Adversarial Learning With Cost-Sensitive Classes

Haojing Shen, Sihong Chen, Ran Wang, Member, IEEE, and Xizhao Wang, Fellow, IEEE

Abstract—It is necessary to improve the performance of some special classes or to particularly protect them from attacks in adversarial learning. This article proposes a framework combining cost-sensitive classification and adversarial learning together to train a model that can distinguish between the protected and unprotected classes, such that the protected classes are less vulnerable to adversarial examples. We find in this framework an interesting phenomenon during the training of deep neural networks, called the Min–Max property, that is, the absolute values of most parameters in the convolutional layer approach 0 while the absolute values of a few parameters are significantly larger, becoming bigger. Based on this Min–Max property which is formulated and analyzed in a view of random distribution, we further build a new defense model against adversarial examples for adversarial robustness improvement. An advantage of the built model is that it performs better than the standard one and can combine with adversarial training to achieve improved performance. It is experimentally confirmed that, regarding the average accuracy of all classes, our model is almost as same as the existing models when an attack does not occur and is better than the existing models when an attack occurs. Specifically, regarding the accuracy of protected classes, the proposed model is much better than the existing models when an attack occurs.

Index Terms—Adversarial examples, adversarial training, attack and defense, cost sensitive, robustness.

I. INTRODUCTION

R ECENT studies have shown that the deep neural network is relatively fragile [2], and it is vulnerable to adversarial examples that are generated by adding adversarial perturbations in clean examples. The methods to generate adversarial examples are called adversarial attacks [2]–[10]. Simultaneously, to defend adversarial attacks, recently, many works have proposed adversarial defenses [2], [3], [6], [11]–[15]. There is an arms race between adversarial attacks and adversarial defenses.

Adversarial training [2], [3], [6] is a simple and effective adversarial defense method. Many studies [3], [6] have shown that adversarial training can effectively defend against white-box attacks. However, it can only defend against a given attack method, which may fail for other stronger or unknown attacks. Meanwhile, it is found in some works that the adversarial training would step into the trap, called obfuscated gradients [11], [15]. The obfuscated gradients give a false sense that the model has a good adversarial robustness against the adversarial attack by limiting the attacker to calculate the gradient of the model. However, an attacker can successfully attack a model with adversarial training by using a transfer attack method, building a new approximation function, or using a nongradient attack.

There are many ways to improve adversarial robustness of the model besides adversarial training and some of them have not caused an obfuscated gradient. The work improving adversarial robustness can be divided into two types of strategies. One is detecting adversarial examples after deep neural networks are built, such as adversarial detecting [16] and network verification [17]. Other is to make deep neural networks more robust before adversaries generate adversarial examples, such as adversarial training and network distillation [12]. For the latter, combined with other fields or techniques, such as uncertainty and off-center technique, many works [18]–[20] construct a special loss function to improve the adversarial robustness.

The main purpose of the existing defense methods [2], [12] is to improve the model’s robustness for overall classes. These defense methods treat every class in the dataset equally, that is, the adversarial robustness of each class needs to be improved simultaneously. However, constructing such an ideal model, which improves the adversarial robustness of each class, is challenging. Actually, in practical applications, not all the classes are equally important in some tasks, that is, samples of certain classes are more important than those of others. For example, there are various denominations of dollar, such as U.S. $1, $2, $10, $20, $50, $100, and so on. Obviously, in the task of identifying dollar bills, we hope that the model can be more accurate of large denominations. As for the adversary, of course, they prefer to attack the large denomination of the bill to obtain the maximum profit. From this perspective, we hope to propose a method to particularly improve the adversarial robustness of more important classes.

Some related works can be retrieved from [21] and [22]. Zhang and Evans [21] pointed out that in practical application, the cost of adversarial transformation depends on specific classes. They combine cost-sensitive learning with robust learning [23] and advocate using the cost-sensitive robustness to measure the classifiers performance for some tasks,
where the adversarial transformation is more important than other transformations. They code the cost of each type of adversarial transformation into a cost matrix, combine it with the robust learning method [23], and propose a new objective function. However, they do not explain why cost-sensitive robustness learning affects the robustness of each class, and the performance of the model is heavily dependent on the robustness learning methods in [23]. Based on the optimal transmission theory and Wasserstein Distance, Terzi et al. [22] proposed the optimal cost of transferring from one kind of distribution to another to improve the model’s cost-sensitive robustness. Their proposed WPGD method can be used to solve the cost-sensitive problem, data imbalance problem, or the balance problem of robustness and accuracy.

