# A Review of Adversarial Attack and Defense for Classification Methods

Yao Li, Minhao Cheng, Cho-Jui Hsieh & Thomas C. M. Lee

Taylor & Francis
Taylor & Francis Group

Check for updates

# A Review of Adversarial Attack and Defense for Classification Methods

Yao Li[a], Minhao Cheng[b], Cho-Jui Hsieh[b], and Thomas C. M. Lee[c]

[a]Department of Statistics & Operations Research, University of North Carolina at Chapel Hill, Chapel Hill, NC; [b]Department of Computer Science, University of California at Los Angeles, Los Angeles, CA; [c]Department of Statistics, University of California at Davis, Davis, CA

## ABSTRACT

Despite the efficiency and scalability of machine learning systems, recent studies have demonstrated that many classification methods, especially Deep Neural Networks (DNNs), are vulnerable to adversarial examples; that is, examples that are carefully crafted to fool a well-trained classification model while being indistinguishable from natural data to human. This makes it potentially unsafe to apply DNNs or related methods in security-critical areas. Since this issue was first identified by Biggio et al. and Szegedy et al., much work has been done in this field, including the development of attack methods to generate adversarial examples and the construction of defense techniques to guard against such examples. This article aims to introduce this topic and its latest developments to the statistical community, primarily focusing on the generation and guarding of adversarial examples. Computing codes (in Python and R) used in the numerical experiments are publicly available for readers to explore the surveyed methods. It is the hope of the authors that this article will encourage more statisticians to work on this important and exciting field of generating and defending against adversarial examples.

## 1. Introduction

Machine learning systems achieve state-of-the-art performance in various tasks in artificial intelligence, such as image classification, speech recognition, machine translation and game-playing (e.g., Simonyan and Zisserman 2014; Silver et al. 2016; Devlin et al. 2019). Despite their tremendous successes, machine learning models have been shown to be vulnerable to adversarial examples. By adding imperceptible perturbations to the original inputs, the attacker can produce adversarial examples to fool a learned classifier (e.g., Szegedy et al. 2014; Goodfellow, Shlens, and Szegedy 2015). Adversarial examples are indistinguishable from the original inputs to human, but are misclassified by the classifier. For an illustration, consider the following images in Figure 1.

To humans, the two images appear to be the same—the vision system of human will identify each image as a bagel. The image on the left side is an ordinary image of a bagel (the original image). However, the image on the right side is generated by adding a small and imperceptible perturbation that forces a particular classifier to classify it as a piano. The example in Figure 1 was generated to fool a deep neural network (DNN) based classifier. However, an adversarial example is not an isolated case that happens only to some neural networks, but a general case for most of the practically used neural networks. Besides, recent works have shown that linear models and many other general statistical learning models, such as logistic regression and tree-based models, are also vulnerable to adversarial examples (Chen et al. 2019). With the wide application of machine learning models, this causes serious concerns about the safety of machine learning systems in security sensitive areas, such as self-driving cars, flight control systems, healthcare systems, and so on. Therefore, studies on adversarial examples have attracted much attention in recent years.

Most of the studies on adversarial examples can be generally classified into two categories: works on how to efficiently generate adversarial examples (i.e., attack), and works on how to guard against such adversarial examples (i.e., defense). This article reviews the latest research on adversarial attack and defense, and compares the state-of-the-art attack and defense methods on benchmark datasets. We illustrate the main concepts via an image classification task with a continuous input space. However, we also note that there have been research activities on adversarial robustness for other applications with discrete input data, such as natural language processing (NLP), discrete models, and nearest neighbor classifiers (e.g., Jia and Liang 2017; Samanta and Mehta 2017; Gao et al. 2018; Cheng et al. 2020b; Yang et al. 2020b).

With this article, it is our hope that more statisticians will be engaged in this important and exciting field of attacking and defending classification methods. As to be demonstrated below, many problems from this field are statistical in nature, and statistical methodologies and principles can be adopted to improve existing methods or even to develop new methods. R codes that demonstrate how to attack a logistic regression model are provided for statisticians who are interested in this problem to experiment with. Finally, we note that there are other review articles in related topics that might be of interest to the reader; for example, Serban, Poll, and Visser (2020) reviewed adversarial phenomenon in the field of object detection, while
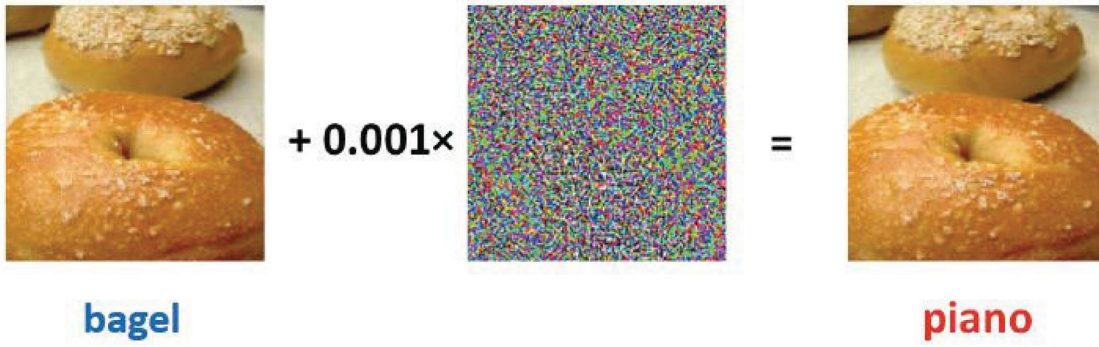
**Figure 1.** An example of adversarial example, taken from Chen et al. (2017).

other authors (e.g., Qiu et al. 2019; Yuan et al. 2019; Ren et al. 2020; Xu et al. 2020) provided reviews of adversarial examples with emphasis in computer vision and natural language processing from the perspective of computer science. These authors categorized the methods in a different manner when compared to ours.

### 1.1. Notation

In this article, all the vectors are denoted as bold symbols. The input to the classifier is represented by $x$ and the label associated with the input is represented by $y$. Thus, one observation is a pair $(x, y)$. The classifier is denoted as $f(\cdot)$ and $f(x)$ represents the output vector of the classifier. The dimension of $f(x)$ is equal to the number of classes of the dataset. Let $f(x)_i$ denote the score of predicting $x$ with label $i$. We caution that, in some works in the literature, $f(x)_i$ is taken as the "probability" that sample $x$ corresponds to label $i$, but very often these $f(x)_i$'s are not probabilities. The prediction of the classifier is denoted as $c(x) = \arg\max_i f(x)_i$; that is, the predicted label is the one with the highest prediction score. We use the $\ell_\infty$ and $\ell_2$ distortion metrics to measure similarity between inputs and report the $\ell_\infty$ distance in the normalized [0, 1] space, and the $\ell_2$ distance as the total root-mean-square distortion normalized by the total number of dimensions.

The rest of this article is organized as follows. Section 2 provides a quick summary of DNNs. This section can be omitted if the reader has a good understanding of the classification task, by treating a DNN as a black-box classifer. Then Section 3 describes the latest developments for attacking classification methods, while Section 4 presents defense methods for defending against such attacks. Lastly, numerical comparisons are given in Section 5 while concluding remarks, including future directions for statistical research, are offered in Section 6.

## 2. Deep Neural Networks

Deep Neural Networks (DNNs) have become one of the most prominent technologies of our time, as they achieve state-of-the-art performance in many machine learning tasks, including but not limited to image classification, text mining, and speech processing. A brief introduction to DNNs is given in this section to help readers understand attack and defense methods. More details of DNNs can be found, for example, in Goodfellow,

Bengio, and Courville (2016). To speed up the pace, the readers may skip this section (and come back later), which will not affect understanding the concepts behind those attack and defense methods. It is because the DNNs described below can be viewed as black-box classifiers and their details are not crucial for the comprehension of such concepts. However, for readers who are interested in the robustness of DNNs and not familiar with their architectures, this section would be helpful.

A DNN consists of a large class of models and learning methods. Here we describe a general form of DNN classifier, that is, widely applied in image classification. For a DNN model $f(\cdot)$ with $L$ layers, each layer is parameterized by a weight matrix $\theta_i$, which holds the knowledge of the DNN model and is updated during the training process. Each neuron of the DNN has an activation function $\phi_i$, which is usually used to introduce nonlinearity into the output of a neuron. Each function $\phi_i$ for $i \in \{1, \ldots, L\}$ is modeled using a layer of neurons, where the function $\phi_i$ takes the output of the previous layer ($h_{i-1}$) as input and produces $h_i$ as the output of the current layer. The function $\phi_i$ is usually chosen to be ReLU (rectified linear unit) $\phi(v) = \max(0, v)$, but there are also many other choices. In fully connected neural networks, the output of each layer is generated by multiplying a weight matrix with the previous layer output and adding a bias term. However, there could be other designs for the weights. For example, in a convolutional neural network (CNN), the weight $\theta_i$ is a kernel matrix. The kernel matrix scans over the input matrix, from left to right and from top to bottom, and each one of the values within the kernel is multiplied by the input value on the same position to generate the output matrix; see Figure 2 for an example. More details of convolutional neural network design can be found in LeCun et al. (1998). In general, a DNN can be expressed as:

$$f(x) = \phi_L(\theta_L, \phi_{L-1}(\theta_{L-1}, \ldots \phi_2(\theta_2, \phi_1(\theta_1, x)))).$$

As illustrated in Figure 3, a deep classifier is a machine learning model that uses a hierarchical composition of $L$ parametric functions to model an input $x$. Depending on the loss applied to the final layer, a neural network can easily handle both regression and classification tasks. For regression, there is only one output unit at the end of the network. For multi-class classification, there are $K$ output units, where $K$ is the number of classes.
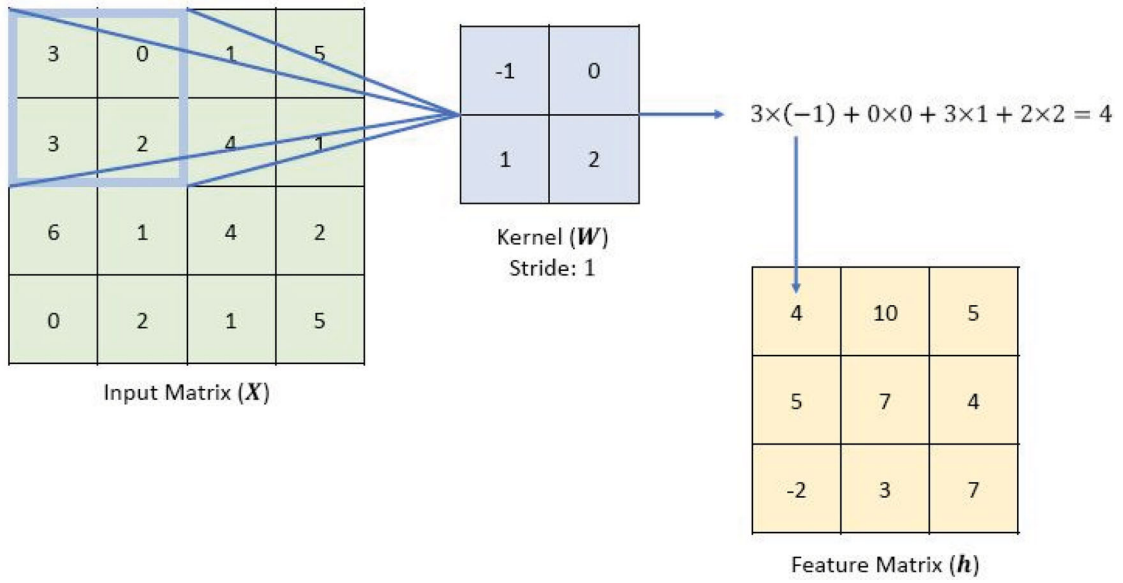
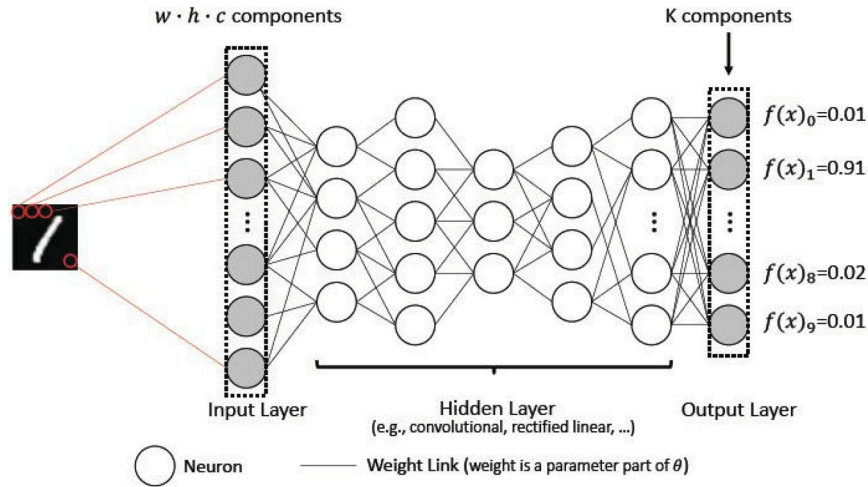**Figure 2.** Convolutional operation with stride (step size) equals to 1.



