Adversarial Examples: Opportunities and Challenges

Jiliang Zhang, *Member, IEEE,* Xiaoxiong Jiang

***Abstract*—With the advent of the era of artificial intelligence (AI), deep neural networks (DNNs) have shown huge superiority over human in image recognition, speech processing, autonomous vehicles and medical diagnosis. However, recent studies indicate that DNNs are vulnerable to adversarial examples (AEs) which are designed by attackers to fool deep learning models. Different from real examples, AEs can hardly be distinguished from human eyes, but mislead the model to predict incorrect outputs and therefore threaten security critical deep-learning applications. In recent years, the generation and defense of AEs have become a research hotspot in the field of AI security. This article reviews the latest research progress of AEs. First, we introduce the concept, cause, characteristic and evaluation metrics of AEs, then give a survey on the state-of-the-art AE generation methods with the discussion of advantages and disadvantages. After that we review the existing defenses and discuss their limitations. Finally, the future research opportunities and challenges of AEs are prospected.**

***Index Terms*—Artificial intelligence (AI), Deep neural networks (DNNs), Adversarial examples (AEs).**

I. Introduction

In the era of AI, DNNs have shown great advantages in autonomous vehicles, robotics, network security, image/speech recognition and natural language processing (NLP). For example, in 2017, an intelligent robot with the superior face recognition ability, named XiaoDu developed by Baidu, defeated a representative from the team of humans strongest brain with the score of 3:2 [1]. On October 19th, 2017, the DeepMind team of Google released the AlphaGo Zero, which shocked the world. Compared with the previous AlphaGo, AlphaGo Zero relies on reinforcement learning without any priori knowledge to grow chess skills and finally beats every human competitor [2].

For AI research, the United States received huge support from the government, such as the Federal Research Fund. In October 2016, the United States issued the project of *Preparing for the Future of Artificial Intelligence* and the *National Artificial Intelligence Research and Development Strategic Plan*, which raised AI to the national strategic level and formulated ambitious blueprints [3], [4]. In 2017, China issued the *New Generation Artificial Intelligence Development Plan*, which mentioned that the scale of the AI core industries would exceed 150 billion CNY by 2020, promoting the development of related industries to enlarge their scale more than 1 trillion CNY. In the same year, AI was written into the nineteenth National Congress report, which pushed the development of AI industries to a new height and filled the gap in the top-level strategy of AI development [5].

In the early stage of AI, people pay more attention to the basic theory and application research. With the rapid development of AI, security issues have attracted great attention. For example, at the Shenzhen Hi-tech Fair on November 16, 2016, a robot named Chubby suddenly broke down, hitting the booth glass and hurting pedestrians without any instruction, which is the first robotic injury incident in the world [6]. Soon after, a crime-killing robot, Knightscope, manufactured by Silicon Valley Robotics, knocked down and injured a 16-month-old boy at the Silicon Valley shopping center [7]. At 22 o'clock, March 22, 2018, an Uber autonomous test vehicle hit the 49-year-old woman named Elaine Herzberg who was died after being sent to the hospital for invalid treatment in the suburbs of Tempe, Arizona. This is the first accidental autonomous vehicles accident in the world [8].

In AI security, adversarial example (AE) has become a new attack to AI systems. In 2014, Szegedy et al. [9] proposed the concept of adversarial example for the first time, which means that a subtle perturbation is added to the input of the neural network to produce a wrong output with high confidence. Even though different models have different architectures and training data, the same set of AEs can be used to attack all related models. AEs show a strong attack power in the field of image classification, speech recognition, malware detection and image captioning. For example, the classifier may misclassify the image as a speed limit sign of 45km/h after the image of a stop traffic sign is processed with AEs, which could lead to a serious traffic accident [10], [11]. Since speech recognition is widely used in mobile devices and embedded devices, many services and data are transcribed through speech recognition systems. This trend allows attackers to use AEs to control the device without the user's awareness, resulting in unpredictable risks. For example, playing hidden voice commands may cause smartphones to access malicious web pages and download malwares [12], [13], [14]. In malware classification, AEs limit their potential application settings. Attackers can slightly modify certain attributes of malware and retain their malicious attributes, but they are still classified as benign by the malware detection system [15], [16]. In image captioning system, an image is used as input to generate some captions describing the image which is perturbed by attackers to generate some image-independent, completely opposite or even malicious captions [17].

In recent years, many AE construction methods and defense techniques have been proposed. This survey elaborates on the related research and development status of AEs. The overall framework is shown in Fig. 1.

Manuscript received xxx; revised xx; accepted xxx. Date of publication 201x; date of current version 201x. This work is supported by the National Natural Science Foundation of China (Grant NOs. 61874042, 61602107), the National Natural Science Foundation of Hunan Province, China (Grant No. 618JJ3072), the Hu-Xiang Youth Talent Program, and the 2017 CCF-IFAA RESEARCH FUND.

J. Zhang and X. Jiang are with the College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China (e-mail: zhangjiliang@hnu.edu.cn).

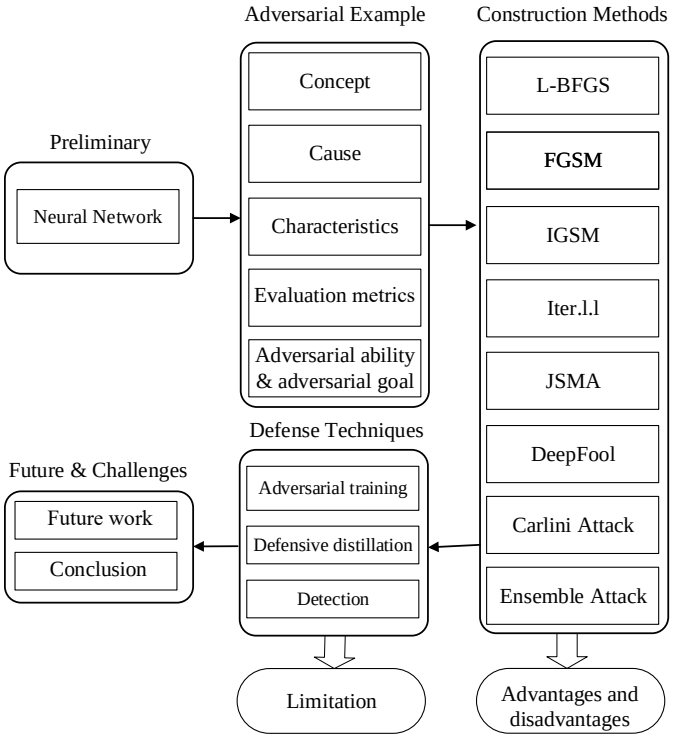

Fig. 1. The overall framework towards adversarial examples for this review.

II. Preliminaries

A. Neural Network

Artificial neural network (ANN) refers to a mathematical model that is similar to the brain synapse connection architecture and simulates the human brain's nervous system to process complex information. ANN is a kind of operational model consisting of many neurons connected to each other. Each neuron represents a specific output function called the activation function. The connection between two nodes represents the weighted value of the signal called the weight. The neural network connects many single neurons together by weights to simulate the human brain to process information.

A simple neural network consisting of an input layer L_1, a hidden layer L_2 and an output layer L_3 is shown in Fig. 2, where the circle represents the neuron of the neural network; the circle labeled "+1" represents the bias unit; the circles labeled "x_1", "x_2", "x_3" are the inputs. Neurons in different layers are connected by weights w. We use $a_i^{(l)}$ to represent the activation value (output value) of the *i-th* unit in *l-th* layer, when l = 1, $a_i^{(1)}$=x_i. With the given inputs and weights, the neural network can calculate the function output $h_{(w,b)}(x)$. The specific steps are as follows:

$$a_1^{(2)} = f(w_{11}^{(1)}x_1 + w_{12}^{(1)}x_2 + w_{13}^{(1)}x_3 + b_{11}^{(1)}) \quad (1)$$

$$a_2^{(2)} = f(w_{21}^{(1)}x_1 + w_{22}^{(1)}x_2 + w_{23}^{(1)}x_3 + b_{21}^{(1)}) \quad (2)$$

$$a_3^{(2)} = f(w_{31}^{(1)}x_1 + w_{32}^{(1)}x_2 + w_{33}^{(1)}x_3 + b_{31}^{(1)}) \quad (3)$$

$$a_1^{(3)} = h_{w,b}^{(x)} = f(w_{11}^{(2)}a_1^{(2)} + w_{12}^{(2)}a_2^{(2)} + w_{13}^{(2)}a_3^{(2)} + b_{11}^{(2)}) \quad (4)$$

The above calculation process is called forward propagation (FP), which is a transfer process of input information through the hidden layer to the output layer. The activation function ReLU: $f(x) = max\{0, x\}$ is used to make the neural network nonlinear between different hidden layers. When the machine learning task is a binary classification, the final output layer uses the activation function sigmoid: $f(x) = 1/(1 + e^{(-x)})$; when the machine learning task is a multi-class problem, the final output layer uses the activation function softmax: $f(x) = \frac{e^{x_k}}{\sum_{k=1}^{k=N} e^{x_k}}$. In the training process, the weights w and the bias b connecting the neurons in different layers are determined by back propagation (BP).