In this article, we focus on a new problem, that is, protecting a particular class under adversarial attacks. Unlike traditional defense strategies, we consider the accuracy rate of specific categories to minimize the impact of adversarial examples. Of course, the accuracy rate of other categories may be sacrificed to some extent, but it is expected that the overall performance is similar to that of the standard training model. We find that this problem has one thing in common with the cost-sensitive problem [24], [25], that is, the cost of misclassification depends on different labels. This is different from the traditional classification problem, which assumes that all misclassification cases have the same cost. In real life, the misclassification cost of many problems is related to the individual categories. For example, in medical diagnosis, the cost of misdiagnosing a cancer patient as a flu patient is much higher than misdiagnosing a flu patient as a cancer patient. It is noteworthy that the traditional cost-sensitive learning, which is different with the proposed problem, does not consider the adversarial robustness of the model or particularly protect a certain class against adversarial attacks.

This article proposes a cost-sensitive adversarial learning model (CSA), which can well resist the adversarial attacks against special classes. We point out that the good robustness of the model is due to a special property of the convolutional layer parameters in the model, which is reflected in LeNet networks as the fact that the absolute values of most parameters in the convolutional layer approach 0 while the absolute values of a few parameters are significantly larger than others. This property indicates that the absolute values of a major part of weight parameters of the convolutional layer in the model attain the minimum (zero), and the absolute values of a minor part of weight parameters go to maximum. Thus, we name it the Min–Max property of weight parameters. Actually, the Min–Max property refers to a kind of approximate sparseness of weight parameters in convolutional layers, which reflects the essential of convolution from low-level to high-level features. Furthermore, when the model makes predictions for the samples of a specific class, only some of the parameters play a key role. Therefore, we explain why the CSA model could improve the adversarial robustness of special classes: the adversarial training brings the Min–Max property to the model parameters, while the cost matrix can locate the parameters that play a decisive role in the prediction results of the target class. When applying the cost matrix to adversarial training, it makes the model endowing part of the positioned parameters with the more obvious Min–Max property than other parameters during the training, thus improving the adversarial robustness of class the target class. According to this explanation, we propose to build a new learning model, called cost-sensitive adversarial extension (CSE), which does not depend on adversarial training. CSE is an end-to-end learning method that does not need adversarial training but can improve the adversarial robustness of the model. An overview of the proposed model is shown in Fig. 1.

The contributions of this article are listed as follows.
1) By incorporating the cost sensitivity into the adversarial training, we provide an algorithm (CSA), which can specifically protect an important class and improve its adversarial robustness.
2) We give a new explanation to the robustness of the model against adversarial attacks and propose a novel robust learning algorithm (CSE), which can make the model resist the adversarial examples without adversarial training. It is noted that most of the state-of-the-art models of adversarial robustness learning do need adversarial training.
limited in the measurable. But we usually use a small perturbation, which is easily fool machines into misclassification. This imperceptible adversarial example is indistinguishable to humans but can study adversarial example generation. The maximum perturbation. Some examples and corresponding adversarial examples as the following form: 

$$A(x) = \{x' | f(x') \neq f(x), ||x' - x||_p \leq \epsilon \}$$

where $A$ is a set, $x$ is a clean example, $x'$ is the adversarial example, $||x' - x||_p$ represents the size of perturbation, and $\epsilon$ is the maximum perturbation. Some examples and corresponding adversarial examples are represented in Fig. 2.

### A. Adversarial Attack

According to whether we can know the model’s structure and weights, the adversarial attacks can be divided into two types: 1) the white-box attack [3], [5], [6], [29] and 2) the black-box attack [30], [31]. In a white-box attack, the adversary knows the model structure and weights when attacking, such as FGSM [30], [31], DeepFool [5], PGD [6], and CW [7]. FGSM is a gradient-based and one-step attack method that updates along the direction of the pixel gradient’s signal function to obtain the adversarial examples. DeepFool looks for the minimum distance from a clean sample to an adversarial example and defines this distance as the model’s adversarial robustness. DeepFool takes advantage of a linear approximation method to generate adversarial examples iteratively. CW [7] is a nongradient-based adversarial attack, one of the most powerful attacks. Carlini and Wagner [7] proposed several objective functions to generate adversarial examples, among which the most effective attack method can effectively attack distillation defense [12]. From the perspective of robust optimization, Madry et al. [6] studied the model’s adversarial robustness, formulated the adversarial training process of the model as a saddle point problem, and used the projected gradient descent to search for more aggressive adversarial examples. They proved that the PGD method is the strongest attack method among the first-order attack methods.

In a black-box attack, the attacker knows nothing about the model except for the model’s output. Generally, one can attack these models by taking advantage of the transferability of adversarial examples [30], [32] or to generate adversarial examples with GAN [31]. The so-called transferability of adversarial examples means that adversarial examples of networks can attack neural networks with different structures, and even the two networks are trained on different datasets [32]. Liu et al. [30] studied the transferability of adversarial examples on the large-scale network model and the large-scale dataset. They find that the adversarial examples with the nontargeted attack are easy to generate, while the adversarial examples with targeted attack could hardly be transferred to the target label. A GAN-based algorithm is proposed [31] to generate adversarial examples. The combination of the image and text domains allows the generated adversarial examples to be more natural and interpretable, which helps understand the black-box model’s local behavior.