**Figure 3.** DNN Classifier: the model takes an image of a handwritten digit as input and predicts the probability of it being in one of the $K = 10$ classes for digits 0 to 9 (Papernot et al. 2017).

For the classification task, the cross-entropy loss is widely used:

$$\ell(\boldsymbol{x}, y) = - \sum_{i=1}^{K} 1\!\!1_{\{i=y\}} \log(f(\boldsymbol{x})_i).$$

Given a dataset with many known input-label pairs $(\boldsymbol{x}, y)$, the DNN will adjust its parameters ($\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n\}$) to reduce the prediction error between the prediction and the correct label. The generic approach to minimize the prediction loss is gradient descent, called back-propagation in this setting (Hecht-Nielsen 1992). Because of the compositional form of the model, the error gradient is derived using the chain rule for differentiation. The gradient is computed by a two-pass algorithm, where a forward pass is used to compute the loss and a backward pass is carried out to derive the gradient. In backward pass, error gradients with respect to network parameters are successively propagated from the network's output layer to its input layer. For large datasets, stochastic gradient descent is usually applied, where the gradient descent is done on a random subset of data.

At the testing stage, the DNN with fixed parameters is used to predict the label of unseen input. The output of the DNN is a probability vector $f(\boldsymbol{x})$, and the prediction is $c(\boldsymbol{x}) = \underset{i}{\arg\max} f(\boldsymbol{x})_i$.

DNNs achieve state-of-the-art performance in the task of image classification. For example, an vision transformer network (Dosovitskiy et al. 2021) can achieve over 99% accuracy when classifying natural images from the benchmark dataset CIFAR10 (Krizhevsky and Hinton 2009). Ideally, we expect the deep classifier to generalize well, making accurate predictions for inputs outside of the domain explored during training. However, recent studies on adversarial attack show that carefully crafted adversarial examples can easily fool the model.

## 3. Attack

Adversarial attacks are processes to generate an adversarial example based on a given natural sample and the victim model. Figure 4 illustrates this process of generating adversarial
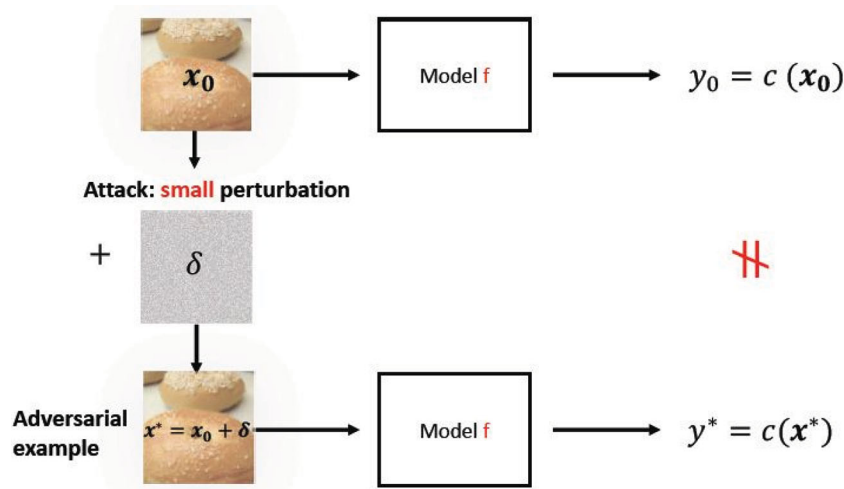
**Figure 4.** Illustration of adversarial attacks.

examples. Here $x_0$ denotes the natural input, and the DNN can correctly predict its label $y_0$. An adversarial attack aims to find a small perturbation $\delta$ such that the adversarial example $x^* = x_0 + \delta$, which looks similar to $x_0$ to humans, will be misclassified by the victim model.

Multiple attack methods have been introduced for the crafting of adversarial examples to attack various DNNs. In general, there are two kinds of attack goals: targeted and untargeted. For untargeted attack, the attack is successful if the input is predicted with any wrong label. Take Figure 1 as example. As long as the bagel image is not classified as bagel, the attack is successful. For targeted attack, the attack is successful only when the adversarial example is classified as the target class. In this example, if the target class is piano, the attack is successful only when the right side image is labeled as piano by the victim classifier.

Based on the information needed, the attack methods can be grouped into three categories: (1) gradient-based, (2) score-based, and (3) decision-based. Most of these methods can perform both targeted and untargeted attacks. In general, one attack method belongs to one of the three categories, while recently it has been shown that an ensemble of attacks from multiple categories can potentially lead to a stronger attack (Croce and Hein 2020). If all the information of the victim model, such as model structure, parameters and so on, is revealed to the attacker, the scenario is called white box setting. If only the predicted scores are available, the scenario is called soft-label black box setting. If only the predicted labels are revealed, the scenario is called hard-label black box setting. There are also gray box settings, where part of the model information is available. Notice that the part of the model information, that is, available/unavailable is problem dependent. For example, Yang et al. (2020a) defined the gray box as the scenario where the attacker has access to the classifier but not the design of the adversarial example detector. We will thus focus on white-box and black-box attacks in this article. The next three sections present representative examples and deeper discussions of the three categories of attack methods.

## 3.1. Gradient-Based Attack

Many existing attack methods fall in this category. These methods leverage the gradients of the loss w.r.t. the input to form adversarial examples. For instance, the Fast Gradient Sign Method (FGSM) (Goodfellow, Shlens, and Szegedy 2015) generates adversarial examples based on the sign of gradients, and uses a step size to control the $\ell_\infty$ norm of perturbation. The Basic Iterative Method (BIM) (Kurakin, Goodfellow, and Bengio 2016) and Projected Gradient Descent Attack (PGD) (Madry et al. 2018) can be viewed as iterative versions of FGSM. PGD crafts adversarial examples by running FGSM for a fixed number of iterations or until misclassification is achieved. The most effective gradient-based adversarial attack methods to date are C&W (Carlini and Wagner 2017b) and PGD. Both C&W attack and PGD have been frequently used to benchmark the defense algorithms due to their effectiveness (Athalye, Carlini, and Wagner 2018). Since gradient information is required to perform attack, gradient-based attack is mainly for the white-box setting. The process of crafting adversarial examples can be formulated as an optimization problem. Depending on the optimization formulations of the attack methods, gradient-based attack methods can be further divided into two sub-categories.

### 3.1.1. Constraint-Optimization Formulation Based Methods

The first sub-category consists of constraint-optimization formulation based methods. Given a model with a fixed parameter $\theta$ and an input pair $(x_0, y_0)$, the process of generating adversarial examples $x^*$ can be described by the following optimization problem:

$$x^* = x_0 + \delta \quad \text{with} \quad \delta = \arg\max_{\delta \in \mathcal{S}} L(\theta, x_0 + \delta, y_0), \quad (1)$$

where $L$ is the loss function, $\delta$ is the adversarial perturbation, and $\mathcal{S} \in \mathbb{R}^d$ is the set of allowed perturbations, usually chosen to be $\mathcal{S} = \{\delta | \|\delta\|_\infty \leq \epsilon\}$ for some $\epsilon$.

The optimization goal is to search for the adversarial perturbation that leads to wrong classification. Therefore, for untargeted attack, the loss function $L$ can be the loss function used to train the classifier, while for targeted attack, other loss functions designed for the target class should be used. By maximizing (1), the attacker is forcing the classifier to err on the classification task. To guarantee that the perturbation is not too large so that

$x_0$ and $x^*$ are indistinguishable to human, the search space is restricted to be within the $\epsilon$-ball around the input.

The optimization problem defined in (1) can be reformulated as

$$\arg\max_{\delta \in \mathcal{S}} L(\boldsymbol{\theta}, \boldsymbol{x}_0 + \boldsymbol{\delta}, y_0) = \underset{\|\boldsymbol{\delta}\|_\infty \le \epsilon}{\operatorname{argmax}} L(\boldsymbol{\theta}, \boldsymbol{x}_0 + \boldsymbol{\delta}, y_0), \quad (2)$$

which can be solved by gradient descent. Projected-Gradient Descent Attack (PGD) crafts adversarial examples by solving this optimization problem with the projected gradient descent method. It is proposed by Madry et al. (2018), which finds adversarial examples in an $\epsilon$-ball of the input. The PGD attack updates in the direction that decreases the probability of the original class most, then projects the result back to the $\epsilon$-ball of the input. The updating equation of the PGD attack is:

$$\boldsymbol{x}^{t+1} = \Pi_\epsilon \left\{ \boldsymbol{x}^t + \alpha \cdot \operatorname{sign}\left(\nabla_{\boldsymbol{x}} L(\boldsymbol{\theta}, \boldsymbol{x}^t, y)\right), \boldsymbol{x}_0 \right\}, \quad (3)$$

where $\boldsymbol{x}_0$ is the original input, $\boldsymbol{x}^t$ is the updated input in step $t$, $\epsilon$ controls the maximum distortion, $\alpha$ is the step size and $\nabla_{\boldsymbol{x}} L(\boldsymbol{\theta}, \boldsymbol{x}^t, y)$ represents the gradient of classification loss $L(\cdot)$ w.r.t. input $\boldsymbol{x}^t$. Basically, PGD attack constructs an adversarial example by adding or subtracting a small error term, $\alpha$, to each input dimension. The decision of adding or subtracting an error term depends on whether the sign of the gradient for an input dimension is positive or negative. Then the result from previous step is projected to the $\epsilon$-ball around the original input. If (3) is run for only one step, it is equivalent to the Fast Gradient Sign Method (FGSM) (Goodfellow, Shlens, and Szegedy 2015). If it is run for multiple steps, it is the PGD attack or the Basic Iterative Method (BIM) (Kurakin, Goodfellow, and Bengio 2016). Recently, Croce and Hein (2020) introduced a new attack method called Autoattack, which combines two extensions of PGD and a decision-based attack to automatically evaluate defense methods without parameter tuning.

### 3.1.2. Regularization-Optimization Formulation Based Methods

The second sub-category consists of regularization-optimization formulation based methods. The C&W attack proposed by Carlini and Wagner (2017a) is a representative one, and is by far one of the strongest attack methods. It can perform both targeted and untargeted attacks, which are formulated as the following optimization problem:

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \|\boldsymbol{x} - \boldsymbol{x}_0\|_2^2 + c g(\boldsymbol{x}) \right\}, \quad (4)$$

where the first term enforces small distortion of the original input and the second term is the loss function that measures the success of the attack. The parameter $c > 0$ controls the trade-off between distortion and attack success.

If it is for untargeted attack, which means the attacker only wants the classifier to make a mistake and does not care the predicted label of the adversarial example, then $g(\boldsymbol{x})$ is defined as:

$$g(\boldsymbol{x}) = \max\{f(\boldsymbol{x})_{y_0} - \max_{i \ne y_0} f(\boldsymbol{x})_i, 0\},$$

where $y_0$ denotes the true label of input $\boldsymbol{x}_0$, $f(\boldsymbol{x})_{y_0}$ represents the score of predicting input $\boldsymbol{x}$ with label $y_0$, and $f(\boldsymbol{x})_i$ represents the score of predicting input $\boldsymbol{x}$ with label $i$. Minimizing $g(\boldsymbol{x})$ will make the prediction score of true class $y_0$ smaller than the prediction scores of classes other than $y_0$. Therefore, the adversarial example $\boldsymbol{x}^*$ will be classified to a wrong class.

If the attacker wants the adversarial example to be classified into a specific target class with label $t$, where $t \ne y_0$, then $g(\boldsymbol{x})$ is defined as:

$$g(\boldsymbol{x}) = \max\{\max_{i \ne t} f(\boldsymbol{x})_i - f(\boldsymbol{x})_t, 0\}.$$

The above targeted attack loss function will push the prediction score of the target class $t$ to be higher than those of other classes. Carlini and Wagner (2017a) showed that their attack can successfully bypass 10 different defense methods designed for detecting adversarial examples.

Inspired by an elastic-net method proposed by Zou and Hastie (2005), Chen et al. (2018) proposed the elastic-net attack (EAD), which also belongs to this sub-category of regularization-optimization formulation based methods. EAD formulates the process of generating adversarial examples as an elastic-net regularized optimization problem, which can be viewed as extended version of the C&W Attack. Their formulation of elastic-net attacks to DNNs for crafting an adversarial example with respect to a labeled sample is as follows:

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \|\boldsymbol{x} - \boldsymbol{x}_0\|_2^2 + \beta \|\boldsymbol{x} - \boldsymbol{x}_0\|_1 + c g(\boldsymbol{x}) \right\},$$

where $g(\boldsymbol{x})$ is the same as the one defined in C&W attack. The authors showed that EAD can improve attack transferability and complement adversarial training.