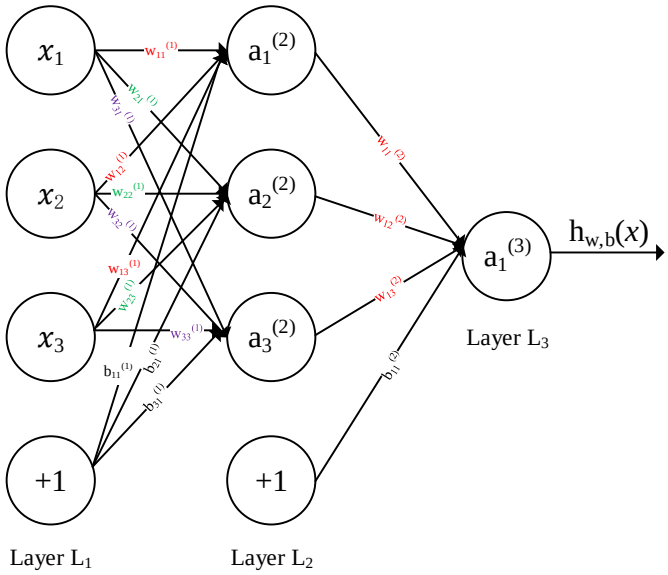

Fig. 2. A three-layer neural network model with the known inputs and weights.

Neural networks belong to a cross-disciplinary research field combining computer, probability, statistics, and brain science. It focuses on how to enable computers to simulate and implement human learning behaviors, thus achieves automatic knowledge acquisition better. However, recent studies show that neural networks are particularly vulnerable to AEs which are generated by adding small perturbations to the inputs. In what follows, we will discuss the AEs in detail.

III. Adversarial Example

In 2014, Szegedy et al. [9] first proposed the concept of adversarial example, which adds a slight perturbation to the input, resulting in the adversarial image being misclassified by the model with a high confidence, while the human eyes cannot recognize the difference. Suppose there is a machine learning model M and an original example C which can be correctly classified by the model, i.e., $M(C) = y_{true}$. However, it is possible to construct an adversarial example C' which is perceptually indistinguishable from C but is classified incorrectly, i.e., $M(C') \neq y_{true}$.

As shown in Fig. 3, the model considers the original image to be a "panda" (57.7%). After adding a slight perturbation to the original image, the original image is classified as a "gibbons" by the same model with 99.3% confidence, while

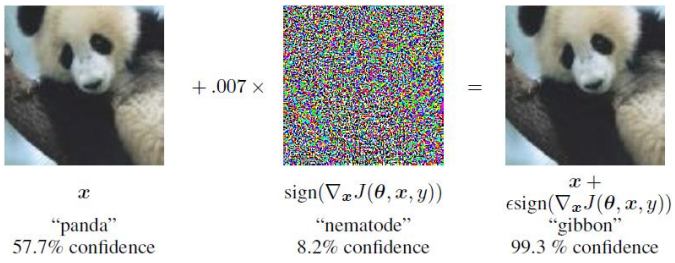

Fig. 3. Generating an adversarial example with the fast gradient sign method, .007 corresponds to a small value ϵ that restricts the norm of the perturbation, $sign(\nabla_x J(\theta, X, y_{true}))$ represents an imperceptibly perturbation [18].

the human eyes completely cannot distinguish the differences between the original image and the adversarial image [18].

In order to facilitate the reader to intuitively understand AEs, we use the neural network model in Fig. 2 as an example to show the change of the outputs by perturbing the inputs. As shown in Fig. 4, $W^{(1)}$ and $W^{(2)}$ are the weight matrices. After adding a small perturbation sign (0.5) to the original inputs, the adversarial inputs $x_1^{'}$, $x_2^{'}$, $x_3^{'}$ are equal to 1.5. Then, after going through the first layer's weight matrix $W^{(1)}$ and the activation function ReLU transform operation, the 1-st layer output $a_1^{'(2)} a_2^{'(2)} a_3^{'(2)}$ are equal to 1.5. Finally, after passing the second layer's weight matrix $W^{(2)}$ and the activation function Sigmoid transform operation, the probability of the output is changed from 0.2689 to 0.8176, which makes the model misclassify with a high confidence. With the increasing of the model depth, the probability of the output class changes more obvious.

A. Cause of Adversarial Examples

AE is a serious vulnerability in deep learning systems and cannot be ignored in security-critical AI applications. However, in the current research, there are no well-recognized explanations why the AEs can be constructed. Analyzing the causes of AEs can effectively help researchers to fix the vulnerability effectively. Some researchers [19] suspected that the reason is over-fitting or insufficient regularization of the model led to insufficient generalization ability that learning models predict unknown data. However, by adding perturbations to a regularized model, Goodfellow et al. [18] found that the effectiveness against AEs did not be improved significantly. Other researchers [20] suspected that AEs are arose from extreme nonlinearity of deep neural networks. However, by adding small perturbations to the inputs of a linear model, Goodfellow et al. found that if the inputs of a linear model had enough dimensions (in general, the dimensions of the inputs are high, because the inputs with too low dimension will result in low accuracy of the model), AEs can also be constructed successfully with high confidence.

Goodfellow et al. [18] believe that the reason for generating AEs is the linear behavior in high dimensional space. In the high dimensional linear classifier, each individual input feature is normalized. For one dimension of each input, small perturbations will not change the overall prediction of the classifier, but small perturbations to all dimensions of the inputs will lead to an effectively change of the output.

As shown in Fig. 5, in a particular direction, by adding or subtracting 0.5 to each dimension of the original example x, the score of class '1' is improved from 5% to 88%. It demonstrates that linear models are vulnerable to AEs and also refutes the hypothesis that the existence of AEs is due to the high nonlinearization of model. Therefore, the existence of the high-dimensional linear space should be the primary cause of AEs.

B. Characteristics of Adversarial Examples

In general, AEs have three characteristics:

Transferability. AEs are not limited to attack a specific neural network. It is unnecessary to obtain architecture and parameters of the model when constructing AEs, as long as the model is trained to perform the same task. AEs generated from one model M_1 can fool a different model M_2 with a similar probability. Therefore, an attacker can use AEs to attack the models that perform the same task, which means that an attacker can construct AEs in the known machine learning model and then attack related unknown models.

Regularization effect. Adversarial training [18] can reveal the defects of models and improve the robustness of examples. However, compared to other regularization methods, the cost of constructing AEs is expensive. Unless researchers can find shortcuts for constructing AEs in the future, they are more likely to use dropout [21] and weight decay (L_2 regularization).

Adversarial instability. In the physical world, it is easy to lose its adversarial for AEs after physical transformations such as translation, rotation, and lighting. In this case, AEs will be correctly classified by the model. This instability characteristic challenges attackers to construct robust AEs while creating difficulties in the deployment of AEs in the real world.

C. Evaluation Metrics

1) Success Rate: When constructing any form of AEs, the success rate is the most direct and effective evaluation criterion. Within a certain range, the success rate to generate AEs is inversely proportional to the magnitude of perturbation. For example, the fast gradient sign method [18] requires the a large perturbation and is prone to label leaking, which means that the model classifies an AE correctly when that AE is generated using the true label but misclassifies a corresponding AE generated without using the true label. Therefore, the success rate is much lower than the iterative method [22] with the lower perturbation and the Jacobian-based saliency map attack method [23] with the specific perturbation. Usually, it is difficult to construct AEs with 100% success rate.

2) Robustness: The robustness of machine learning models is related to the classification accuracy [24], [25]. Models with better performance are less vulnerable to AEs, i.e., models with higher accuracies only require less perturbations to construct AEs. Robustness is a metric to evaluate the resilience of DNNs to AEs [26]. In general, a robust DNN model has two features:

- The model has high accuracy both inside and outside of the dataset;

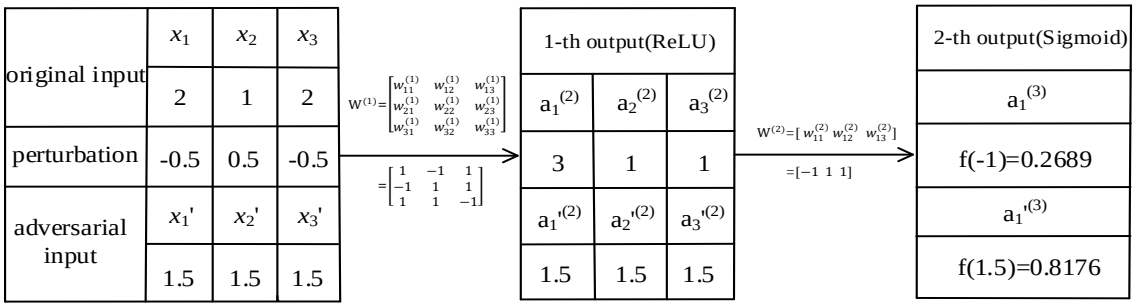

Fig. 4. The perturbation to the original inputs for a three-layer neural network (Fig. 2). The value of the original inputs x_1, x_2, x_3 and the weights $W^{(1)}, W^{(2)}$ are initialized randomly.

Original example	2	-1	3	-2	2	2	1	-4	5	1
Weights	-1	-1	1	-1	1	-1	1	1	-1	1
Adversarial example	1.5	-1.5	3.5	-2.5	2.5	1.5	1.5	-3.5	4.5	1.5

Before perturbation：
-2+1+3+2+2-2+1-4-5+1=-3
The probability of class ‘1’ ： 1/(1+exp(-(-3)))=0.0474

After perturbation：：
-1.5+1.5+3.5+2.5+2.5-1.5+1.5-3.5-4.5+1.5=2
The probability of class ‘1’ ： 1/(1+exp(-2))=0.88
The probability of class ‘1’ increased from 5% to 88%

Fig. 5. The probability of class ‘1’ before and after the perturbation.

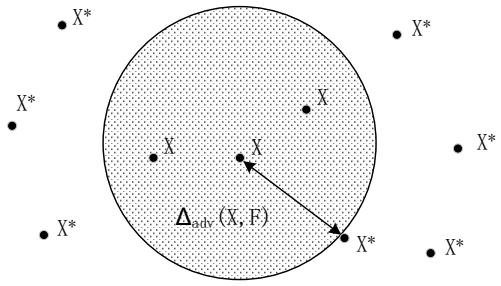

Fig. 6. **Visualizing the metric of robustness:** This 2D representation illustrates the metric as the radius of the disc at the original example *X* and going through the closest adversarial example X* constructed from *X* [26].

