenbarth, T.; Sunar, B. (2018): DeepCloak: adversarial crafting as a defensive measure to cloak processes. arXiv:1808.01352.

Khrulkov, V.; Oseledets, I. (2018): Art of singular vectors and universal adversarial perturbations. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8562-8570.

Krizhevsky, A.; Sutskever, I.; Hinton, G. E. (2012): Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, pp. 1097-1105.

Kurakin, A.; Goodfellow, I.; Bengio, S. (2016): Adversarial machine learning at scale. arXiv:1611.01236.

LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E. et al. (1989): Backpropagation applied to handwritten zip code recognition. *Neural Computation*, vol. 1, no. 4, pp. 541-551.

Lee, H.; Han, S.; Lee, J. (2017): Generative adversarial trainer: defense to adversarial perturbations with GAN. arXiv:1705.03387.

Liang, B.; Li, H.; Su, M.; Li, X.; Shi, W. et al. (2017): Detecting adversarial examples in deep networks with adaptive noise reduction. arXiv:1705.08378.

Lin, C. Y.; Chang, S. F. (2001): A robust image authentication method distinguishing JPEG compression from malicious manipulation. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 2, pp. 153-168.

Lu, J.; Issaranon, T.; Forsyth, D. (2017): Safetynet: detecting and rejecting adversarial examples robustly. *Proceedings of the IEEE International Conference on Computer Vision*, pp. 446-454.

Luo, Y.; Boix, X.; Roig, G.; Poggio, T.; Zhao, Q. (2015): Foveation-based mechanisms alleviate adversarial examples. arXiv:1511.06292.

Meng, D.; Chen, H. (2017): Magnet: a two-pronged defense against adversarial examples. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 135-147.

Moosavi-Dezfooli, S. M.; Fawzi, A.; Fawzi, O.; Frossard, P. (2017): Universal adversarial perturbations. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1765-1773.

Moosavi-Dezfooli, S. M.; Fawzi, A.; Frossard, P. (2016): Deepfool: a simple and accurate method to fool deep neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2574-2582.

Nguyen, L.; Wang, S.; Sinha, A. (2018): A learning and masking approach to secure learning. *International Conference on Decision and Game Theory for Security*, pp. 453-464.

Papernot, N.; McDaniel, P.; Jha, S.; Fredrikson, M.; Celik, Z. B. et al. (2016): The limitations of deep learning in adversarial settings. *IEEE European Symposium on Security and Privacy*, pp. 372-387.

Papernot, N.; McDaniel, P.; Wu, X.; Jha, S.; Swami, A. (2016): Distillation as a defense to adversarial perturbations against deep neural networks. *IEEE Symposium on Security and Privacy*, pp. 582-597.

Sarkar, S.; Bansal, A.; Mahbub, U.; Chellappa, R. (2017): UPSET and ANGRI: breaking high performance image classifiers. arXiv:1707.01159.

Simonyan, K.; Zisserman, A. (2014): Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556.

Song, Y.; Kim, T.; Nowozin, S.; Ermon, S.; Kushman, N. (2017): Pixeldefend: leveraging generative models to understand and defend against adversarial examples. arXiv:1710.10766.

Su, J.; Vargas, D. V.; Sakurai, K. (2019): One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S. et al. (2015): Going deeper with convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-9.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D. et al. (2013): Intriguing properties of neural networks. arXiv:1312.6199.

Tuncay, G. S.; Demetriou, S.; Ganju, K.; Gunter, C. (2018): Resolving the predicament of android custom permissions. *Proceedings 2018 Network and Distributed System Security Symposium.*

Xie, C.; Wang, J.; Zhang, Z.; Zhou, Y.; Xie, L. et al. (2017): Adversarial examples for semantic segmentation and object detection. *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1369-1378.