We note that most gradient-based attack methods fall into the above two sub-categories, but there are also other kinds of gradient-based attack methods, such as Jacobian-based Saliency Map Attack (Papernot et al. 2016), Deepfool (Moosavi-Dezfooli, Fawzi, and Frossard 2016), Network for Adversary Generation (Mopuri et al. 2018) and Adversarial Transformation Networks (Baluja and Fischer 2018). All of these methods perform adversarial attacks based on the gradient information.

### 3.1.3. Choice of Perturbation Set in Adversarial Attacks

Both constraint and regularization optimization based formulations require a distance measurement to quantify the difference between the original example and the perturbed example. Despite $\ell_\infty$ norm being used for PGD-based attacks (1) and the $\ell_2$ norm being widely used in C&W based attacks (4), in general we can use any $\ell_p$ norm for both constraint and regularization based formulations. For example, $\ell_1$ norm and a mixed of different norms have been used in (Chen et al. 2018). Further, recently it has been recognized that $\ell_p$ norm based attacks may not be realistic, so researchers have started to explore more flexible perturbation sets. For instance, Wong, Schmidt, and Kolter (2019) uses Wasserstein distance to measure the perturbation strength, and Wong and Kolter (2020) proposed learning the perturbation sets based on human perception.

### 3.2. Score-Based Attack

In reality, detailed model information, such as the gradient, may not be available to the attackers. The score-based attack methods

do not require access to gradients. They perform adversarial attacks based on the output scores $f(\boldsymbol{x})_i$'s of the victim classifier. For example, Chen et al. (2017) proposed a method to estimate the gradient with score information and craft adversarial examples with the estimated gradient. Ilyas et al. (2018) introduced a method which leverages natural evolutionary strategy to estimate the gradient and generate adversarial examples. In general, score-based attacks can be divided into two sub-categories.

### 3.2.1. Gradient-Approximation Based Methods
The first sub-category are gradient-approximation based methods. As the name suggests, these methods first approximate the gradients or the signs of the gradients, then generate adversarial examples using the approximated information.

Zeroth Order Optimization Based Attack (ZOO) proposed by Chen et al. (2017) uses a finite difference method to approximate the gradient of the loss w.r.t. the input. Then the C&W attack is applied to generate an adversarial example. It uses the following formula to estimate the gradient:

$$\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}_{(i)}} \approx \frac{f(\boldsymbol{x} + h\boldsymbol{e}_i) - f(\boldsymbol{x} - h\boldsymbol{e}_i)}{2h},$$

where $h$ is a small constant and $\boldsymbol{e}_i$ is a standard basis vector with only the $i$th component as 1, and $i$ ranges from 1 to the dimension of input. The time used to estimate the gradients grows with the dimension. When the dimension of the input is large, the authors introduced several techniques to scale-up the estimation, making it possible for the method to craft adversarial examples in reasonable time for large DNNs trained on large datasets, such as ImageNet (Deng et al. 2009).

Many other methods are introduced to efficiently approximate the gradient and generate adversarial examples based on it, such as NES attack (Ilyas et al. 2018) and Bandits Attack (Ilyas, Engstrom, and Madry 2019). NES attack uses natural evolutionary strategies (hence, the acronym NES) to estimate the gradient of loss w.r.t. the input then generates adversarial examples based on the estimated gradient:

$$\frac{1}{\sigma n} \sum_{i=1}^{n} \boldsymbol{\delta}_i f(\boldsymbol{x} + \sigma \boldsymbol{\delta}_i),$$

where $n$ is the number of searches to estimate gradient, $\boldsymbol{\delta}_i$ is random direction sampled from $\mathcal{N}(0, I)$ and $\sigma$ is the standard deviation of the search step. The authors also extended the method to a partial-information setting, where only part of the scores or top-$k$ sorted labels are given.

In fact, the most important information for adversarial attack is the sign of the gradient, which provides the direction of optimization. Many methods are proposed to estimate the sign of the gradient and use that information to craft adversarial examples; notable examples include SignSGD (Liu et al. 2018a) and Sign hunter (Al-Dujaili and O'Reilly 2020). To further improve the efficiency of gradient approximation or sign estimation, some researchers propose using a substitute model, which is a model trained with the same data as the victim model and performs similarly to the victim model. Representative examples include Subspace attack (Guo, Yan, and Zhang 2019b) and Transfer-based prior (Cheng et al. 2019a). Also, Wang et al. (2020a) shows

how to exploit the data distribution to identify importance subspace for black-box attacks. These attack methods also belong to the gradient-approximation based category since their key element is to approximate gradient information efficiently.

### 3.2.2. Other Methods
The second sub-category of score-based methods consists of methods that do not approximate gradient related information to generate adversarial examples. For example, Li et al. (2019) proposed the Gaussian black-box adversarial attack (Nattack), which searches the adversarial examples by modeling the adversarial population with a Gaussian distribution. The intuition comes from the fact that one can find various adversarial examples for the same input using different attack methods, which suggests there is a population of adversarial examples. Other methods that fall into this sub-category include the GenAttack (Alzantot et al. 2018), the Simple Black-Box Attack (Guo et al. 2019a) and Square attack (Andriushchenko et al. 2020).

### 3.3. Decision-Based Attack
In many practical situations, the attacker has access only to the predicted labels of the model, but not any gradient or score information. When only the predicted label $c(\boldsymbol{x})$ is available, both gradient-based and score-based methods do not work. Papernot et al. (2017) introduced a transfer attack method which requires only the observations of the labels predicted by the model. The main idea is to train a substitute model which is similar to the target model and attack the substitute model instead. Boundary Attack was subsequently proposed by Brendel, Rauber, and Bethge (2018), which searches for adversarial examples based on random-walk on the decision boundary. Many extensions of Boundary Attack have been proposed to improve the efficiency and performance of it (Brunner et al. 2019; Chen and Jordan 2019; Chen, Jordan, and Wainwright 2020; Guo, Frank, and Weinberger 2020). There are also decision-based attack methods that are neither transfer-based nor random-walk based (Ilyas et al. 2018; Cheng et al. 2020a; Guo, Yan, and Zhang 2019b). In general, the goal of decision-based attack is to generate adversarial examples with the predicted labels returned by the victim model. Decision-based attack methods can be divided into three sub-categories.

### 3.3.1. Transfer-Based Attacks
The first category consists of transfer-based attack methods. Various researchers have observed that, if two DNNs are trained with similar data, even though the two models may have very different structures, adversarial examples generated on one model can be used to fool another one. Based on this observation, Papernot et al. (2017) proposed to train a substitute model based on a small amount of training data and generate adversarial examples based on the substitute model. They showed that the adversarial examples generated on the substitute model can also fool the target classifier. The proposed method does not require too many samples to train the substitute model and can achieve relatively high success rate in untargeted task. However, the method does not work well in targeted task. Liu et al. (2017) proposed an ensemble-
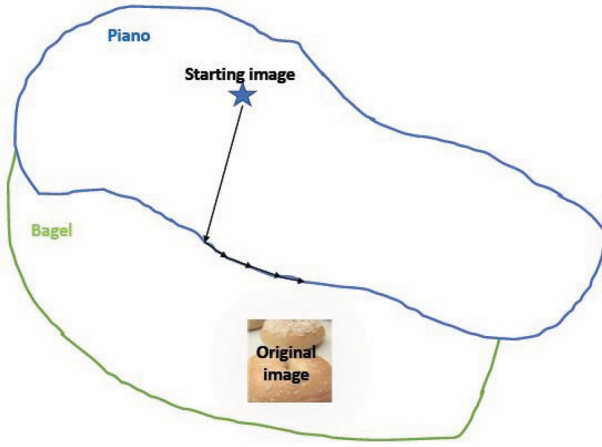
**Figure 5.** An illustration of Boundary Attack. In essence the Boundary Attack performs rejection sampling along the boundary between adversarial and non-adversarial images (Brendel, Rauber, and Bethge 2018).

based approach to generating transferable adversarial examples. This ensemble approach increases the proportion of target adversarial examples transferring with their target labels.

### 3.3.2. Random-Walk Based Attacks

The second category consists of methods that are based on random walk on the boundary. Brendel, Rauber, and Bethge (2018) proposed Boundary Attack, a method that does not rely on the gradient of loss w.r.t. the input and performs well under both targeted and untargeted settings. The method starts with a sample categorized in the targeted class and seeks to minimize the perturbation while staying adversarial. The process of boundary attack is shown in Figure 5 and details of the method are described in Algorithm 1.

---

**Algorithm 1** Boundary Attack

1: **Input:** decision model $c(\cdot)$, original input $\boldsymbol{x}$
2: **Output:** $\boldsymbol{x}^{adv}$
3: **Initialization:** $t = 0$, $\tilde{\boldsymbol{x}}^0 \sim \mathcal{U}(0, 1)$ s.t. $c(\tilde{\boldsymbol{x}}^0) =$ target class
4: **while** $t < T$ **do**
5: $\quad t = t + 1$
6: $\quad$ draw random perturbation from proposal distribution $\boldsymbol{\eta}^t \sim \mathcal{P}(\tilde{\boldsymbol{x}}^{t-1})$
7: $\quad$ **if** $c(\tilde{\boldsymbol{x}}^{t-1} + \boldsymbol{\eta}^t) =$ target class **then**
8: $\quad\quad$ set $\tilde{\boldsymbol{x}}^t = \tilde{\boldsymbol{x}}^{t-1} + \boldsymbol{\eta}^t$
9: $\quad$ **else**
10: $\quad\quad$ set $\tilde{\boldsymbol{x}}^t = \tilde{\boldsymbol{x}}^{t-1}$
11: $\quad$ **end if**
12: **end while**
13: **Return** $\tilde{\boldsymbol{x}}^T$

Note that $c(\boldsymbol{x})$ is the label assigned to the input $\boldsymbol{x}$ by the classifier, and $T$ is the maximum number of iterations.

---

Define input space as $\mathcal{D}$, the process of drawing $\boldsymbol{\eta}^t$ from the proposal distribution is as follows:

- Draw $\boldsymbol{\eta}^t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$.
- Rescale and clip $\boldsymbol{\eta}^t$ to make sure $\tilde{\boldsymbol{x}}^{t-1} + \boldsymbol{\eta}^t \in \mathcal{D}$ and $\|\boldsymbol{\eta}^t\|_2 = \delta \cdot d(\boldsymbol{x}, \tilde{\boldsymbol{x}}^{t-1})$, where $d(\cdot, \cdot)$ represents distance

function between two samples and $\delta$ is a hyperparameter that controls the scale of the perturbation.
- Orthogonal perturbation: project $\boldsymbol{\eta}^t$ onto a sphere around the original input $\boldsymbol{x}$ such that $d(\boldsymbol{x}, \tilde{\boldsymbol{x}}^{t-1} + \boldsymbol{\eta}^t) = d(\boldsymbol{x}, \tilde{\boldsymbol{x}}^{t-1})$.
- Move towards the original input such that $\tilde{\boldsymbol{x}}^{t-1} + \boldsymbol{\eta}^t \in \mathcal{D}$ and $d(\boldsymbol{x}, \tilde{\boldsymbol{x}}^{t-1}) - d(\boldsymbol{x}, \tilde{\boldsymbol{x}}^{t-1} + \boldsymbol{\eta}^t) = \epsilon \cdot d(\boldsymbol{x}, \tilde{\boldsymbol{x}}^{t-1})$ both hold, where $\epsilon$ controls the step size of the movement.

Boundary Attack achieves performance comparable to state-of-the-art white-box attacks on DNNs trained for classification. Chen and Jordan (2019) introduced Boundary Attack++, using binary information at the decision boundary to estimate gradient direction. They showed that Boundary Attack++ requires significantly fewer model queries than Boundary Attack. Guo, Frank, and Weinberger (2020) proposed a decision-based black-box attack, which improves the query-efficiency of Boundary Attack by restricting the search for adversarial examples to a low frequency domain. There are many other methods belong to this category, such as Guessing Smart (Brunner et al. 2019).

### 3.3.3. Optimization-Based Attacks

Instead of conducting random walks, a process which does not have any convergence guarantees, more recently researchers have found that attacks in the decision-based setting can also be formulated as solving a zeroth order optimization problem (Cheng et al. 2019b). They showed that both PGD and C&W loss are ill-defined in the decision-based setting, and instead we need to redefine the problem as finding the best direction $\boldsymbol{\theta}$ of adversarial examples. Given $\boldsymbol{x}_0$, a function $g(\cdot)$ that measures the distance between $\boldsymbol{x}_0$ and the decision boundary along direction $\boldsymbol{\theta}$ can be defined as

$$g(\boldsymbol{\theta}) = \arg\min_{\lambda > 0} \left( f\left(\boldsymbol{x}_0 + \lambda \frac{\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|}\right) \neq y_0 \right). \tag{5}$$

The attack problem can then be posed as

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} g(\boldsymbol{\theta}), \tag{6}$$

and the adversarial example, that is, closest to $\boldsymbol{x}_0$ is $\boldsymbol{x}^* = \boldsymbol{x}_0 + g(\boldsymbol{\theta}^*) \frac{\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|}$. Although the gradient of $g(\boldsymbol{\theta})$ cannot be directly computed, the function value of $g(\boldsymbol{\theta})$ can be computed by binary search, thus standard zeroth order optimization solvers can be applied for solving (6). In OPT attack (Cheng et al. 2019a), Randomized Gradient-Free (RGF) method (Nesterov and Spokoiny 2017) is used to address the problem. Later, Cheng et al. (2020a) showed that the gradient sign of (6) can be computed in a more query-efficient way, leading to an improved attack called Sign-OPT. On the other hand, Chen, Jordan, and Wainwright (2020) proposed another optimization-based formulation, leading to yet another query-efficient algorithm called HotSkipJump Attack. Recently, Chen and Gu (2020) proposed RayS, which reformulates the continuous optimization problem in Cheng et al. (2019a) into a discrete one for $\ell_\infty$ norm attack.