- The classifier of a smoothing model can classify inputs consistently near a given example.

The robustness of a DNN model *F* is defined as:

$$\rho_{adv}(F) = E_{\mu}[\Delta_{adv}(X, F)] \tag{5}$$

As shown in Fig. 6, in the circle with a radius of $\Delta_{adv}(X, F)$, the class output of the classifier is a constant original example *X*, and the output becomes an adversarial example X^* outside the circle. Therefore, the magnitude of the perturbation $\Delta_{adv}(X, F)$ is proportional to the robustness of the model, i.e., the higher the minimum perturbation value needed to misclassify the example, the stronger the robustness of the DNN is.

3) Transferability: AEs generated for one machine learning model can be used to misclassify another model, even if both models have different architectures and training data. This property is called as the transferability. The AEs can transferred among different models because a contiguous subspace with a large dimension in the adversarial space is shared among different models [27]. Transferability makes attackers only need to train the alternative model to construct AEs, and then deploys these AEs to the target model to be attacked.

The transferability of AEs can be measured by the transfer rate, i.e., the ratio of the number of transferred AEs to the total number of AEs constructed by the original model. In the non-targeted attack, the percentage of the number of AEs generated by one model are correctly classified by another model is used to measure the non-targeted transferability, we refer to this percentage as accuracy rate. A lower accuracy means better non-targeted transfer rate. In the targeted attack, the AEs generated for one model that are classified as the target label by the other model as the percentage of the targeted label to measure the targeted transferability, we refer to this percentage as matching rate. A higher matching rate means better targeted transfer rate.

The transfer rate of AEs depends on two factors. One factor is the model-related parameters, including model architecture, model capacity, and test accuracy. The transfer rate of AEs is higher among models with similar architecture, lower model capacity (the number of model parameters) and higher test accuracy [28]. Another factor is the magnitude of adversarial perturbation. Within a certain perturbation range, the transfer rate of AEs is proportional to the magnitude of adversarial perturbation, i.e., the greater perturbation to the original example, the higher transfer rate of the constructed AEs. The minimum perturbation required for different methods of constructing AEs is different.

4) Perturbation: Too small perturbations to the original examples are difficult to construct AEs while too large perturbations can be distinguished by human eyes easily. Therefore, perturbations need to achieve a balance between constructing AEs and human visual system. For example, the perturbation is difficult to control for FGSM [18] which incurs the effect of label leaking easily. To address these issues, an optimized FGSM based on iterative method [22] can control the perturbation within a threshold range, thus significantly improves the success rate of constructing AEs. However, the transfer rate of such AEs is low. Later, a saliency map-based method [23] was proposed. The key step includes: 1) *direction sensitivity estimation*: evaluate the sensitivity of each class for each input feature; 2) *perturbation selection*: use the sensitivity information to select a minimum perturbation δ among the

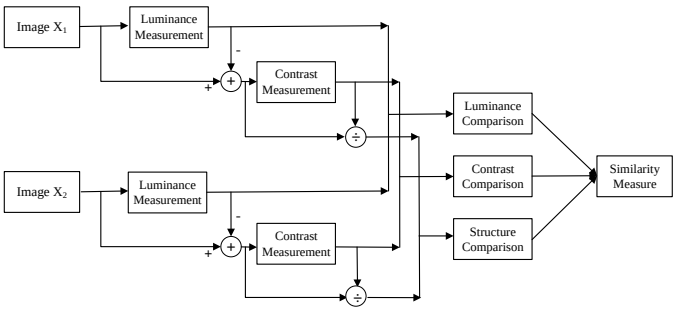

Fig. 7. The structural similarity (SSIM) measurement system [29].

input dimension which is most likely to misclassify the model.

In general, L_2 distance norm is used to measure the perturbation between the AE and the original example:

$$d(x^{'}, x) = ||x_i^{'} - x_i||_2 = \sqrt{\sum_i (x_i^{'} - x_i)^2} \tag{6}$$

where x_i is an original example, $x_i^{'}$ is an adversarial example, $d(x^{'}, x)$ represent the distance metric between x and $x^{'}$. It shows that the larger $d(x^{'}, x)$ to the original example, the greater perturbation needed to construct AEs successfully.

5) Perceptual adversarial similarity score (PASS): The AE is not only misclassified by the models, but also visually recognized as a correct class by humans. Wang et al. [29] showed that the human visual system is sensitive to structural changes, and proposed the structural similarity (SSIM) index as a metric to measure the similarity between two images. Luminance and contrast associated with the object structure are defined as the structure information of the image.

The structure of the SSIM measurement system is shown in Fig. 7. For two different aligned images X_1 and X_2, the SSIM measurement system consists of three comparisons: luminance, contrast, and structure. First, the luminance of each image is compared; second, the standard deviation (the square root of variance) is used as an estimate of the contrast of each image; third, the image is normalized by its own standard deviation as an estimate of the structure comparison; finally, the three components are combined to produce an overall similarity measure. Therefore, the perturbation of an image can be modeled as:

$$SSIM(X, Y) = \frac{1}{m} \sum_{n=1}^{m} [L(x_n, y_n)^{\alpha} C(x_n, y_n)^{\beta} S(x_n, y_n)^{\gamma}] \tag{7}$$

where m is the number of pixels; L, C, and S are the luminance, contrast, and structure of the image, respectively; hyper-parameters α, β and γ are used to weight the relative importance of L, C, and S, respectively, the default setting is $\alpha = \beta = \gamma = 1$.

Based on SSIM measurement system, Rozsa et al. [31] proposed the Perceptual Adversarial Similarity Score (PASS) to quantify human perception of AEs. The PASS between x and $x^{'}$ is defined as:

$$PASS(x^{'}, x) = SSIM(\psi(x^{'}, x), x) \tag{8}$$

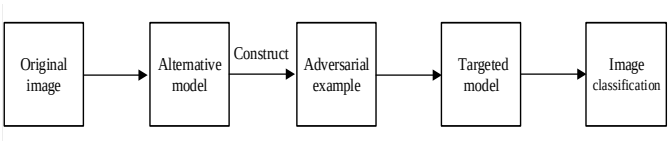

Fig. 8. **Black-box attack.** Alternative model: attackers know the model architecture and parameters; targeted model: attackers do not know the model architecture and parameters.

where $\psi(x^{'}, x)$ represents the homography transform (a mapping from one plane to another) from the original image x to the adversarial image $x^{'}$. PASS can quantify the AEs by measuring the similarity of the original image and the adversarial image. An appropriate PASS threshold can be set to distinguish the AEs with excessive perturbation. Meanwhile, attackers can also use the PASS threshold to optimize the methods of constructing AEs. Therefore, constructing an AE should satisfy:

$$argmin\ x^{'} : f(x^{'}) \neq y\ and\ PASS(x, x^{'})) \geq \theta \tag{9}$$

where $f(x^{'}) \neq y$ represents that the AE is misclassified by the model, θ is the PASS threshold set by the attacker, $PASS(x, x^{'})) \geq \theta$ represents that the AE is not recognized by the human eyes.

D. Adversarial Abilities and Adversarial Goals

Adversarial ability is determined by how much attackers understand the model. Threat models in deep learning system are classified into the following types according to the attacker's abilities.

White-box attack [31]: Attackers know the architecture and the parameters of the machine learning model, and can interact with the machine learning system in the process of generating AEs attack. However, it is difficult for attackers to obtain the architecture and the parameters of the model in practical applications, hence it is relatively rare to launch white-box attacks.

Grey-box attack [31]: Attackers know some model information such as architecture, learning rate, training data and training steps, except model parameters. This attack is a byproduct of black-box attack and is still not common in practical applications.

Black-box attack [26]: Attackers do not know the architecture and parameters of the machine learning model, but can interact with the machine learning system. For example, the output can be classified by randomly test vectors. Attackers utilize the transferability of AE to train an alternative model to construct AEs first, and then use the generated AEs to attack the unknown target model, as shown in Fig. 8.

In adversarial deep learning, the adversarial goal is to make the model misclassifies the output. According to the different influence of the perturbation on the classifier, we classify the adversarial goals into four types:

(1) **Confidence reduction:** it is to reduce the confidence of output classification.

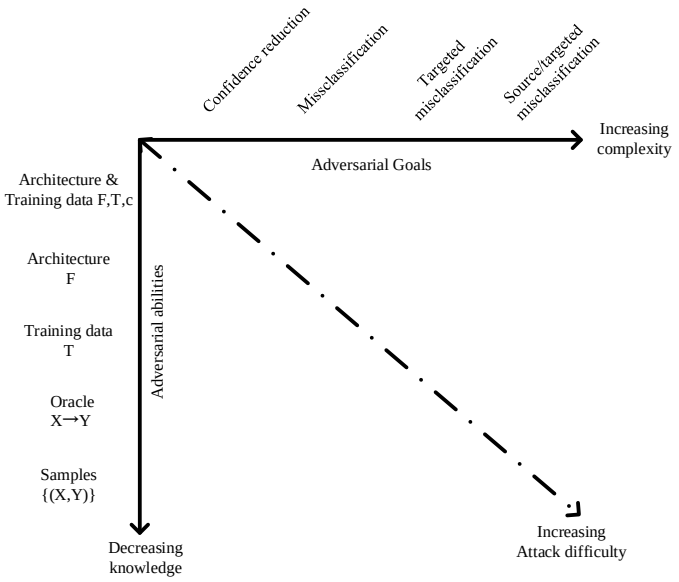

Fig. 9. Adversarial abilities and adversarial goals [23].

(2) **Non-targeted misclassification:** it is to alter the output classification to any class which is different from the original class.

(3) **Targeted misclassification:** it is to force the output classification to be the attacker-specific target class.

(4) **Source/Target misclassification:** the attacker selects a specific input to generate an attacker-specific target class.

As shown in Fig. 9, the vertical axis represents the adversarial abilities which include architecture and training data, architecture, training data, oracle (according to the relationship of change between the inputs and the outputs provided by the model to generate AEs), and examples (just know the inputs and the outputs but cannot modify these inputs to observe the difference in the outputs). The horizontal axis represents the adversarial goals, and the increasing complexity from left to right is confidence reduction, non-targeted misclassification, targeted misclassification, and source/target misclassification. In general, the weaker adversarial ability or the higher adversarial goal, the more difficult the model to be attacked.

IV. Methods of Constructing Adversarial Examples

In recent years, DNNs have become one of most popular research fields in the machine learning and have attracted much attention in academia and industry. However, recent studies show that DNNs are vulnerable to AEs which are generated by adding small perturbation to the inputs. the neural network model is misclassified while the human visual system cannot perceive the perturbation. Based on this counterintuitive phenomenon, how to construct AEs well has become a recent research focus.

A. Mainstream Attack Methods

1) *L-BFGS*

Szegedy et al. [9] proposed L-BFGS (Large; Broy-den, Fletcher, Goldforb, Shanno) to construct AEs. Given an image x, attackers construct an image $x^{'}$ similar to x with L_2 norm, and $x^{'}$ can be labeled as different class. The optimization problem is:

$$minimize \quad ||x - x^{'}||_2^2 \tag{10}$$

where $||x - x^{'}||_2^2$ is L_2 norm. The attack goal is to make $f(x^{'}) = l, x^{'} \in [0,1]^n$, where l is the target class. $f(x^{'}) = l$ is the nonlinear and non-convex function which is difficult to be solved directly. Therefore, the problem is solved approximately by the box-constrained L-BFGS.

$$minimize \quad c \cdot ||x - x^{'}||_2^2 + loss_{F,l}(x^{'}) \tag{11}$$

where c is a randomly initialized hyper-parameter which is determined by line search; $loss_{F,l}()$ is the loss function. $f(x^{'}) = l$ is approximated by minimizing the loss function. Although this method is stable and effective, the calculation is complicated.

2) *FGSM*

Goodfellow et al. [18] proposed a simplest and fastest method to construct AEs, named fast gradient sign method (FGSM). The generated images are misclassified by adding perturbation and linearizing the cost function in the gradient direction. Given an original image *X*, the problem can be solved with:

$$X^{adv} = X + \varepsilon sign(\nabla_x J(X, y_{true})) \tag{12}$$

where X^{adv} represents an adversarial example from *X*, ϵ is a randomly initialized hyper-parameter, $sign(*)$ is a sign function, y_{true} is the real label corresponding to *X*, and $J(*)$ is the cost function used to train the neural network, $\nabla_X(*)$ represents the gradient of *X*. Both Fig. 3 and Fig. 4 use this attack method to misclassify the images successfully.

There are two main differences between FGSM and L-BFGS. First, FGSM is optimized with the L_∞ norm. Secondly, FGSM is a fast AE-construction method because it does not require an iterative procedure to compute AEs, and hence it has lower computation cost than other methods. However, FGSM is prone to label leaking effect. Therefore, Kurakin et al. [33] proposed FGSM-pred which uses the predicted label y_{pred} instead of true label y_{true}. Researchers [22] also use the gradients with L_2 and L_∞ norm, i.e., $sign(\nabla_x J(X, y_{true}))$ is changed to $\frac{sign(\nabla_x J(X, y_{true}))}{||sign(\nabla_x J(X, y_{true}))||_2}$ and $\frac{sign(\nabla_x J(X, y_{true}))}{||sign(\nabla_x J(X, y_{true}))||_\infty}$, these two methods are named as FGSM grad.L_2 and FGSM grad.L_∞, respectively.

3) *IGSM*

It is difficult for FGSM to control the perturbation in constructing AEs. Goodfellow et al. [22] proposed an optimized FGSM, named iterative gradient sign method (IGSM), which applies perturbations to multiple smaller steps and clips the results after each iteration to ensure that the perturbations are within the neighborhood of the original image. For the *N-th* iteration, the update process is:

$$X_0^{adv} = X, X_{N+1}^{adv} = Clip_{X,\epsilon}\{X_N^{adv} + \varepsilon sign(\nabla_x J(X_N^{adv}, y_{true}))\} \tag{13}$$

where $Clip_{X,\epsilon}(*)$ denotes $[X - \epsilon, X + \epsilon]$.

IGSM is non-linear in the gradient direction and requires multiple iterations, but it is simpler than L-BFGS method in calculation, and the success rate of AE construction is higher than FGSM. IGSM can be further divided into two types: 1) reducing the confidence of the original prediction as the original class; 2) increasing the confidence of the prediction that originally belongs to the class with smallest probability. The following Iter l.l method is the second case.

4) *Iter l.l*

FGSM and L-BFGS try to increase the probability of predicting wrong results, but do not specify which wrong class should be selected by the model. These methods are sufficient for small datasets such as MNIST and CIFAR-10, For ImageNet with a larger number of classes and the varying degrees of significance between classes, FGSM and L-BFGS may construct uninteresting misclassifications, such as misclassifying one type of cat into another cat. To generate more meaningful AEs, a novel AE generation method is proposed by perturbing the target class with the lowest probability so that this least-likely class turns to become the correct class after the perturbation, which is called as iterative least-likely class method (iter l.l) [22]. To make the adversarial image X^{adv} be classified as y_{LL}, we have the following procedure:

$$X_0^{adv} = X, X_{N+1}^{adv} = Clip_{X,\epsilon}\{X_N^{adv} - \varepsilon sign(\nabla_x J(X_N^{adv}, y_{LL}))\} \tag{14}$$

where y_{LL} represents the least likely (the lowest probability) target class. For a classifier with good performance, the least likely class is usually quite different from the correct class. Therefore, this attack method can lead to some interesting errors, such as misclassifying a cat as an aircraft. It is also possible to use a random class as the target class. In this case, this method is called as iteration random class method (iter rnd).

5) *JSMA*

Papernot et al. [23] proposed Jacobian-based Saliency Map Attack (JSMA), which is based on the L_0 distance norm. The basic idea is to construct a saliency map with the gradients, and then model the gradients based on the impact of each pixel. The gradients are directly proportional to the probability that the image is classified as the target class, i.e., changing a larger value will significantly increase the likelihood that the model classifies the image as the target class. JSMA allows us to select the most important pixel (the maximum gradient) based on the saliency map and then perturb the pixel to increase the likelihood of labeling the image as the target class. More specifically, JSMA includes the following steps:

(1) Compute forward derivative $\nabla F(X)$.

$$\nabla F(X) = \frac{dF(X)}{dX} = [\frac{dF_j(X)}{dX_i}]_{i\in 1...M, j\in 1...N} \tag{15}$$

(2) Construct a saliency map *S* based on the forward derivative, as shown in Fig. 10.

(3) Modify the most important pixel based on the saliency map, repeat this process until the output is the target class or the maximum perturbation is got.

When the model is very sensitive to the change of the inputs, compared to other attack methods, this method can calculate the minimum perturbation more easily to misclassify and make the perturbed examples difficult to be perceived by human eyes. Although JSMA is less efficient, it is subtler, and the success rate and the transfer rate are relatively higher.

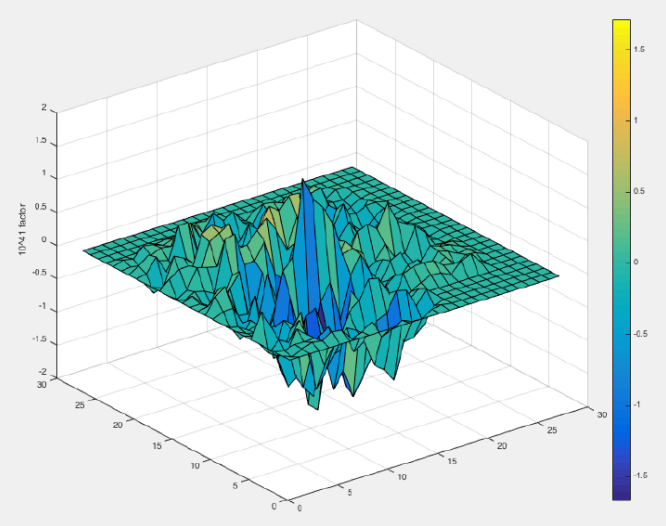

Fig. 10. Saliency map S with the 28×28 image pixel, large absolute values correspond to features with a significant impact on the output when perturbed the input [23].

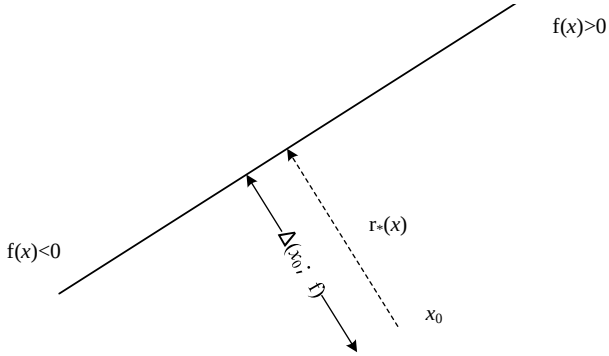

Fig. 11. Adversarial examples for a linear binary classifier [34].