## 4. Defense

There has been extensive research on improving the robustness of DNNs for defending against adversarial examples. In general, methods that aim to increase model robustness fall into the

following four main categories: (1) augmenting the training data with adversarial examples, (2) leveraging randomness to defend against adversarial attacks, (3) removing adversarial perturbations with projection, and (4) detecting the adversarial examples instead of classifying them correctly.

## 4.1. Adversarial Training

To improve the robustness of a classifier against adversarial examples, one natural idea is to train the model with adversarial examples. Goodfellow, Shlens, and Szegedy (2015) proposed generating adversarial examples with FGSM attack and adding them back to the training dataset to improve the robustness of the model. Later Kurakin, Goodfellow, and Bengio (2016) suggested using a multi-step FGSM to further improve the adversarial robustness.

Madry et al. (2018) introduced a min-max formulation, which iteratively generates adversarial examples with PGD attack while training the model. Athalye, Carlini, and Wagner (2018) showed that Madry's adversarial training can survive strong attacks while many other state-of-the-art defense methods cannot. Instead of minimizing the original loss function $L(\theta, x, y)$, the following min-max objective function is used:

$$\min_{\theta} \max_{\delta \in S} L(\theta, x + \delta, y).$$

The inner maximization problem aims to find an adversarial perturbation of a given data point $x$ within the perturbation set $S$ that achieves a high loss. Indeed, this is precisely the problem of attacking any given neural network. On the other hand, the goal of the outer minimization problem is to find the model parameters that minimize the "adversarial loss" given by the inner attack problem. Therefore, the model is trained with the adversarial example in the $\epsilon$-ball of the original sample. Recently, it has been shown that such min-max optimization is guaranteed to converge under certain assumptions (Gao et al. 2019).

Many defense methods based on Madry's adversarial training have been proposed to further improve its efficiency and performance. For example, Zhang et al. (2019b) proposed TRADES, a theoretically driven upper bound minimization algorithm that achieved the top-1 rank in the NeurIPS 2018 defense competition.

In other methods, researchers treated misclassified examples and correctly classified examples as the same when performing adversarial training. Ding et al. (2020) noticed the importance of misclassified examples and improved the performance of adversarial training by combining the usual cross-entropy loss with a margin maximization loss term applied to the correctly classified examples. Wang et al. (2020b) found that misclassified examples have more impact on the final robustness than correctly classified examples and incorporate mis-classified examples in adversarial training as a regularizer.

Other than just adding adversarial examples into the training process, Wang et al. (2019) studied the convergence quality of adversarial examples found in the inner maximization and proposed a dynamic adversarial training method that changes adversarial strength in inner maximization according to convergence quality. The convergence of adversarial training has also been studied in many other works (e.g., Gao et al. 2019).

Adversarial training performs well against attacks but generating adversarial examples during training could be expensive. Consequently, various researchers have developed methods for reducing the computational overhead brought by adversarial training (Shafahi et al. 2019; Zhang et al. 2019a; Wong, Rice, and Kolter 2020). Many extensions of adversarial training have also been proposed to improve the performance of adversarial training (Carmon et al. 2020; Zhang et al. 2019b; Gowal et al. 2020; Wu, Xia, and Wang 2020; Pang et al. 2020, 2021).

## 4.2. Randomization

Another group of effective defense methods leverages randomness to defend against adversarial examples. Adversarial perturbation can be viewed as noise, and various methods have been proposed to improve the robustness of DNNs by incorporating random components into the model. The randomness can be introduced to:

- the input; that is, randomizing the input of a neural network to remove the potential adversarial perturbation (e.g., Xie et al. 2018; Cohen, Rosenfeld, and Kolter 2019),
- the hidden layer output; that is, adding Gaussian noise to the input and hidden output (e.g., Liu et al. 2018b) and introducing pruning methods to randomize the network output (e.g., Dhillon et al. 2018), or
- the parameters of the classifier; that is, leveraging a Bayesian component to add randomness to the weights of the model (e.g., Liu et al. 2019).

The robustness of neural networks against random noise has been analyzed both theoretically (Fawzi, Moosavi-Dezfooli, and Frossard 2016; Franceschi, Fawzi and Fawzi 2018) and empirically (Dodge and Karam 2017).

Xie et al. (2018) introduced a simple preprocessing method to randomize the input of neural networks, hoping to remove the potential adversarial perturbation. During the testing phase, the input is randomly resized to several sizes, then around each of the resized inputs, zeros are randomly padded. The authors demonstrated that this simple method can be applied to large-scale datasets, such as Imagenet. Similarly, Zantedeschi, Nicolae, and Rawat (2017) showed that, by using a modified ReLU activation layer (called BReLU) and adding noise to the origin input to augment the training data, the learned model will gain some stability to adversarial examples. However, other researchers have found that this defense method is not robust against strong white-box attack, such as PGD and C&W (Carlini and Wagner 2017b).

Instead of adding random components to the input space, Liu et al. (2018b) proposed a "noise layer" to introduce randomness to both the input and the hidden layer output. In this "noise layer," randomly generated Gaussian noise is added to the input:

$$x \leftarrow x + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2 I),$$

where $\sigma$ is a hyperparameter. Larger values of $\sigma$ lead to better robustness but worse prediction accuracy, while smaller values of $\sigma$ result in better prediction accuracy but deteriorated robustness. The noise layer is applied in both training and testing

phases, so the prediction accuracy will not be largely affected. The authors showed that Random Self-Ensemble (RSE) is equivalent to an ensemble possessing an infinite number of noisy models without any additional memory overhead. However, the experiments demonstrated that, although RSE can increase the robustness of DNNs, it also sacrifices a nonnegligible amount of accuracy. Later, Liu et al. (2019) introduced a new min-max formulation to combine adversarial training with Bayesian Neural Networks (BNNs). All weights in a Bayesian neural network are represented by probability distributions over possible values, rather than having a single fixed value. The proposed framework, called Adv-BNN, combines adversarial training with randomness, is shown to have significant improvement over previous approaches including RSE and Madry's adversarial training.

Although these methods demonstrate improvement of robustness against adversarial examples, they typically lack theoretical guarantee for their performances. Lecuyer et al. (2019); Li et al. (2018a); Cohen, Rosenfeld, and Kolter (2019) introduced a method called Randomized Smoothing and proved a tight robustness guarantee in the $\ell_2$ norm. The details of Randomized Smoothing are summarized below:

- Training stage:

  1. Train the base classifier with Gaussian data augmentation with variance $\sigma^2$. That is, the base classifier is trained on the natural examples and noisy examples, where the noisy examples are generated by adding Gaussian noise to natural examples.
  2. The trained base classifier is denoted as $f$.

- Testing stage: for a given input testing sample $\boldsymbol{x}$:

  1. Generate $n$ noise corrupted samples by adding Gaussian noise with variance $\sigma^2$ to the input $\boldsymbol{x}$.
  2. Feed the $n$ noise corrupted samples into the trained base classifier $f$ and obtain the top-2 frequent predicted classes from the predictions. The top-2 frequent classes are denoted as $\hat{c}_A, \hat{c}_B$ and the corresponding frequencies are denoted as $n_A, n_B$.
  3. If BINOMPVALUE$(n_A, n_A + n_B, 0.5) \leq \alpha$, the final predicted class is $\hat{c}_A$, otherwise abstain. The function BINOMPVALUE$(n_A, n_A + n_B, p)$ returns the $p$-value of the two-sided hypothesis test that $n_A \sim$ Binomial$(n_A + n_B, p)$ and $\alpha$ is the abstention threshold generated based on Hung and Fithiane (2019).

They also provided a process to generate certificated radius around the input. Theoretical analysis was provided to show that the classifier is robust around the input within the found $\ell_2$ radius.

### 4.3. Projection

The development of generative models, such as auto-encoders and generative adversarial networks, gives rise to another line of research, which removes adversarial noise by fitting generative models on the training data (Jalal et al. 2017; Meng and Chen 2017; Li et al. 2018b; Samangouei, Kabkab, and Chellappa 2018). These defense mechanisms employ the power of generative models to combat the threat from adversarial examples. The input to the classifier is first fed into the generative model and then classified. Since the generative model is trained on natural examples, adversarial examples will be projected to the manifold learned by the generative model. Furthermore, "projecting" the adversarial examples onto the range of the generative model can have the desirable effect of reducing the adversarial perturbation.

Meng and Chen (2017) introduced MagNet as a robust framework against adversarial examples. It trains multiple auto-encoders to move adversarial examples closer to the manifold of natural examples. An auto-encoder is a type of neural network that consists of two major parts: the encoder that maps the input to a low-dimensional space and the decoder that recovers the input from the low-dimensional embedding. The auto-encoder is usually trained on the reconstruction loss w.r.t. the input. Therefore, the high-dimensional data is summarized by the low-dimensional embedding through the training process. In MagNet, one auto-encoder is chosen at random at testing time to filter the input samples, and thus adversarial perturbation could potentially be removed through this encoding and decoding process.

Instead of using auto-encoders, Samangouei, Kabkab, and Chellappa (2018) proposed to train a Generative Adversarial Networks (GAN) (Goodfellow, Shlens, and Szegedy 2015) to fit the training data distribution and help remove adversarial noise. It first trains a GAN to model the distribution of the training data. Then, at inference time, it finds an output of the GAN, that is, close to the input, and feeds that output into the classifier. This process "projects" input samples onto the range of the GAN's generator, which can potentially help remove the effect of adversarial perturbations. Here the GAN is proposed to estimate the generative model through an adversarial process, in which two major parts are simultaneously trained: a generative model $\boldsymbol{G}$ that captures the data distribution, and a discriminative model $\boldsymbol{D}$ that estimates the probability that a sample came from the training data rather than $\boldsymbol{G}$. The objective of training a GAN is:

$$\min_{\boldsymbol{G}} \max_{\boldsymbol{D}} \left\{ \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[ \log \boldsymbol{D}(\boldsymbol{x}) \right] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})} \left[ \log(1 - \boldsymbol{D}(\boldsymbol{G}(\boldsymbol{z}))) \right] \right\},$$

where $p_{\text{data}}(\boldsymbol{x})$ represents the data distribution and $p_{\boldsymbol{z}}(\boldsymbol{z})$ represents a prior distribution, that is, predefined. Therefore, $\boldsymbol{D}$ and $\boldsymbol{G}$ together play a two-player min-max game to recover the data distribution.

Defense-GAN is similar to MagNet as both of these methods aim to filter out the adversarial noise by using a generative model. Samangouei, Kabkab, and Chellappa (2018) showed that Defense-GAN is more robust than MagNet on several benchmark datasets. There are some other works also follow similar patterns (Jalal et al. 2017; Song et al. 2018).

All of the above methods train a separate generative model to perform the adversarial noise filtering. Li et al. (2018b) proposed a different defense framework, termed ER-Classifier, which combines the process of filtering and classification in one framework. In fact, any deep classifier can be viewed as a combination of these two parts as illustrated in Figure 6: an encoder part to extract useful features from the input data and
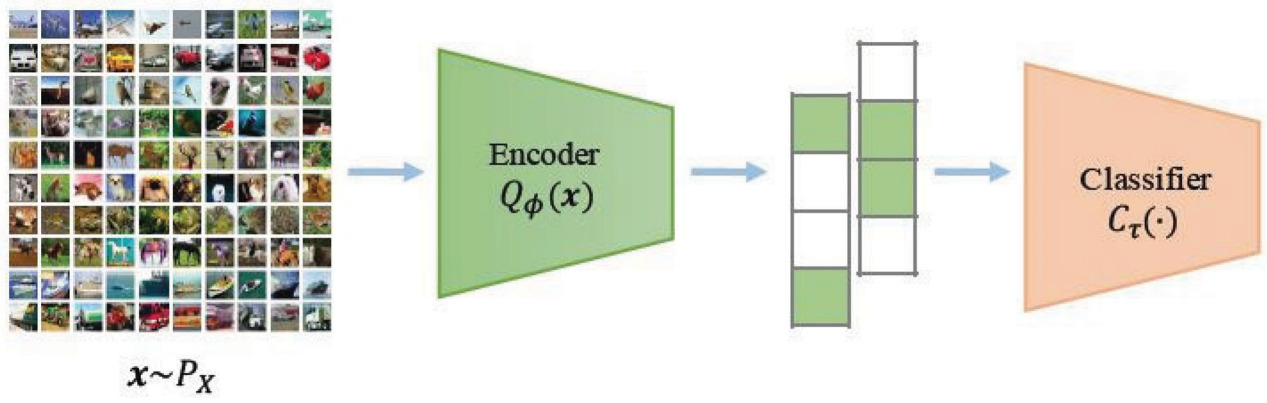
**Figure 6.** A deep classifier.

a classifier part to perform classification based on the extracted features. Both the encoder and the classifier are neural networks.