6) *DeepFool*

Mohsen et al. [34] proposed a non-targeted attack method based on L_2 distance norm, called DeepFool. Assuming that the neural network is completely linear, there must be a hyperplane separating one class from another. Based on this assumption, we analyze the optimal solution to this problem and construct AEs. The corresponding optimization problem is as follows:

$$r_*(x_0) = argmin||r||_2 \tag{16}$$

subject to $sign(f(x_0 + r)) \neq sign(f(x_0))$, where *r* indicates the perturbation.

As shown in Fig. 11, x_0 is the original example, f(x) is a linear binary classifier, the straight line $wx+b = 0$ is the decision boundary, and $r_*(x)$ is the distance from the original example to the decision boundary, i.e., the distance from x_0 to the straight line $wx+b = 0$. The distance is equivalent to the perturbation $\Delta(x_0; f)$. Therefore, when $\Delta(x_0; f) > r_*(x)$, the AE can be generated.

Compared with the L-BFGS, DeepFool is more efficient and powerful. The basic idea is to find the decision boundary that is the closest to *x* in the image space, and then use the boundary to fool the classifier. It is very difficult to solve this problem directly in neural networks with high dimension and nonlinear space. Therefore, a linearized approximation is used to iteratively solve this problem. The approximation is to linearize the intermediate x_0 classifier in each iteration, and obtain an optimal update direction on the linearized model. Then x_0 is iteratively updated in this direction by a small step α, repeating the linear update process until x_0 crosses the decision boundary. Finally, AEs can be constructed with subtle perturbations.

7) *Carlini Attack (C&W)*

Carlini and Wagner [35] proposed a powerful attack method based on L-BFGS. The attack with L_0, L_2, L_∞ distance norm can be targeted or non-targeted, and we take the non-targeted L_2 norm as an example here. The corresponding optimization problem is:

$$minimize_\delta \ ||\delta||_2 + c \cdot f(x+\delta) \tag{17}$$

where $x+\delta \in [0,1]^n$, *c* is a hyper-parameter that can balance these two terms, and δ is a small perturbation. The definition of the objective function $f(x)$ is as follows:

$$f(x) = max(max\{Z(x^{'})_i : i \neq t)\} - Z(x^{'})_t, -l) \tag{18}$$

where $Z(x^{'})$ is the last hidden layer, *t* is the target label, and *l* is a hyper-parameter, which is used to control the confidence level of the model misclassification, and the AE $x^{'}$ can be classified as *t* with high confidence by adjusting the value of *l*. In general, high confidence attacks have large perturbations and high transfer rates, and the Carlini Attack based on the L_0, L_2, L_∞ distance metric can defeat defensive distillation successfully. There are three improvements to this attack based on L-BFGS:

a) Use the gradient of the actual output in the model instead of the gradient of softmax.

b) Apply different distance metrics (L_0, L_2, L_∞).

c) Apply different objective functions f(*x*).

8) *Ensemble Attack*

Liu et al. [36] proposed an ensemble attack method combining multiple models to construct AEs. If an adversarial image remains adversarial for multiple models, it is highly likely to be transferred to other models. The basic idea is to give *k* white-box models, an original image *x* and its true label *y*, the outputs are $J_1, ..., J_k$. The optimization problem based on the combination method is as follows (targeted attack):

$$argmin_{x'} - log((\sum_{i=1}^{k} \alpha_i J_i(x^{'})) \cdot 1_{y'}) + \lambda d(x, x^{'}) \tag{19}$$

where $y^{'}$ is the target label specified by the attacker, $\sum_{i=1}^{k} \alpha_i J_i(x^{'})$ is the ensemble model, and α_i is the weight of the *i-th* model, $\sum_{i=1}^{k} \alpha_i = 1$, λ is a randomly initialized parameter that is used to control the weight of the two terms. The goal of the attack method is that the generated AEs are still adversarial for the other black-box model J_{k+1}. Since the decision boundaries for different models are almost the same, the transferability of targeted AEs is improved significantly.

B. Other Attack Methods

From the perspective of attackers, the attack goal is to construct strong AEs with small perturbations and fool the model with high confidence while not being distinguished by human eyes. Recently, in addition to typical AE constructing methods introduced above, a lot of other attack methods have been proposed. Dong et al. [37] proposed a momentum iterative attack method. The basic idea is to add momentum based on the previous IGSM. The weakness of previous iterative attacks is that transferability (black-box attack) is weakened when the number of iterations increases, which can be addressed after adding momentum in iterative attacks. Momentum iterative attack not only enhances the attack ability on the white-box model, but also increases the success rate for black-box model. Xia et al. [38] proposed AdvGAN to construct AEs. The basic idea is to use generative adversarial networks to construct targeted AEs, which not only learns and preserves the distribution of the original examples, but also guarantees the diversity of perturbations and enhances generalization ability significantly. Tramr et al. [39] proposed an ensemble attack RAND+FGSM. First, they added a small random perturbation RAND to escape the non-smooth vicinity of the data point before computing the gradients. Then, they applied the FGSM to enhance the attack ability greatly. Compared with the FGSM, this method has a higher success rate and can effectively avoid label leaking. Su et al. [40] proposed an extreme one pixel attack method which only changes one pixel for each image to construct AEs. This method can attack a broader classes of DNNs without having access to any adversarial information. However, such simple perturbation can be recognized by the human eyes. Weng et al. [41] proposed an computationally feasible method called Cross Lipschitz Extreme Value for nEtwork Robustness (CLEVER), which applies extreme value theory to estimate a lower bound of the minimum adversarial perturbation required to misclassify the image. CLEVER is the first attack-independent method, and can evaluation the intrinsic robustness of neural networks.

C. Comparison among Various Attack Methods

As discussed in Section IV, a large number of methods for constructing AEs are proposed. However, these methods have their own advantages and disadvantages. Based on the evaluation indicators we discussed in Section III.C, we conducted lots of experiments and then compared the transfer rate and the success rate for different attack methods under different perturbations.

1) Model Architecture: As shown in Fig. 12, we use the dataset MNIST (50000 examples for the training set and 10000 examples for the test set) to train model *A* and model *B*, where *A* is a 5-layers neural network model including 2 convolution layers, 2 fully connected layers and 1 output layer; *B* is also a 5-layers neural network model including 3 convolutional layers, 1 fully connected layer, and 1 output layer. The regularization technique dropout is applied to prevent overfitting during the training.

TABLE I
THE SUCCESS RATE OF ADVERSARIAL EXAMPLES

Method \ Perturbation	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
L-BFGS [9]	21.4/7.9 5.4/45.6	56.9/32.7 30.4/81.3	78.4/54.3 64.9/89.5	84.2/64.6 82.7/93.1	86.9/70.3 88.5/95.6	88.3/76.1 91.2/96.7	89.4/78.0 92.1/96.9	92.8/84.6 92.3/98.2	94.2/86.4 93.2/95.1	92.4/84.6 91.9/94.5
FGSM [18]	16.7/7.7 5.0/36.8	49.0/29.2 28.7/77.9	69.7/48.8 62.3/87.3	79.0/60.1 78.1/90.8	81.9/67.3 84.4/92.6	82.3/72.2 86.5/93.3	81.6/75.7 85.7/93.6	80.3/77.0 84.1/93.3	79.2/77.4 82.8/92.5	78.4/77.2 81.4/91.3
IGSM [22]	26.5/9.2 5.8/58.1	89.6/47.4 38.8/96.4	99.7/78.2 80.7/99.6	100/91.5 96.5/99.9	100/96.6 99.3/100	100/98.5 99.8/100	100/99.4 99.9/100	100/99.6 99.9/100	100/99.8 100/100	100/99.9 100/100
JSMA [23]	22.7/8.4 6.0/52.5	58.4/34.8 32.7/86.2	85.9/61.2 68.8/91.4	88.6/67.1 84.3/94.2	90.2/72.3 88.7/96.4	91.1/78.4 92.8/97.5	92.6/80.3 92.9/97.8	94.3/87.2 93.1/98.3	95.7/88.3 94.8/96.5	94.3/86.7 93.2/95.4
DeepFool [34]	23.6/8.3 7.6/48.8	57.8/33.2 30.9/84.1	80.6/56.4 66.5/91.2	85.3/65.6 80.1/92.9	88.4/69.7 86.5/94.1	90.2/77.6 91.4/95.8	90.6/78.5 92.2/96.0	91.5/86.4 93.1/97.7	94.8/87.2 93.8/95.7	94.4/86.3 92.5/95.1
Carlini Attack [35]	**26.5/7.4** **8.6/58.0**	**98.0/42.0** **45.0/99.0**	**100/69.0** **86.0/100**	**100/85.0** **95.0/100**	**100/92.0** **98.0/100**	**100/94.0** **99.0/100**	**100/98.0** **100/100**	**100/98.0** **100/100**	**100/100** **100/100**	**100/100** **100/100**
Ensemble Attack [36]	8.2/4.5 2.7/18.4	48.8/29.6 23.2/77.7	71.8/55.6 64.3/89.0	81.1/69.0 81.9/91.5	85.6/75.6 87.1/92.5	87.4/80.1 88.4/92.8	87.5/82.7 87.9/93.1	87.4/84.0 85.9/92.8	86.5/84.1 83.4/91.8	85.3/83.9 81.1/90.5

TABLE II
THE SUCCESS RATE OF ADVERSARIAL EXAMPLES

Method \ Perturbation	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
L-BFGS [9]	35%	48%	65%	72%	80%	81%	82%	80%	80%	80%
FGSM [18]	32%	38%	42%	48%	59%	63%	45%	31%	15%	12%
IGSM [22]	30%	70%	78%	85%	88%	92%	92%	91%	90%	90%
JSMA [23]	45%	68%	75%	88%	94%	95%	98%	98%	96%	96%
DeepFool [34]	42%	45%	65%	72%	89%	89%	89%	88%	87%	88%
Carlini Attack [35]	**72%**	**90%**	**96%**	**98%**	**98%**	**99%**	**99%**	**98%**	**99%**	**98%**
Ensemble Attack [36]	56%	74%	85%	88%	92%	92%	95%	94%	94%	96%

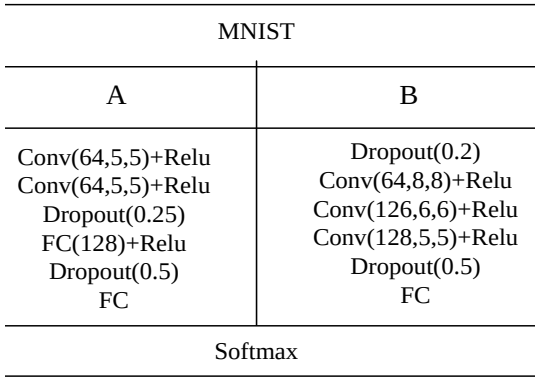

Fig. 12. Model architectures used for the MNIST dataset. Conv: convolutional layer, FC: fully connected layer.