ER-Classifier is similar to a regular deep classifier, which first projects the input to a low-dimensional space with an encoder $G$, then performs classification based on the low-dimensional embedding with an classifier $C$. The novelty is that at the training stage, the low-dimensional embedding of ER-Classifier is stabilized with a discriminator $D$ by minimizing the dispersion between the distribution of the embedding and the distribution of a selected prior. The goal of the discriminator is to separate the true code sampled from a prior and the "fake" code produced by the encoder, while the encoder will try to produce generated code, that is, similar to the true one. The result of this competition is that the distribution of the embedding space will be pushed toward the prior distribution. Therefore, it is expected that this regularization process can help remove the effects of any adversarial distortion and push the adversarial examples back to the natural data manifold. Another difference is that the embedding space dimension is much smaller for the ER-classifier, when compared with a general deep classifier, making it easier for the training process to converge. Details of the framework are illustrated in Figure 7. Li et al. (2018b) showed that ER-Classifier achieves state-of-the-art performance against strong adversarial attack methods on several benchmark datasets.

### 4.4. Detection

Methods in this fourth category aim to detect the existence of adversarial examples, rather than trying to classify them into the correct classes. The main assumption behind these methods is that the adversarial examples came from a different distribution when comparing to those from the natural data. In other words, adversarial examples do not lie on the data manifold, and DNNs classify correctly only the samples near the manifold of training data (Tanay and Griffin 2016). Therefore, adversarial examples may be detected with carefully designed detectors.

A straightforward way toward adversarial example detection is to build a simple binary classifier separating the adversarial examples apart from the clean data (Gong, Wang, and Ku 2017). The advantage is that it serves as a preprocessing step without imposing any assumptions on the model it protects. However, this method suffers from a generalization limitation.

In a similar method, a small "detector" subnetwork is trained as an augmentation of DNNs on the binary classification task to distinguish true data from adversarial perturbations (Metzen et al. 2017), where the inputs (to this detector) are taken from intermediate layers of the classification network. Alternatively, a cascade classifier may be built on the outputs from the convolutional layers (Li and Li 2017) to efficiently detect adversarial examples. Zheng and Hong (2018) proposed modeling the output of a classifier with Gaussian mixture models and performing hypothesis testing to detect adversarial examples. Roth, Kilcher, and Hofmann (2019) showed that adversarial examples exist in cone-like regions in very specific directions from their corresponding natural inputs and proposed a new test statistic to detect adversarial examples with the findings. In a recent work, Yang et al. (2020a) studied the feature attributions of adversarial examples and proposed a detection method based on feature attribution scores.

Instead of building a detector based on input or output of the network, another line of research proposes to study the characteristics of adversarial examples and build detectors based on such features (Feinman et al. 2017; Grosse et al. 2017; Lee et al. 2018; Ma et al. 2018; Raghuram et al. 2020). In order to detect adversarial examples that lie far from the data manifold, Feinman et al. (2017) proposed KD-Detection to perform density estimation on the training data in the feature space of the last hidden layer to help detect adversarial examples.

Kernel density estimation is used to measure how far a sample is from the provided data manifold. Suppose that $x_1, \ldots, x_n$ are training samples drawn from an unknown probability density $p_X(x)$. Given $x$, we can use the following function to estimate the density score at $x$:

$$\hat{p}_X(x) = \frac{1}{n} \sum_{i=1}^{n} K_\sigma(x, x_i),$$

where $K_\sigma(\cdot, \cdot)$ is a kernel function. In the KD-Detection framework, one kernel density model is fitted for each class. Therefore, if $x$ is predicted with label $y$, the samples $\{x_i\}_{i=1}^{n}$ used to do the estimation are training samples from class $y$. After the kernel density is fitted, both natural examples and adversarial examples are fed into the model and density function to generate density scores. A logistic regression model is then fitted on the kernel density scores to detect whether the input is adversarial or not.
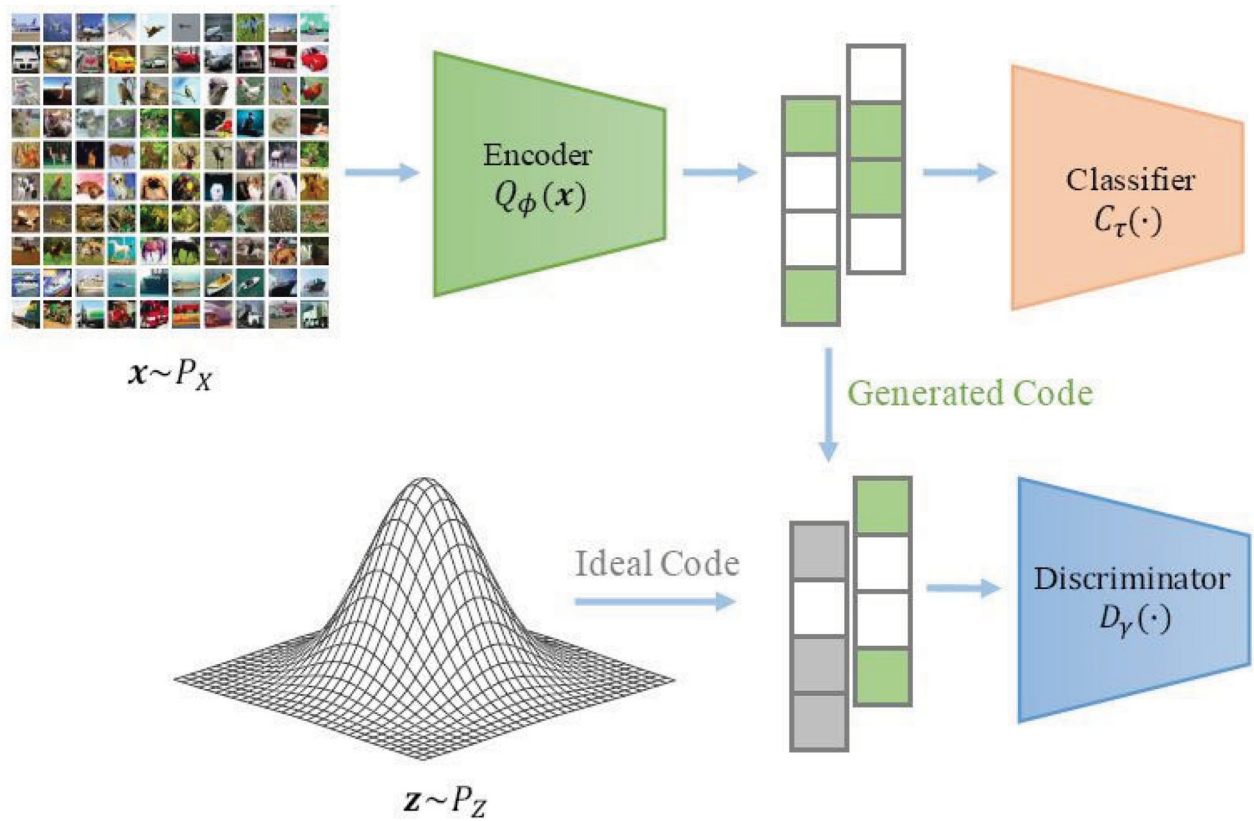
**Figure 7.** Overview of ER-Classifier framework.

Many recent works fall into this category, studying the properties of adversarial examples and detecting them with hidden layer features. Ma et al. (2018) observed Local Intrinsic Dimension (LID) of hidden-layer outputs differ between the original and adversarial examples, and leveraged these characteristics to detect adversarial examples. In another work, Lee et al. (2018) generated the class conditional Gaussian distributions with respect to hidden layer output of the DNN under Gaussian discriminant analysis, which result in a confidence score based on the Mahalanobis distance (MAHA), followed by a logistic regression model on the confidence scores to detect adversarial examples. Recently, a joint statistical test pooling information from multiple layers is proposed in Raghuram et al. (2020) to detect adversarial examples.

## 5. Numerical Experiments

In this section, we show how the attack methods fool various classifiers, including classical models like logistic regression and more recent models like DNNs. In the first part, the results of attacking a logistic regression model with projected gradient descent attack are shown. In the second part, we select representative attack methods from each category (gradient-based, score-based, and decision-based) and compare their performances. In the last part, we select one defense method from each category (adversarial training, randomization, projection, and detection) and compare their performances. The following benchmark datasets are used in our experiments:

- MNIST (LeCun 1998): a handwritten digit dataset that consists of 60,000 training images and 10,000 testing images. These are $28 \times 28$ black and white images in 10 different classes.

- CIFAR10 (Krizhevsky and Hinton 2009): a natural image dataset that contains 50,000 training images and 10,000 testing images with 10 different classes. These are low resolution $32 \times 32$ color images.

The input image is stored in RGB format with three channels, where the pixel values of each channel range from 0 to 256. We use the $\ell_\infty$ and $\ell_2$ distortion metrics to measure similarity between images and report the $\ell_\infty$ distance in the normalized $[0, 1]$ space (e.g., a distortion of 0.031 corresponds to 8/256), and the $\ell_2$ distance as the total root-mean-square distortion normalized by the total number of pixels. Computing codes for carrying out the following numerical experiments can be downloaded at *https://github.com/reviewadvexample/revew_adv_defense*.

### 5.1. Attacking Logistic Regression Model

We train a logistic regression model to perform binary classification on two handwritten digits, zero and one. The model is trained on all the images with label 0 or 1 in the training dataset and tested on all the images with label 0 or 1 in the testing dataset. Therefore, the training and testing sets are subsets of the training and testing sets of MNIST. There are 5923 handwritten-zero images and 6742 handwritten-one images in the training set. For testing set, there are 980 handwritten-zero images and 1135 handwritten-one images.

The loss function of the logistic regression model used is

$$f(\boldsymbol{w}) = \sum_{i=1}^{n} \log(1 + \exp(-y_i \boldsymbol{w}^T \boldsymbol{x}_i)), \qquad (7)$$
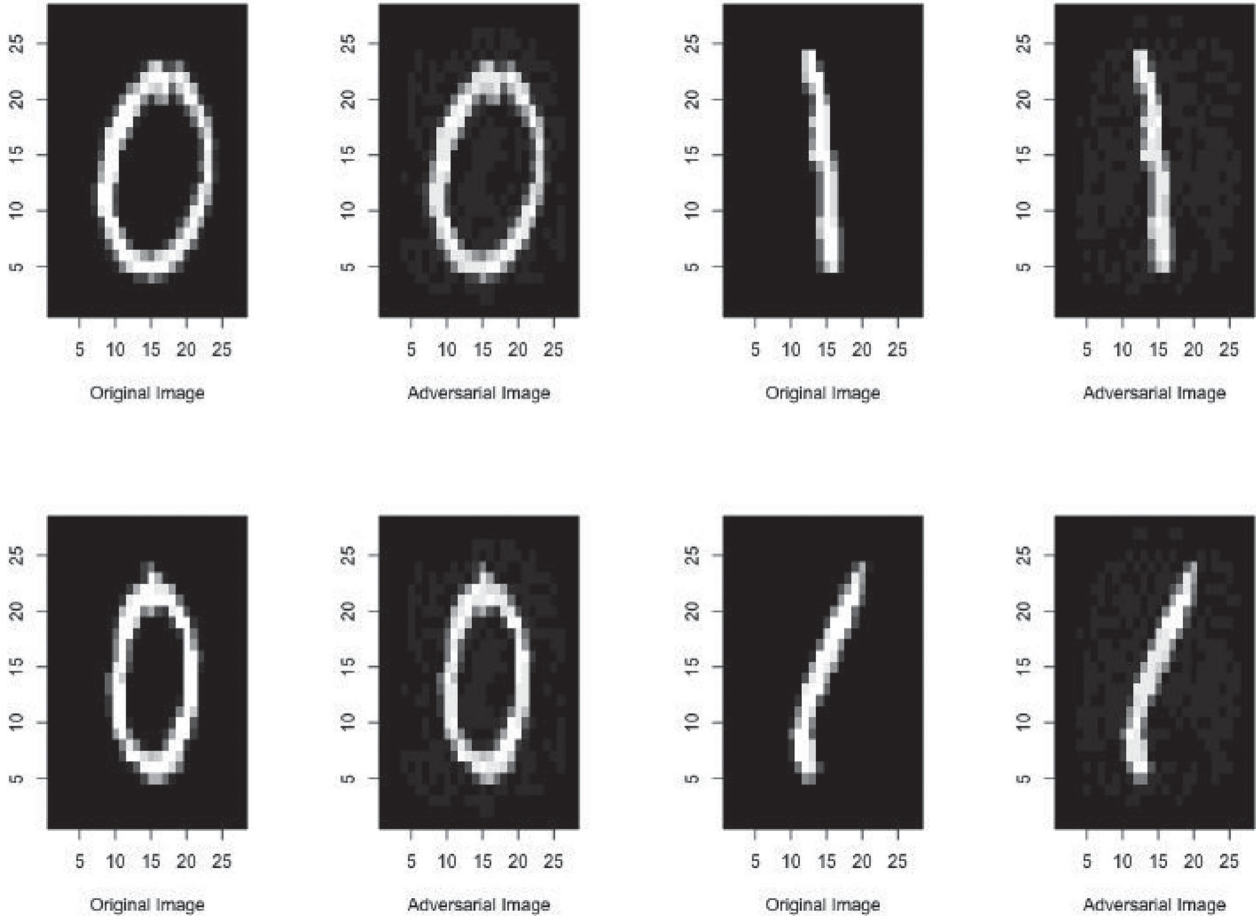
**Figure 8.** Four examples of attack results on a logistic regression model. In each pair of images, the left shows the original handwritten image and the right shows the adversarial image, which is classified incorrectly by the logistic regression model.

where $w \in \mathbb{R}^{784}$ is the weight, $x_1, x_2, \ldots, x_n \in \mathbb{R}^{784}$ are input images vectorized from the original $28 \times 28$ pixel space dimension, and $y_1, y_2, \ldots, y_n \in \{-1, 1\}$ are labels with $-1$ corresponding to handwritten-zero and 1 corresponding to handwritten-one.