2) The Transfer Rate of Adversarial Examples: As shown in Table I, we apply the attack methods in Section IV.A to compare the transfer rate of AEs on the models A and B in Fig. 12. Each of the four numbers in the table represents the transfer rate of $A \rightarrow A$, $A \rightarrow B$, $B \rightarrow A$, $B \rightarrow B$. Under the specific attack method, the magnitude of the perturbation is proportional to the transfer rate, i.e., the greater the perturbation, the higher the transfer rate. Under the certain perturbation, the Carlini Attack has the highest transfer rate. Under the certain perturbation and the specific attack method, the transfer rate is higher between similar or identical models. For example, the transfer rate of $A \rightarrow A$ is higher than the transfer rate of $A \rightarrow B$; the transfer rate between two models is asymmetric, such as the transfer rate of $B \rightarrow A$ and the transfer rate of $A \rightarrow B$ are different.

3) The Success Rate of Adversarial Examples: As shown in Table II, we apply the attack methods in Section IV.A to compare the success rate of AEs on the model A in Fig. 12. Under the specific attack method, the magnitude of the perturbation is proportional to the success rate, i.e., within a certain perturbation range, the greater the perturbation, the higher the success rate. Under the certain perturbation, the optimal perturbations are diversity for different attack methods. Among them, the Carlini Attack has the highest success rate in constructing AEs.

V. DEFENSES AGAINST ADVERSARIAL EXAMPLES

AEs bring a great threat to the security-critical AI applications such as face payment and autonomous vehicles based on image recognition in deep learning. Vulnerability to AEs

is not unique to deep learning, all machine learning models are vulnerable to AEs. Therefore, defending against AEs is urgent for machine learning security. In this section, we will briefly describe the basic goals of defending against AEs first, then detail the current defense techniques and their limitations, and present some suggestions for future research work on the problems of the current defense techniques finally.

A. Defense Goals

Generally, there are four defense goals:

1) **Low impact on the model architecture:** when constructing any form of defenses against AEs, the primary consideration for researchers is making a minimal modification to model architectures.
2) **Maintain model speed:** running time is very important for the availability of DNNs. It should not be affected during testing. With the deployment of defenses, DNNs should still maintain high performance on large datasets.
3) **Maintain accuracy:** defenses should have as little impact as possible on the classification accuracy of models.
4) **Defense should be targeted:** defense should work at points where AEs are relatively close to the training set. Because the examples that are very far from the dataset are relatively secure, perturbations to these examples are easily detected by the classifier.

B. Current Defenses

1) Adversarial Training

AEs have been used to improve anti-interference ability for AI models. In 2015, Goodfellow et al. [18] proposed the adversarial training to improve the robustness of the model. The basic idea is to add AEs to the training data and continuously generate new AEs at each step of the training so that the robustness of the model defense against the AEs is enhanced. The number and relative weight of AEs in each batch are controlled by the loss function independently. The corresponding loss function is as follows:

$$LOSS = \frac{1}{(m-k)+\lambda k}\Big(\sum_{i\in CLEAN} L(X_i|y_i) + \lambda \sum_{i\in ADV} L(X_i^{adv}|y_i)\Big) \tag{20}$$

where $L(X|y)$ is a loss function of the example *X* with a real label *y*, *m* is the total number of training examples, *k* is the number of AEs, and λ is a hyper-parameter used to control the relative weight of the AEs in the loss function. When *k* = 0.5*m*, i.e., when the number of AEs is the same as the number of original examples, the model has the best effect in the adversarial training.

Adversarial training is not the same as data augmentation. The data is augmented by amplifying the data and the augmented data may appear in the test set. However, AEs are usually not shown in the test set but can reveal the defects of the model. Adversarial training can be viewed as the process of minimizing classification error rates when the data is maliciously perturbed. In the following two situations, we suggested to use adversarial training:

i) **Overfitting:** when a model is overfitting, a regularization term is needed.
ii) **Security:** when AEs refer to security problems, adversarial training is the most secure method among all known defenses with only a small loss of accuracy.

Although the model is very robust to white-box attacks after adversarial training, it is still very vulnerable to the AEs generated from other models, i.e., the model is not robust to black-box attacks. Based on this attribute, Tramr et al. [39] proposed the concept of ensemble adversarial training. The main idea is to augment the training data which constructed not only from the model being trained, but also from the other pre-trained models. The benefit is that it can increase the diversity of AEs and improve the generalization ability significantly. Compared to standard adversarial training, the model is weak in defending against white-box attack in the ensemble adversarial training, but the defensive ability of the model is enhanced significantly in the black-box attack.

AEs also offer the possibility to implement semi-supervised learning. The model assigns label $y^{'}$ to point *x* which are not related to label in the dataset. When the model performs well, $y^{'}$ has large possibility of being a true label. Then the attacker constructs an AE x' that causes the classifier to output a label $y^{''}$, and $y^{'} \neq y^{''}$. The goal is to assign *x* and x' the same label. These AEs that do not use real label and generated by a trained model to provide label are called virtual adversarial examples. Based on the concept of virtual adversarial examples, Miyato et al. [42] proposed a method that adding virtual adversarial examples to the dataset for training, we refer to this method as virtual adversarial training. The virtual adversarial loss function is as follows:

$$LOSS = KL[p(\cdot|x;\widehat{\theta})||p(\cdot|x+r_{v-adv};\widehat{\theta})] \tag{21}$$

where $KL[p||q]$ represents the KL divergence between the distributions *p* and *q*, r_{v-adv} represents the adversarial perturbation. The adversarial loss function defined in the adversarial training needs the real label *y*, but the virtual adversarial loss function only needs the input *x* and does not need the real label *y*, which enables the researcher to apply virtual adversarial training in semi-supervised learning successfully. The goal of virtual adversarial training is to give the possibly same predicted value before and after adding the perturbation, which makes semi-supervised learning can use unlabeled examples to achieve significant effect in text classification and machine translation [43].

2) Defensive Distillation

Adversarial training needs AEs to train the model, so the defense is specific to the process of AEs construction. The defensive capability is divergent for different attack methods. To solve this problem, in 2016, Papernot et al. [26] proposed a method to improve the robustness for any neural networks, which is called as defensive distillation. The distillation method originally uses a small model to simulate a large and computationally intensive model, which does not affect the accuracy and can solve the problem of information

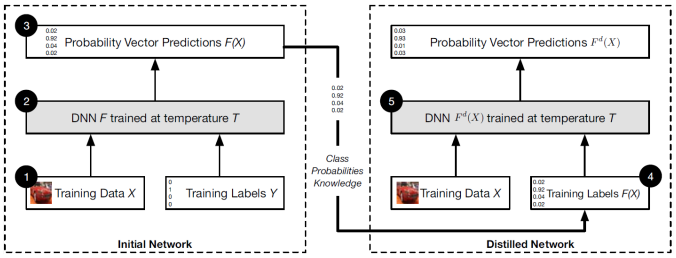

Fig. 13. **The pipeline of defensive distillation.** The initial network is trained at temperature T on the training set $(X,Y(X))$, the distilled network is trained at the same temperature T on the new training set $(X,F(X))$ [26].

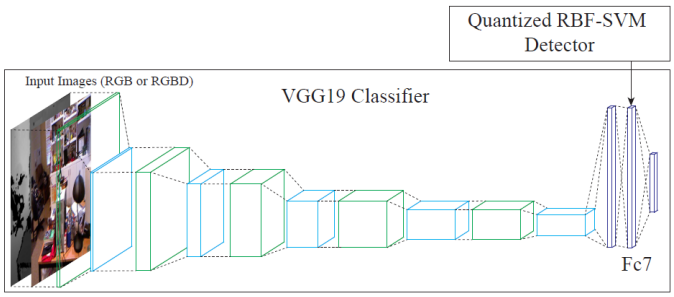

Fig. 14. **SafeNet architecture.** SafetyNet consists of a conventional classifier with an RBF-SVM that uses discrete codes to detect adversarial examples [45].

missing. Different from the traditional distillation technique, defensive distillation aims to smooth the model during the training process by generalizing examples outside the training data. Specifically, one model is trained in a normal way first, and another compression model is trained with the probability vector learned from the first model. The robustness of the second distilled model is significantly enhanced. The specific training steps are shown in Fig. 13.

i) The probability vectors produced by the first DNN are used to label the dataset. These new labels are called soft labels as opposed to hard class labels.

ii) The newly labelled dataset is used to train the second DNN model. The second model can also be trained by using a combination of the hard and soft labels. Since the second model combines the knowledge of the first model, the size of the second model is smaller in scale, less computationally expensive and more robust than the first model.