The projected gradient descent (PGD) method attacks the model using gradient information. The gradient of the loss w.r.t. a certain input image $x_i$ is:

$$\nabla f(x_i) = -\frac{y_i}{1 + \exp(y_i w^T x_i)} w.$$

PGD attacks the model by repeating the following updating equation:

$$x^{t+1} = \Pi_\epsilon \left\{ x^t + \alpha \cdot \text{sign}\left(\nabla f(x_i)\right), x_i \right\}.$$

In our experiment, we set $\alpha = 1$, $\epsilon = 8$, which reflects 0.031 in $[0, 1]$ scale since $\frac{8}{256} \approx 0.031$, and run the updates for 40 steps.

The prediction accuracy under no attack is 99.95%. Under PGD attack, the prediction accuracy drops to 72.96%. In Figure 8, we visualize four attack results, where in each pair the image on the left side is the original image and the image on the right side is the adversarial example based on the left one. We observe that the adversarial examples look remarkably similar to the original images but the predictions on these examples are altered by some invisible adversarial noise.

## 5.2. Comparison of Different Attack Methods

In this section we compare the performances of several attack methods including gradient-based, score-based and decision-based. The gradient-based attacks are compared on all samples from the test sets of the two benchmark datasets described above. Due to the slow speed of generating adversarial examples with score-based and decision-based approaches, the corresponding methods are evaluated on 300 random samples from each of the test sets.

For the dataset MNIST, we train a network consisting of two convolutional layers with, respectively, 20 and 50 filters with kernel size 5, each followed by $2 \times 2$ max-pooling, and a fully connected layer of size 500. For CIFAR10, we use the standard VGG-16 structure (Simonyan and Zisserman 2014) with batch normalization.

The attack methods tested are:

- Gradient-based: FGSM (Goodfellow, Shlens, and Szegedy 2015) and PGD (Madry et al. 2018).
- Score-based: ZOO (Chen et al. 2017) and Square attack (Andriushchenko et al. 2020).
- Decision-based: Boundary attack (Brendel, Rauber, and Bethge 2018), OPT attack (Cheng et al. 2019a) and Sign-OPT (Cheng et al. 2020a).

**Table 1.** Accuracy on MNIST and CIFAR10 under gradient-based attack.

| Method | MNIST | | | | CIFAR10 | | | |
|---|---|---|---|---|---|---|---|---|
| | Clean Acc | $\epsilon = 0.1$ | $\epsilon = 0.2$ | $\epsilon = 0.3$ | Clean Acc | $\epsilon = 0.01$ | $\epsilon = 0.02$ | $\epsilon = 0.03$ |
| FGSM | 98.58% | 75.98% | 17.66% | 2.63% | 93.34% | 56.46% | 50.85% | 47.61% |
| PGD | 98.58% | 69.42% | 6.47% | 0.02% | 93.34% | 5.71% | 0.12% | 0.01% |

**Table 2.** Accuracy on MNIST and CIFAR10 under score-based attack.

| | Method | Clean Acc | Max.iter= 100 | Max.iter= 200 | Max.iter= 500 |
|---|---|---|---|---|---|
| MNIST | ZOO | 99.33% | 99.33% | 99.33% | 99.00% |
| | Square | 99.33% | 73.33% | 34.00% | 16.67% |
| CIFAR10 | ZOO | 93.67% | 93.67% | 90.33% | 76.67% |
| | Square | 93.67% | 84.67% | 70.67% | 54.33% |

### 5.2.1. Results on Gradient-Based Attack

The results of attacking the two classification models on MNIST and CIFAR10 with gradient-based methods are shown in Table 1. The strength of attack is represented by $\epsilon$, which controls the maximum distortion level. The larger the value of $\epsilon$, the stronger the attack. It is easy to see that PGD is stronger than FGSM since PGD runs the optimization process with Equation (3) multiple times to search for adversarial examples, while FGSM runs the optimization for only one step.

### 5.2.2. Results on Score-Based Attack

The results of score-based attacks on MNIST and CIFAR10 are shown in Table 2. For ZOO (Chen et al. 2017), the maximum number of iterations (Max.iter) limits the number of searches used to perform gradient estimation. The larger the value, the better the approximation. Therefore, a large quantity of maximum number of iterations usually results in better performance. Similarly, for Square attack (Andriushchenko et al. 2020), the maximum number of iterations controls the number of random-walk steps to search for adversarial examples. Therefore, a large value usually leads to better performance. The authors of Square attack studied the property of adversarial distortion and leveraged it to perform the attack. The method achieves better performances with the same maximum number of iterations when compared to the ZOO.

### 5.2.3. Results on Decision-Based Attack

The results of decision-based experiments, in which only the predicted labels are available to the attacks, are shown in Figure 9. Under this setting, only limited information is available, making it harder to craft adversarial examples. In Figure 9, we compare Boundary (Brendel, Rauber, and Bethge 2018), OPT attack (Cheng et al. 2019a) and Sign-OPT (Cheng et al. 2020a) by visualizing the relationship between the number of queries required versus distortion. Boundary attack is an example of random-walk based attacks and the other two are examples of optimization based attacks. The horizontal axis of each figure represents number of times the victim model is called to generate adversarial example. The vertical axis represents distortion level; that is, the $\ell_2$ norm Euclidean distance between adversarial example and the original input. We observe that, under the same distortion level, Sign-OPT and OPT attacks require much fewer number of searches to craft

adversarial examples than Boundary attack, and Sign-OPT is more efficient than OPT attack.

### 5.3. Comparison of Different Defense Methods

We first compare defense methods from the first three categories (adversarial training, randomization and projection) and delay the last category (detection) toward the end of this section. It is because detection methods are based on a different philosophy when comparing with the first three categories: they aim to detect the adversarial examples rather than to classify them correctly. In other words, they are not directly comparable with other defense methods. As before, we compare these methods with the same two benchmark datasets.

The following methods from the first three categories are selected for comparison:

- No defense: Model without any defense method.
- Adversarial training: Madry's Adv (Madry et al. 2018) and TRADES (Zhang et al. 2019b).
- Randomization: RSE (Liu et al. 2018b) and Ran.Smooth (Cohen, Rosenfeld, and Kolter 2019).
- Projection: ER-CLA (Li et al. 2018b).

The comparison results of the first three categories of defense methods are shown in Figure 10. In the plot, the horizontal axis represents $\epsilon$, the strength of the PGD attack. The larger the $\epsilon$, the stronger the attack. The vertical axis represents the testing accuracy under the PGD attack. For all models, the accuracy drops as the attack strength increases except for Ran.Smooth on MNIST when the attack strength increases from 0 to 0.05. We observe that TRADES and Ran.Smooth perform better on MNIST and CIFAR10. On CIFAR10, RSE and Madry's adversarial training perform similarly, while TRADES performs best when attack is strong. Another point to notice is that, when there is no attack, all the other methods perform worse than ER-Classifier. In fact, ER-Classifier and adversarial training methods can be combined together to further improve the results. In Li et al. (2018b), the authors showed that combing ER-Classifier with adversarial training methods outperform both Madry's adversarial training and TRADES. All five defense methods perform much better than the model without any defense method.

Now we compare defense methods from the detection category. The methods are:

- KD: Kernel Density detection (Feinman et al. 2017).
- LID: Local Ontrinsic Dimensionality detection (Ma et al. 2018).
- ODD: Odds are odd detection (Roth, Kilcher, and Hofmann 2019).
- ReBeL: Reading Between the Layers (Raghuram et al. 2020).

We report area under the curve (AUC) of the ROC curve as the performance evaluation criterion, as well as the true positive rates (TPR) by thresholding false positive rates (FPR) at 0.01, 0.05, and 0.1, as it is practical to keep misclassified natural data at a low proportion. Note that TPR represents the proportion of adversarial examples correctly classified as adversarial, while FPR represents the proportion of natural data incorrectly misclassified as adversarial. Before calculating the perfor-
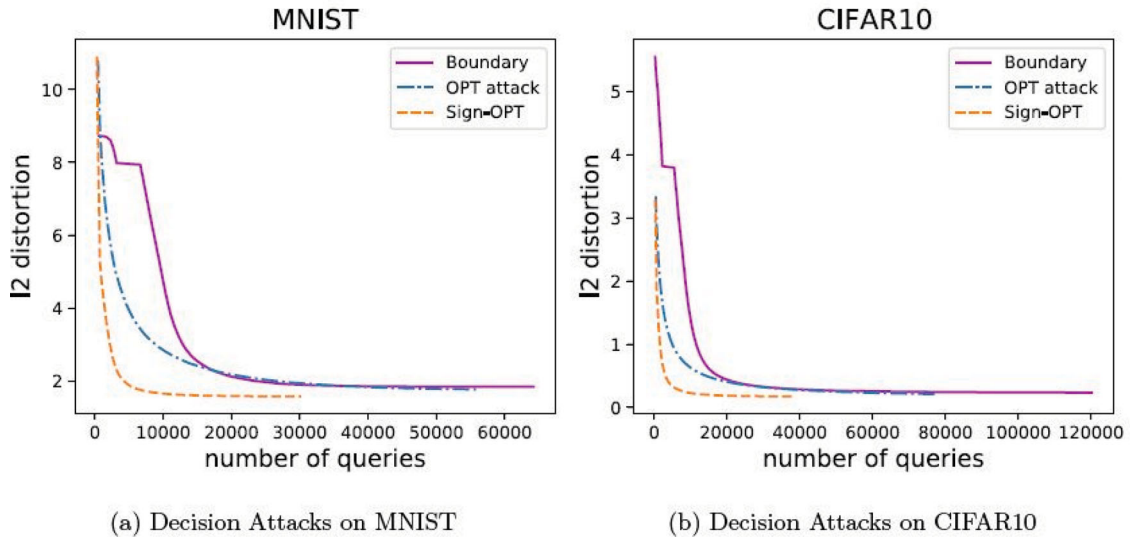
(a) Decision Attacks on MNIST  (b) Decision Attacks on CIFAR10

**Figure 9.** Performances of decision-based attacks. Number of queries represents the number of times victim model is called.
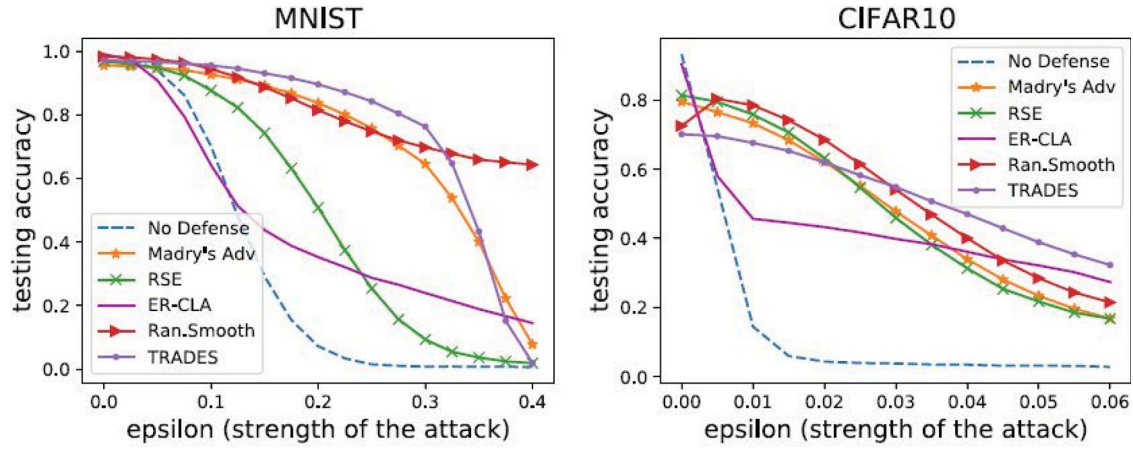