The basic idea of defensive distillation is to generate smooth classifiers that are more resilient to AEs, reducing the sensitivity of the DNN to input perturbation. In addition, it improves the generalization ability because there is no need to modify the neural network architecture, therefore, it has low training overhead and no testing overhead. Although Carlini and Wagner [44], [35] proposed an attack method to demonstrate that the defensive distillation does not improve the robustness of neural networks significantly, but it is still a good research direction to defense AEs in the following three ways :

(i) Consider defensive distillation under different types of perturbation (FGSM, L-BFGS, etc.);

(ii) Investigate the effect of distillation on other DNN models and AE constructing algorithms;

(iii) Study various distance metrics such as L_0, L_2, L_∞ between the original examples and the AEs.

3) *Detection*

Adversarial training is proposed to enhance the robustness of the model. However, this method lacks generalization ability and is pdifficult to popularize. Defensive distillation was proposed to defend AEs but is then defeated by a strong Carlini Attack scheme. In 2017, Lu et al. [45] proposed an RBF-SVM based detector to detect whether the input is normal or adversarial (as shown in Fig. 14). The detector can get the internal state of some back layers in the original classification neural network. If the detector finds out that the example is adversarial example, then it will be rejected.

We assume that the detector is difficult to attack, and the output of ReLU activation function is processed in the binary format. Since normal examples and AEs generate different binary codes, detectors can compare the code during the test to determine whether the input is normal or adversarial.

At present, AE detectors are mainly divided into the following classes [46]:

Detection Based on Secondary Classification: Generally, there are two kinds of secondary classification detection methods. The first one is adversarial training detector [47], [48], which is similar to adversarial training. The main idea is to add a new classification label to AEs during training. If an adversarial example is detected, the model will classify it into the new class. The second one is to take the characteristics extracted from AEs and original examples during the convolution layer as input. Then the labeled input data is used to train neural network detectors. This method performed well on detecting over 85% of AEs.

Detection Based on Principal Component Analysis (PCA): The essence of PCA is to transform the original features linearly and maps them to a low-dimensional space with the best possible representation of the original features. PCA-based detection methods are mainly divided into two types. The first one uses PCA in the input layer [49], because AEs have a greater weight processed by PCA than original examples. The second one uses PCA in the hidden layer [50]. If the result of each hidden layer matches the feature of original examples, the detector will classify the input as original examples.

Detection Based on Distribution: There are two main distribution-based detection methods. The first kind of distribution-based detectors uses the maximum mean discrepancy (MMD) [47]. Assuming there are two sets of images S_1 and S_2, S_1 contains all the original examples, S_2 contains either all AEs or all original examples. If S_1 and S_2 have the same distribution, then S_2 has original examples; otherwise, S_2 is full of AEs. The second one uses Kernel Density Estimation [51]. Since AEs have a different density distribution from the original examples, they can be detected with high confidence by the estimation of the density ratio. If the density ratio of one example is close to 1, it belongs to original example, while the density ratio is much larger than

1, it belongs to AEs.

Other Detection Methods: Dropout randomization [21] is a method to use dropout randomly during AE detection. Original examples always generate correct labels, but AEs are of high possibility to be different from the label corresponding to original examples. In addition, another method called Mean Blur [50] uses the filter to perform mean blurring on the input image. This method is simple and can effectively improve the robustness of models.

C. Other Defense Techniques

From the perspective of defenders, our goal is to train a model where no AEs exist in this model or AEs cannot be easily generated. Some novel researches on defending AEs have been proposed. Meng et al. [31] proposed a framework MagNet, including one or more separate detector networks and a reformer network. The detector network learns to distinguish normal examples from AEs by approximating the manifold of normal examples. The reformer network moves AEs towards the manifold of normal examples. As MagNet is independent of any AEs construction process, it is therefore effective in the black-box and gray-box attacks. Dong et al. [52] proposed a high-level representation guided denoiser (HGD) method. Even though the success rate is a bit lower through using HGD, fewer training images and less training time is consumed than previous defense methods. Ma et al. [53] proposed Local Intrinsic Dimensionality (LID) to describe the dimensional attributes of the adversarial subspace in the AEs, and it is proved that these features can distinguish normal examples from AEs effectively. The analysis of the LID characteristics of the adversarial area provides not only a new direction for defenses, but also more challenges for potential attacks. Baluja et al. [54] proposed adversarial transformation networks (ATNs) to increase the diversity of perturbations. ATNs improve the effectiveness of adversarial training based on specific AEs. However, ATNs may produce similar perturbations when using iterative methods. Note that iterative methods are not currently suitable for adversarial training.

D. Limitations of Defenses

As discussed above, a lot of defenses have been proposed. In what follows, we summarize their advantages and disadvantages.

As shown in Table III, adversarial training is simple and can significantly improve the robustness of models. However, AEs are required in the training process, which brings high overhead. Besides, it is difficult to theoretically explain which attack method to construct AEs for adversarial training can achieve the best robustness of models. Defensive distillation can greatly reduce the sensitivity to perturbation without modifying the neural network architectures. Therefore, defensive distillation incurs low overhead in training and testing. However, defensive distillation needs to add distillation temperature and modify the objective function, which increases the complexity of designing defensive models. Besides, attackers can easily bypass the defensive distillation following the three strategies: 1) choose a more suitable objective function; 2) calculate the final layer of gradient instead of the second-to-last layer of gradient; 3) attack a fragile model and then transfer to the distillation model. Detectors do not need to modify the model architecture and parameters; hence the complexity is low. However, its performance is highly correlated with the type of detector. In addition, this method only detects the existence of AEs and does not improve the robustness of the model.

VI. Future work

AE construction and defense are one of the research hotspots in the AI security field. Although many AE construction methods and defense techniques have been proposed, various unresolved problems still exist. This section summarizes the problems to this field and put forward to some future research directions.

In term of AE construction, there are three major problems:

1) **It is difficult to build a generalized AE-construction method.** In recent years, a lot of AE-construction methods have been proposed such as the gradient-based FGSM, JSMA, the classifier decision boundary-based DeepFool and the ensemble attack method combining multiple models. These methods can achieve good performance in some evaluation metrics, but they are difficult to construct a generalized AE. Therefore, defenders can propose the efficient defenses against these specific attacks. For example, the gradient can be hidden or obfuscated to prevent against the gradient-based AE-construction methods; the objective function can be modified to smooth decision boundaries to prevent against on the classifier decision boundaries-based AE-construction methods.

2) **It is difficult to control the magnitude of perturbation for target images.** In the mainstream attack methods, attackers construct AEs by perturbing target images to fool neural network models. However, it is difficult to control the magnitude of perturbation because too small perturbation can not generate AEs and too large perturbation can be perceived by human eyes easily.

3) **AEs are difficult to maintain adversarial stability in real-world applications.** The image perturbed at specific distances and angles may led to the model misclassification, but a lot of images are perturbed at different distances and angles failed to fool the classifier [55]. Moreover, AEs may lost its adversarial with physical transformation such as blurring, rotation, scaling and illumination [56]. Actually, it is hardly for AEs to maintain stability in the real-world applications.

Therefore, to address these issues, we propose to improve AE quality in the following three directions.

1) **Construct AEs with a high transfer rate.** With the diversification of neural network models, the effectiveness of attacksp for one single model is not enough. Based on the transferability, constructing AEs with high transfer rate is a prerequisite to evaluate the effectiveness of black-box attacks and a key metric to evaluate generalized attacks.

2) **Construct AEs without perturbing the target image.** When constructing AEs, the magnitude of perturbation to the target image is determined by experiments and the optimal

TABLE III
THE ADVANTAGES AND DISADVANTAGES OF DEFENSES

Defenses	Advantages	Disadvantages
Adversarial training [18]	Simple operation, good defense effect	Training is difficult to converge, high overhead
Defensive distillation [26]	Low overhead, well generalization	Only suitable for energy probability distribution, model-dependent, high complexity
Detector [45]	Low complexity, model-independent	Weak generalization, detector-dependent, not improve the robustness

perturbation will be different in various models. It increases the complexity of attacks and affects the success rate and the transfer rate. Therefore, Constructing AEs without perturbing the target image is a new and challenging research direction.

3) **Model the physical transformation.** In the physicalp world, attackers need to consider not only the magnitude of the perturbations but also the physical transformations such as translation, rotation, brightness, and contrast. However, it is difficult for attackers to use traditional algorithms to generate real-world AEs with high adversarial stability. Therefore, it is an efficient way to model physical perturbations to improve the adversarial stability of the constructed real-world AEs.

In terms of defending against AEs, there are two main issues at present.

1) **Defense is highly related to model architectures and parameters.** The black-box attack does not need to obtain model architecture and parameters to construct AEs. Therefore, it is difficult for defenders to resist the black-box attack by modifying the model architectures or parameters. For example, defensive distillation needs to modify and retrain the target classifier, which was defeat by a strong Carlini Attack method. The defensive effect of detector is related to the existence of AEs, which was broken by constructing new loss functions.

2) **Weak generalization for defense models.** Adversarial training and detector are representative defense techniques. Adversarial training can improve the robustness of the model by adding AEs to training dataset. Detector can detect examples based on AEs in the dataset. However, the robustness is different when defending the AEs generated by different attack methods, i.e., the generalization ability of defense models is weak.

VII. CONCLUSION

Deep neural networks (DNNs) have recently been achieving state-of-the-art performance on a variety of pattern-recognition tasks. However, recent researches show that DNNs, like many other machine learning models, are vulnerable to AEs. Although many AE construction and defense methods have been proposed, there are still some challenges to be solved. The state-of-the-art research is still in the adversarial development stage of "while the priest climbs a post, the devil climbs ten". In this survey, we summarize the state-of-the-art AE construction methods and the corresponding defense techniques, and then discuss several future research directions and challenges along with the future trends in this field. Although AEs have caused researchers to question deep learning, it also prompts both academia and industry to better understand the difference between AI and our human brain.