**Figure 10.** Comparison of Different Defense Methods on MNIST and CIFAR10.

**Table 3.** Performance of detection methods against strong attacks.

| Data | Metric | C&W | | | | FGSM | | | | PGD | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | KD | LID | ODD | ReBeL | KD | LID | ODD | ReBeL | KD | LID | ODD | ReBeL |
| CIFAR10 | AUC | 0.945 | 0.947 | 0.955 | **0.968** | 0.873 | 0.957 | 0.968 | **0.990** | 0.791 | 0.777 | **0.963** | 0.962 |
| | TPR(FPR@0.01) | 0.068 | 0.220 | **0.591** | 0.309 | 0.136 | 0.385 | 0.224 | **0.698** | 0.018 | 0.093 | 0.059 | **0.191** |
| | TPR(FPR@0.05) | 0.464 | 0.668 | **0.839** | 0.726 | 0.401 | 0.753 | 0.709 | **0.974** | 0.148 | 0.317 | **0.819** | 0.789 |
| | TPR(FPR@0.10) | 0.911 | 0.856 | 0.901 | **0.954** | 0.572 | 0.875 | **1.000** | **1.000** | 0.285 | 0.448 | **0.999** | **0.999** |
| MNIST | AUC | 0.932 | 0.785 | 0.968 | **0.980** | 0.933 | 0.888 | 0.952 | **0.992** | 0.801 | 0.861 | 0.967 | **0.975** |
| | TPR(FPR@0.01) | 0.196 | 0.079 | 0.212 | **0.630** | 0.421 | 0.152 | **0.898** | 0.885 | 0.062 | 0.170 | **0.607** | 0.382 |
| | TPR(FPR@0.05) | 0.616 | 0.263 | **0.911** | 0.900 | 0.692 | 0.503 | 0.908 | **0.990** | 0.275 | 0.396 | **0.934** | 0.851 |
| | TPR(FPR@0.10) | 0.818 | 0.397 | **1.000** | 0.972 | 0.796 | 0.678 | 0.917 | **1.000** | 0.429 | 0.552 | 0.945 | **0.956** |

NOTE: Best results are in bolded.

mance metrics, all the samples that can be classified correctly by the model are removed. The results are reported in Table 3. ODD (Roth, Kilcher, and Hofmann 2019) and ReBeL (Raghuram et al. 2020) show superior or comparable performance over the other detection methods across the two benchmark datasets.

## 6. Concluding Remarks

In this article, we described the attack and defense problems in classification. We provided a taxonomy of these methods,

explained the main ideas behind them, and numerically compared their performances. As mentioned before, it is our hope that this article can spark statisticians' interest in this exciting area of research, as we firmly believe that statisticians are well suited to provide unique and significant contributions to the field. One successful example is the EAD attack of Chen et al. (2018), where the elastic-net penalty of Zou and Hastie (2005) was a key component of the method. Another successful example is the ER-Classifer of Li et al. (2018b), where the method of Levina and Bickel (2005) was used to estimate the intrinsic

dimension of the data, and achieve dimension reduction. We conclude this article with some future work directions that are particularly suited for statisticians to tackle.

### 6.1. Statistical Properties of Adversarial Examples

It is important and useful to study the statistical properties of adversarial examples, as these can provide useful insights and guidelines for building stronger attack and defense methods. So far work in this area is rather sparse. Some notable exceptions include the study of the hidden layer output distribution of adversarial examples (Lee et al. 2018), the variance of the hidden output (Raghuram et al. 2020), and the input space feature attribution (Yang et al. 2020a).

### 6.2. Efficient Modeling for Improved Decision-Based Attacks

In reality, many classification systems are blackboxes to the outsiders, and therefore gradient-based and score-based attack methods cannot be applied. Also, these classification systems typically limit the number of queries any outsider can make, and hence many of the current decision-based attack methods will fail. The problem of developing a strong decision-based attack method can be viewed as estimating (or modeling) a black-box classification model with a limited number of carefully chosen "design points" (most likely in a sequential manner) as input to the classifier.

### 6.3. Theoretical Guarantee for Robust Defense Methods

With the ever increasing size and complexity of datasets and applications, there will always be a demand for more reliable and efficient defense methods that are scalable. So far the trend is that many defense methods were shown to have excellent empirical properties at the time of their creation, but then very often they were soon beaten by newer attack methods (Athalye, Carlini, and Wagner 2018). This suggests that defense methods cannot be blindly trusted. One ambitious project is to formally define the notion of robustness in this attack and defense context, which includes attaching a robust score to any defense method; that is, the higher the score, the more reliable the method. This is similar to the use of breakdown point in classical robust statistics. It is likely that there will be different reasonable definitions for robustness, each designed for one type of attacks. If a new method achieves high scores for different robustness definitions, then we have strong confidence that this method will work well.

### Acknowledgments

### References

Al-Dujaili, A., and O'Reilly, U.-M. (2020), "Sign Bits Are All You Need for Black-Box Attacks," in *International Conference on Learning Representations*. [6]

Alzantot, M., Sharma, Y., Chakraborty, S., and Srivastava, M. (2018), "Genattack: Practical Black-Box Attacks With Gradient-Free Optimization," arXiv preprint arXiv:1805.11090. [6]

Andriushchenko, M., Croce, F., Flammarion, N., and Hein, M. (2020), "Square Attack: A Query-Efficient Black-Box Adversarial Attack via Random Search," in *European Conference on Computer Vision*, 484–501. [6,12,13]

Athalye, A., Carlini, N., and Wagner, D. (2018), "Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples," in *International Conference on Machine Learning*, 274–283. [4,8,15]

Baluja, S., and Fischer, I. (2018), "Learning to Attack: Adversarial Transformation Networks," in *Proceedings of AAAI-2018*. [5]

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., and Roli, F. (2013), "Evasion Attacks Against Machine Learning at Test Time," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 387–402.

Brendel, W., Rauber, J., and Bethge, M. (2018), "Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models," *International Conference on Learning Representations*. [6,7,12,13]

Brunner, T., Diehl, F., Le, M. T., and Knoll, A. (2019), "Guessing Smart: Biased Sampling for Efficient Black-Box Adversarial Attacks," in *Proceedings of the IEEE International Conference on Computer Vision*, 4958–4966. [6,7]

Carlini, N., and Wagner, D., (2017a), "Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 3–14. [5]

——— (2017b), "Towards Evaluating the Robustness of Neural Networks," in *2017 IEEE Symposium on Security and Privacy (SP)*, 39–57. [4,8]

Carmon, Y., Raghunathan, A., Schmidt, L., Liang, P., and Duchi, J. C. (2020), "Unlabeled Data Improves Adversarial Robustness," *Advances in Neural Information Processing Systems*. [8]

Chen, H., Zhang, H., Si, S., Li, Y., Boning, D., and Hsieh, C.-J. (2019), "Robustness Verification of Tree-Based Models," in *Advances in Neural Information Processing Systems*, Vol. 32, Curran Associates, Inc. [1]

Chen, J., and Gu, Q. (2020), "Rays: A Ray Searching Method for Hard-Label Adversarial Attack," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1739–1747. [7]

Chen, J., and Jordan, M. I. (2019), "Boundary Attack++: Query-Efficient Decision-Based Adversarial Attack," arXiv preprint arXiv:1904.02144. [6,7]

Chen, J., Jordan, M. I., and Wainwright, M. J. (2020), "Hopskipjumpattack: A Query-Efficient Decision-Based Attack." In *2020 IEEE Symposium on Security and Privacy (SP)*, 1277–1294. [6,7]

Chen, P.-Y., Sharma, Y., Zhang, H., Yi, J., and Hsieh, C.-J. (2018) "EAD: Elastic-Net Attacks to Deep Neural Networks via Adversarial Examples," in *Thirty-Second AAAI Conference on Artificial Intelligence*. [5,14]

Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., and Hsieh, C.-J. (2017), "Zoo: Zeroth Order Optimization Based Black-Box Attacks to Deep Neural Networks Without Training Substitute Models," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 15–26. [2,6,12,13]

Cheng, M., Le, T., Chen, P.-Y., Zhang, H., Yi, J., and Hsieh, C.-J. (2019a), "Query-Efficient Hard-Label Black-Box Attack: An Optimization-Based Approach," in *International Conference on Learning Representations*, available at *https://openreview.net/forum?id=rJlk6iRqKX*. [6,7,12,13]

Cheng, M., Singh, S., Chen, P. H., Chen, P.-Y., Liu, S., and Hsieh, C.-J. (2020a) "Sign-Opt: A Query-Efficient Hard-Label Adversarial Attack," in *International Conference on Learning Representations*. [6,7,12,13]

Cheng, M., Yi, J., Chen, P.-Y., Zhang, H., and Hsieh, C.-J. (2020b), "Seq2sick: Evaluating the Robustness of Sequence-to-Sequence Models with Adversarial Examples," *Proceedings of the AAAI Conference on Artificial Intelligence*, 34, 3601–3608. [1]

Cheng, S., Dong, Y., Pang, T., Su, H., and Zhu, J. (2019b), "Improving Black-Box Adversarial Attacks with a Transfer-Based Prior," in *Advances in Neural Information Processing Systems*, 10932–10942. [7]

Cohen, J., Rosenfeld, E., and Kolter, Z. (2019), "Certified Adversarial Robustness via Randomized Smoothing," in *International Conference on Machine Learning*, 1310–1320. [8,9,13]

Croce, F., and Hein, M. (2020), "Reliable Evaluation of Adversarial Robustness with an Ensemble of Diverse Parameter-Free Attacks," in *International Conference on Machine Learning*, 2206–2216. [4,5]

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009), "Imagenet: A Large-Scale Hierarchical Image Database," in *IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009*, 248–255. [6]

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019), "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Minneapolis, MN: Association for Computational Linguistics. [1]

Dhillon, G. S., Azizzadenesheli, K., Lipton, Z. C., Bernstein, J., Kossaifi, J., Khanna, A., and Anandkumar, A. (2018), "Stochastic Activation Pruning for Robust Adversarial Defense," *International Conference on Learning Representations*. [8]

Ding, G. W., Sharma, Y., Lui, K. Y. C., and Huang, R. (2020), "Mma Training: Direct Input Space Margin Maximization Through Adversarial Training," in *International Conference on Learning Representations*. [8]

Dodge, S., and Karam, L. (2017), "A Study and Comparison of Human and Deep Learning Recognition Performance Under Visual Distortions," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, 1–7. [8]

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2021) "An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale," *International Conference on Learning Representations*. [3]

Fawzi, A., Moosavi-Dezfooli, S.-M., and Frossard, P. (2016), "Robustness of Classifiers: From Adversarial to Random Noise," in *Advances in Neural Information Processing Systems*, 1632–1640. [8]

Feinman, R., Curtin, R. R., Shintre, S., and Gardner, A. B. (2017), "Detecting Adversarial Samples from Artifacts," *International Conference on Machine Learning*. [10,13]

Franceschi, J.-Y., Fawzi, A., and Fawzi, O. (2018), "Robustness of Classifiers to Uniform $\ell_p$ and Gaussian Noise," in *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics* (eds. A. Storkey and F. Perez-Cruz), vol. 84 of *Proceedings of Machine Learning Research*, 1280–1288. PMLR. [8]

Gao, J., Lanchantin, J., Soffa, M. L., and Qi, Y. (2018), "Black-Box Generation of Adversarial Text Sequences to Evade Deep Learning Classifiers," in *2018 IEEE Security and Privacy Workshops (SPW)*, 50–56. [1]

Gao, R., Cai, T., Li, H., Hsieh, C.-J., Wang, L., and Lee, J. D. (2019), "Convergence of Adversarial Training in Overparametrized Neural Networks," *Advances in Neural Information Processing Systems*, 32, 13029–13040. [8]

Gong, Z., Wang, W., and Ku, W.-S. (2017), "Adversarial and Clean Data Are Not Twins," arXiv preprint arXiv:1704.04960. [10]

Goodfellow, I., Bengio, Y., and Courville, A. (2016), *Deep Learning*. Cambridge, MA: MIT Press. [2]

Goodfellow, I., Shlens, J., and Szegedy, C. (2015), "Explaining and Harnessing Adversarial Examples," in *International Conference on Learning Representations*. [1,4,5,8,9,12]

Gowal, S., Qin, C., Uesato, J., Mann, T., and Kohli, P. (2020), "Uncovering the Limits of Adversarial Training Against Norm-Bounded Adversarial Examples," arXiv preprint arXiv:2010.03593. [8]

Grosse, K., Manoharan, P., Papernot, N., Backes, M., and McDaniel, P. (2017), "On the (Statistical) Detection of Adversarial Examples," arXiv preprint arXiv:1702.06280. [10]