REFERENCES

[1] J. Changjian, "The Brain (game show)," Wikipedia, [Online]. Available: https:P//wikivisually.com/wiki/The_Brain_(game_show), 2017.

[2] D. S. Demis Hassabis, "AlphaGo Zero: Learning from scratch," DeepMind, [Online]. Available: https://deepmind.com/blog/alphago-zero-learning-scratch/, 2018.

[3] E. Felten, "Preparing for the Future of Artificial Intelligence," [Online]. Available: https://obamawhitehouse.archives.gov/blog/2016/05/03/preparing-future-artificial-intelligence, 2016.

[4] B. Biegel and J. F. Kurose, "The National Artificial Intelligence Research and Development Strategic Plan," White House, no. October, p. VIII-40, 2016.

[5] R. C. Graham Webster, Paul Triolo, Elsa Kania, "A Next Generation Artificial Intelligence Development Plan," China Copyright and Media, [Online]. Available: https://chinacopyrightandmedia.wordpress.com/2017/07/20/a-next-generation-artificial-intelligence-development-plan/, 2017.

[6] Shailaja Neelakantan, "Chinese robot 'Fatty' goes haywire, smashes booth, injures 1 at trade fair in Shenzhen," [Online]. Available: https://timesofindia.indiatimes.com/world/china/Chinese-robot-Fatty-goes-haywire-smashes-booth-injures-1-at-trade-fair-in-Shenzhen/articleshow/55535893.cms, 2016.

[7] G. McDonald, "Danger, danger! 10 alarming examples of AI gone wild," InfoWorld, [Online]. Available: https://www.infoworld.com/article/3184205/technology-business/danger-danger-10-alarming-examples-of-ai-gone-wild.html, 2017.

[8] F. Siddiqui, "Self-driving Uber vehicle strikes and kills pedestrian," [Online]. Available: https://www.washingtonpost.com/news/dr-gridlock/wp/2018/03/19/uber-halts-autonomous-vehicle-testing-after-a-pedestrian-is-struck/?noredirect=on&utm_term=.7bec7eab9f87, 2018.

[9] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I.Goodfellow, R. Fergus, "Intriguing properties of neural networks.". arXiv preprint arXiv:1312.6199, 2014.

[10] J. Evans, "How to trick a neural network into thinking a panda is a vulture," 2015.

[11] Evan and Brendon, "Ensembling as a Defense Against Adversarial Examples," *CS 229: Final Project*, 2016.

[12] R. Taori, A. Kamsetty, B. Chu, and N. Vemuri, "Targeted Adversarial Examples for Black Box Audio Systems,". arXiv preprint arXiv:1805.07820, 2018.

[13] N. Carlini and D. Wagner, "Audio Adversarial Examples: Targeted Attacks on Speech-to-Text,". arXiv preprint arXiv:1801.01944, 2018.

[14] D. Iter, J. Huang, and M. Jermann, "Generating Adversarial Examples for Speech Recognition,". 2017.

[15] W. Hu and Y. Tan, "Generating Adversarial Malware Examples for Black-Box Attacks Based on GAN,". arXiv preprint arXiv:1702.05983, 2017.

[16] X. Liu, Y. Lin, H. Li, and J. Zhang, "Adversarial Examples: Attacks on Machine Learning-based Malware Visualization Detection Methods,". arXiv preprint arXiv:1808.01546, 2018.

[17] H. Chen, H. Zhang, P.-Y. Chen, J. Yi, and C.-J. Hsieh, "Show-and-Fool: Crafting Adversarial Examples for Neural Image Captioning,". arXiv preprint arXiv:1712.02051, 2017.

[18] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples,". arXiv preprint arXiv:1412.6572, 2015.

[19] K. Taga, K. Kameyama and K. Toraichi, "Regularization of hidden layer unit response for neural networks," *2003 IEEE Pacific Rim Conference on Communications Computers and Signal Processing (PACRIM 2003)* (Cat. No.03CH37490), Victoria, BC, Canada, 2003, pp. 348-351 vol.1.

[20] N. J. Cotton and B. M. Wilamowski, "Compensation of Sensors Nonlinearity with Neural Networks," *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, Perth, WA, 2010, pp. 1210-1217.

[21] G. Hinton, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," vol. 15, pp. 1929-1958, 2014.

[22] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world,". arXiv preprint arXiv:1607.02533, 2016.

[23] N. Papernot, P. Mcdaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," *IEEE European Symposium on Security and Privacy*, pp. 372-387, 2016.

[24] D. Su, H. Zhang, H. Chen, J. Yi, P. Y. Chen, and Y. Gao, "Is Robustness the Cost of Accuracy? – A Comprehensive Study on the Robustness of 18 Deep Image Classification Models,". arXiv preprint arXiv:1808.01688, 2018.

[25] A. Rozsa, M. Gnther, and T. E. Boult, "Are accuracy and robustness correlated?, *IEEE International Conference on Machine Learning and Applications*, pp. 227-232, 2017.

[26] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks," *IEEE Symposium on Security and Privacy (SP)*, pp. 582-597, 2016.

[27] F. Tramr, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "The Space of Transferable Adversarial Examples". arXiv preprint arXiv:1704.03453, 2017.

[28] L. Wu, Z. Zhu, and C. Tai, "UNDERSTANDING AND ENHANCING THE TRANSFERABILITY OF ADVERSARIAL EXAMPLES,". arXiv preprint arXiv:1802.09707, 2018.

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

[30] A. Rozsa, E. M. Rudd, and T. E. Boult, "Adversarial Diversity and Hard Positive Generation," *IEEE Computer Society*, pp. 410-417, 2016.

[31] D. Meng and H. Chen, "MagNet: a Two-Pronged Defense against Adversarial Examples," *ACM Conference on Computer and Communications Security (CCS)*, 2017.

[32] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical Black-Box Attacks against Machine Learning," *the ACM on Asia Conference on Computer and Communications Security*, pp. 506-519. ACM, 2017.

[33] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial Machine Learning at Scale,". arXiv preprint arXiv:1611.01236, 2017.

[34] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: a simple and accurate method to fool deep neural networks," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2574-2582, 2016.

[35] N. Carlini and D. Wagner, "Towards Evaluating the Robustness of Neural Networks," *2017 IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, pp. 39-57, 2017.

[36] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into Transferable Adversarial Examples and Black-box Attacks,". arXiv preprint arXiv:1611.02770, 2016.

[37] Y. Dong et al., "Boosting Adversarial Attacks with Momentum,", arXiv preprint arXiv:1710.06081, 2018

[38] F. Xia and R. Liu, "Adversarial Examples Generation and Defense Based on Generative Adversarial Network,". arXiv preprint arXiv:1712.00170, 2016.

[39] F. Tramr, A. Kurakin, N. Papernot, D. Boneh, and P. McDaniel, "Ensemble Adversarial Training: Attacks and Defenses,". arXiv preprint arXiv:1705.07204, 2017.

[40] J. Su, D. V. Vargas, S. Kouichi,"One pixel attack for fooling deep neural networks". arXiv preprint arXiv:1710.08864, 2017

[41] T. W. Weng, H. Zhang, P. Y. Chen, J. Yi, D. Su, Y. Gao, C. J. Hsieh, L. Daniel, "Evaluating the robustness of neural networks: An extreme value theory approach." *International Conference on Learning Representations (ICLR)*, 2018.

[42] T. Miyato, S. Maeda, M. Koyama, and S. Ishii, "Virtual Adversarial Training: a Regularization Method for Supervised and Semi-supervised Learning," pp. 1-14, 2017.

[43] T. Miyato, A. M. Dai, and I. Goodfellow, "Adversarial Training Methods for Semi-Supervised Text Classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1-11, 2016.

[44] N. Carlini and D. Wagner, "Defensive Distillation is Not Robust to Adversarial Examples,". arXiv preprint arXiv:1607.04311, 2016.

[45] J. Lu, T. Issaranon, and D. Forsyth, "SafetyNet: Detecting and Rejecting Adversarial Examples Robustly," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp: 446-454, 2017.

[46] N. Carlini and D. Wagner, "Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods," *the 10th ACM Workshop on Artificial Intelligence and Security*, pp:3-14, 2017.

[47] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel, "On the (Statistical) Detection of Adversarial Examples,". arXiv preprint arXiv:1702.06280, 2017.

[48] Z. Gong, W. Wang, and W. Ku, "Adversarial and Clean Data Are Not Twins,". arXiv preprint arXiv:1704.04960, 2017.

[49] D. Hendrycks and K. Gimpel, "Early Methods for Detecting Adversarial Images,". arXiv preprint arXiv:1608.00530, 2016.

[50] X. Li and F. Li, "Adversarial Examples Detection in Deep Networks with Convolutional Filter Statistics," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 5764-5772, 2017.

[51] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting Adversarial Samples from Artifacts,". arXiv preprint arXiv:1703.00410, 2017.

[52] Y. Dong, "Adversarial Attacks and Defenses," [Online]. Available: https://blog.csdn.net/cf2SudS8x8F0v/article/details/78999169, 2018.

[53] X. Ma et al., "Characterizing Adversarial Subspaces Using Local Intrinsic Dimensionality,". arXiv preprint arXiv:1801.02613, 2017.

[54] S. Baluja, I. Fischer, "Adversarial Transformation Networks:Learning to Generate Adversarial Examples". arXiv preprint arXiv:1703.09387, 2017

[55] J. Lu, H. Sibai, E. Fabry, and D. Forsyth, "NO Need to Worry

about Adversarial Examples in Object Detection in Autonomous Vehicles,". arXiv preprint arXiv:1707.03501, 2017.

[56] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing Robust Adversarial Examples,". arXiv preprint arXiv:1707.07397, 2017.