Guo, C., Frank, J. S., and Weinberger, K. Q. (2020), "Low Frequency Adversarial Perturbation," in *Proceedings of the 35th Uncertainty in Artificial Intelligence Conference* (eds. R. P. Adams and V. Gogate), vol. 115 of *Proceedings of Machine Learning Research*, 1127–1137. Tel Aviv, Israel: PMLR. [6,7]

Guo, C., Gardner, J., You, Y., Wilson, A. G., and Weinberger, K. (2019a), "Simple Black-Box Adversarial Attacks," in *Proceedings of the 36th International Conference on Machine Learning*, eds. K. Chaudhuri and R. Salakhutdinov, vol. 97 of *Proceedings of Machine Learning Research*, 2484–2493. Tel Aviv, Israel: PMLR. [6]

Guo, Y., Yan, Z., and Zhang, C. (2019b), "Subspace Attack: Exploiting Promising Subspaces for Query-Efficient Black-Box Attacks," in *Advances in Neural Information Processing Systems*, eds. H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, vol. 32, Curran Associates, Inc. [6]

Hecht-Nielsen, R. (1992), "Theory of the Backpropagation Neural Network," in *Neural Networks for Perception*, 65–93. Elsevier. [3]

Hung, K., and Fithian, W. (2019), "Rank Verification for Exponential Families," *The Annals of Statistics*, 47, 758–782. [9]

Ilyas, A., Engstrom, L., Athalye, A., and Lin, J. (2018), "Black-Box Adversarial Attacks with Limited Queries and Information," in *International Conference on Machine Learning*, 2137–2146. [6]

Ilyas, A., Engstrom, L., and Madry, A. (2019), "Prior Convictions: Black-Box Adversarial Attacks with Bandits and Priors," in *International Conference on Learning Representations*. [6]

Jalal, A., Ilyas, A., Daskalakis, C., and Dimakis, A. G. (2017), "The Robust Manifold Defense: Adversarial Training Using Generative Models," arXiv preprint arXiv:1712.09196. [9]

Jia, R., and Liang, P. (2017), "Adversarial Examples for Evaluating Reading Comprehension Systems," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2021–2031. Copenhagen, Denmark: Association for Computational Linguistics. [1]

Krizhevsky, A., and Hinton, G. (2009), "Learning Multiple Layers of Features from Tiny Images," Technical Report, Citeseer. [3,11]

Kurakin, A., Goodfellow, I., and Bengio, S. (2016), "Adversarial Examples in the Physical World." [4,5,8]

LeCun, Y. (1998), "The MNIST Database of Handwritten Digits," *http://yann.lecun.com/exdb/mnist/*. [11]

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998), "Gradient-Based Learning Applied to Document Recognition," in *Proceedings of the IEEE*, 86, 2278–2324. [2]

Lecuyer, M., Atlidakis, V., Geambasu, R., Hsu, D., and Jana, S. (2019), "Certified Robustness to Adversarial Examples with Differential Privacy," in *2019 IEEE Symposium on Security and Privacy (SP)*, 656–672. [9]

Lee, K., Lee, K., Lee, H., and Shin, J. (2018), "A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks," in *Advances in Neural Information Processing Systems*, 7167–7177. [10,11,15]

Levina, E., and Bickel, P. J. (2005), "Maximum Likelihood Estimation of Intrinsic Dimension," in *Advances in Neural Information Processing Systems*, 777–784. [14]

Li, B., Chen, C., Wang, W., and Carin, L. (2018a), "Certified Adversarial Robustness with Additive Noise," arXiv preprint arXiv:1809.03113. [9]

Li, X., and Li, F. (2017), "Adversarial Examples Detection in Deep Networks with Convolutional Filter Statistics," in *2017 IEEE International Conference on Computer Vision*, 5775–5783. [10]

Li, Y., Li, L., Wang, L., Zhang, T., and Gong, B. (2019), "Nattack: Learning the Distributions of Adversarial Examples for an Improved Black-Box Attack on Deep Neural Networks," in *International Conference on Machine Learning*, 3866–3876. PMLR. [6]

Li, Y., Min, M. R., Yu, W., Hsieh, C.-J., Lee, T. C. M., and Kruus, E. (2018b), "Optimal Transport Classifier: Defending Against Adversarial Attacks by Regularized Deep Embedding," arXiv preprint arXiv:1811.07950. [9,10,13,14]

Liu, S., Chen, P.-Y., Chen, X., and Hong, M. (2018a), "signsgd via Zeroth-Order Oracle," in *International Conference on Learning Representations*. [6]

Liu, X., Cheng, M., Zhang, H., and Hsieh, C.-J. (2018b), "Towards Robust Neural Networks via Random Self-Ensemble," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 369–385. [8,13]

Liu, X., Li, Y., Wu, C., and Hsieh, C.-J. (2019), "Adv-BNN: Improved Adversarial Defense Through Robust Bayesian Neural Network," in *International Conference on Learning Representations*. [8,9]

Liu, Y., Chen, X., Liu, C., and Song, D. (2017), "Delving into Transferable Adversarial Examples and Black-Box Attacks," *International Conference on Learning Representations*. [6]

Ma, X., Li, B., Wang, Y., Erfani, S. M., Wijewickrema, S., Schoenebeck, G., Song, D., Houle, M. E., and Bailey, J. (2018), "Characterizing Adversarial Subspaces Using Local Intrinsic Dimensionality," *International Conference on Learning Representations*. [10,11,13]

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2018), "Towards Deep Learning Models Resistant to Adversarial Attacks," in *International Conference on Learning Representations*. [4,5,8,12,13]

Meng, D., and Chen, H. (2017), "Magnet: A Two-Pronged Defense Against Adversarial Examples," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 135–147. [9]

Metzen, J. H., Genewein, T., Fischer, V., and Bischoff, B. (2017), "On Detecting Adversarial Perturbations," *International Conference on Learning Representations*. [10]

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. (2016), "Deepfool: A Simple and Accurate Method to Fool Deep Neural Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2574–2582. [5]

Mopuri, K. R., Ojha, U., Garg, U., and Babu, R. V. (2018), "Nag: Network for Adversary Generation," in *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*. [5]

Nesterov, Y., and Spokoiny, V. (2017), "Random Gradient-Free Minimization of Convex Functions," *Foundations of Computational Mathematics*, **17**, 527–566. [7]

Pang, T., Yang, X., Dong, Y., Su, H., and Zhu, J. (2021), "Bag of Tricks for Adversarial Training," in *International Conference on Learning Representations*. [8]

Pang, T., Yang, X., Dong, Y., Xu, K., Zhu, J., and Su, H. (2020), "Boosting Adversarial Training with Hypersphere Embedding," in *Advances in Neural Information Processing Systems*, eds. H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan and H. Lin, vol. 33, 7779–7792. Curran Associates, Inc. [8]

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. (2017), "Practical Black-Box Attacks Against Machine Learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 506–519. [3,6]

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. (2016), "The Limitations of Deep Learning in Adversarial Settings," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, 372–387. [5]

Qiu, S., Liu, Q., Zhou, S., and Wu, C. (2019), "Review of Artificial Intelligence Adversarial Attack and Defense Technologies," *Applied Sciences*, 9, 909. [2]

Raghuram, J., Chandrasekaran, V., Jha, S., and Banerjee, S. (2020), "Detecting Anomalous Inputs to DNN Classifiers by Joint Statistical Testing at the Layers," arXiv preprint arXiv:2007.15147. [10,11,13,14,15]

Ren, K., Zheng, T., Qin, Z., and Liu, X. (2020), "Adversarial Attacks and Defenses in Deep Learning," *Engineering*, 6, 346–360. [2]

Roth, K., Kilcher, Y., and Hofmann, T. (2019), "The Odds Are Odd: A Statistical Test for Detecting Adversarial Examples," in *International Conference on Machine Learning*, 5498–5507. [10,13,14]

Samangouei, P., Kabkab, M., and Chellappa, R. (2018), "Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models," in *International Conference on Learning Representations*. [9]

Samanta, S., and Mehta, S. (2017), "Towards Crafting Text Adversarial Samples," arXiv preprint arXiv:1707.02812. [1]

Serban, A., Poll, E., and Visser, J. (2020), "Adversarial Examples on Object Recognition: A Comprehensive Survey," *ACM Computing Surveys (CSUR)*, 53, 1–38. [1]

Shafahi, A., Najibi, M., Ghiasi, M. A., Xu, Z., Dickerson, J., Studer, C., Davis, L. S., Taylor, G., and Goldstein, T. (2019), "Adversarial Training for Free!," in *Advances in Neural Information Processing Systems*, 3353–3364. [8]

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., and Lanctot, M. (2016) "Mastering the Game of Go With Deep Neural Networks and Tree Search," *Nature*, 529, 484–489. [1]

Simonyan, K., and Zisserman, A. (2014), "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv preprint arXiv:1409.1556. [1,12]

Song, Y., Kim, T., Nowozin, S., Ermon, S., and Kushman, N. (2018), "Pixeldefend: Leveraging Generative Models to Understand and Defend Against Adversarial Examples," in *International Conference on Learning Representations*. [9]

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014), "Intriguing Properties of Neural Networks," in *International Conference on Learning Representations*. [1]

Tanay, T., and Griffin, L. (2016), "A Boundary Tilting Perspective on the Phenomenon of Adversarial Examples," arXiv preprint arXiv:1608.07690. [10]

Wang, L., Zhang, H., Yi, J., Hsieh, C.-J., and Jiang, Y. (2020a), "Spanning Attack: Reinforce Black-Box Attacks with Unlabeled Data," *Machine Learning*, 109, 2349–2368. [6]

Wang, Y., Ma, X., Bailey, J., Yi, J., Zhou, B., and Gu, Q. (2019), "On the Convergence and Robustness of Adversarial Training," in *International Conference on Machine Learning*, 6586–6595. [8]

Wang, Y., Zou, D., Yi, J., Bailey, J., Ma, X., and Gu, Q. (2020b), "Improving Adversarial Robustness Requires Revisiting Misclassified Examples," in *International Conference on Learning Representations*. [8]

Wong, E., and Kolter, J. Z. (2020), "Learning Perturbation Sets for Robust Machine Learning," arXiv preprint arXiv:2007.08450. [5]

Wong, E., Rice, L., and Kolter, J. Z. (2020), "Fast Is Better than Free: Revisiting Adversarial Training," in *International Conference on Learning Representations*. [8]

Wong, E., Schmidt, F., and Kolter, Z. (2019), "Wasserstein Adversarial Examples via Projected Sinkhorn Iterations," in *International Conference on Machine Learning*, 6808–6817. [5]

Wu, D., Xia, S.-T., and Wang, Y. (2020), "Adversarial Weight Perturbation Helps Robust Generalization," *Advances in Neural Information Processing Systems*, 33. [8]

Xie, C., Wang, J., Zhang, Z., Ren, Z., and Yuille, A. (2018), "Mitigating Adversarial Effects Through Randomization," in *International Conference on Learning Representations*. [8]

Xu, H., Ma, Y., Liu, H.-C., Deb, D., Liu, H., Tang, J.-L., and Jain, A. K. (2020), "Adversarial Attacks and Defenses in Images, Graphs and Text: A Review," *International Journal of Automation and Computing*, 17, 151–178. [2]

Yang, P., Chen, J., Hsieh, C.-J., Wang, J.-L., and Jordan, M. (2020a), "Mlloo: Detecting Adversarial Examples with Feature Attribution," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 34, 6639–6647. [4,10,15]

Yang, P., Chen, J., Hsieh, C.-J., Wang, J.-L., and Jordan, M. I. (2020b), "Greedy Attack and Gumbel Attack: Generating Adversarial Examples for Discrete Data," *Journal of Machine Learning Research*, 21, 1–36. [1]

Yuan, X., He, P., Zhu, Q., and Li, X. (2019), "Adversarial Examples: Attacks and Defenses for Deep Learning," *IEEE Transactions on Neural Networks and Learning Systems*, 30, 2805–2824. [2]

Zantedeschi, V., Nicolae, M.-I., and Rawat, A. (2017), "Efficient Defenses Against Adversarial Attacks," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 39–49. [8]

Zhang, D., Zhang, T., Lu, Y., Zhu, Z., and Dong, B. (2019a), "You Only Propagate Once: Accelerating Adversarial Training via Maximal Principle," in *Advances in Neural Information Processing Systems*, 227–238. [8]

Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., and Jordan, M. (2019b), "Theoretically Principled Trade-Off Between Robustness and Accuracy," in *International Conference on Machine Learning*, 7472–7482. [8,13]

Zheng, Z., and Hong, P. (2018), "Robust Detection of Adversarial Attacks by Modeling the Intrinsic Properties of Deep Neural Networks," in *Advances in Neural Information Processing Systems*, 7913–7922. [10]

Zou, H., and Hastie, T. (2005), "Regularization and Variable Selection via the Elastic Net," *Journal of the Royal Statistical Society*: Series B, 67, 301–320. [5,14]