### B. Adversarial Training

Adversarial training is one of the most effective defense methods. Since the model is required to classify the adversarial examples correctly, the adversarial examples can be generated and added to the training set to retrain the model. The framework of adversarial training is shown in Fig. 1. Many defense works are based on this framework. For example, Goodfellow et al. [3] used FGSM to generate adversarial examples to retrain the model. They point out that the adversarial training procedure can be seen as minimizing the worst case error when data are perturbed by an adversary. After adversarial training, the adversarial robustness of the model is significantly improved. Some conclusions are drawn in [33]. However, both Goodfellow et al. [3] and Huang et al. [33] conduct their experiments only on the MNIST dataset. Some more complicated experiments on ImageNet are conducted by Kurakin et al. [34]. When they train their networks, they utilize adversarial examples and clean examples in each training step. They present that adversarial training increases deep neural networks’ robustness against one-step attacks but would not help under iterative attacks. To search more powerful attack method, Madry et al. [6] mentioned that adversarial training could be formulated as a minimum–maximization problem and proposed the PGD attack method based on the projected gradient descent. They proved that their method is the most powerful among the first-order attack methods. The model based on adversarial training with PGD obtains high performance against adversarial examples [15].

These days, many defense methods are based on the minimum-maximum optimization problem mentioned in [6]. Liu et al. [35] proposed a novel single-step adversarial training method, which can resist single-step and iterative adversarial examples. Zhang et al. [14] proposed TRADES,
the minimization algorithm of the strictest upper limit in theoretical probability distribution and measurable predictive variables, and won the first place in NeurIPS2018. Moreover, recently, some works [36]–[39] propose to reduce the computation of adversarial examples, such as Qin et al. [39] proposed a linear regularization to solve the problem that the computational cost of adversarial training grows prohibitively as the size of the model and number of input dimensions increase. Moreover, some works [11], [15] point out that adversarial training would cause obfuscated gradients. Athalye et al. [15] identified obfuscated gradients would cause a false sense of security in defenses against adversarial examples. According to this weakness, they successfully attack eight methods which were proposed in ICLR 2018.

C. Cost-Sensitive Learning

Cost-sensitive learning [24], [40], [41] is a common algorithm to solve unequal misclassification or unbalanced data learning. The main idea is: although a model can improve the overall performance, some problems bring worse consequences than others. In other words, misclassification of different problems may bring different levels of consequences. If an important performance cannot be guaranteed by only considering the overall performance, it may bring unimaginable bad effects. For example, in medical practice, it is clear that misdiagnosing a person with real cancer as a healthy person is much higher than the cost of misdiagnosing a healthy person as a cancer patient. Since cancer patients are in the minority in real life, the method tends to diagnose patients as healthy. Cost-sensitive learning is an algorithm to solve such problems. Its core element is the cost matrix. Taking medical diagnosis as an example, its cost-sensitive matrix may be written as the following form:

\[ C = \begin{bmatrix} 0 & 100 \\ 1 & 0 \end{bmatrix} \]

where the column represents the actual label, and the row represents the predictive label such as \( C(1, 2) \) represents the cost of diagnosing a cancer patient as a healthy person.

Many existing works on cost-sensitive learning [24], [25], [42]–[47] fall roughly into two categories. One is to adjust the sample’s distribution [24], [43], [44], [46], [47]. It transforms the frequencies of categories to proportions according to the cost of misclassification. The advantage is that the change of sample distributions may affect the performance of the algorithm. The other is metacost learning [25], [42], [45], a method to transform the general classification model into a cost-sensitive model. Kukar and Kononenko [42] first applied cost-sensitive learning to neural networks. Although there are many works on cost-sensitive learning, there are few studies on cost-sensitive adversarial learning. In [48], cost-sensitive learning is first applied to adversarial training, and a minimum–maximization method for generating robust classifiers is proposed. This method can directly minimize the error cost of convex optimization problems, but it is only applicable to linear classifiers. Asif et al. [48] encoded each adversarial transformation into a matrix, called the cost matrix. They then use the adversarial robust learning method in [23] to propose a new objective function for training the cost-sensitive robust classifier. Terzi et al. [22] proposed the WPGD method and used it in adversarial training. It provides a simple method to make the model cost sensitive to control the balance of accuracy–robustness.

III. ADVERSARIAL LEARNING WITH COST-SENSITIVE CLASSES

Section III-A gives some basic symbol definitions, such as empirical risk minimization, adversarial training, and cost-sensitive learning. Then, we formally describe the problem this article solved. Section III-B shows the details about the CSA algorithm, which combines cost-sensitive learning and adversarial training to improve the model’s adversarial robustness. Actually, by analyzing the convolutional layer’s parameters, we find some characteristics of the model trained by adversarial training. For the sake of description, we abbreviate these characteristics as the Min–Max property. According to the Min–Max property, we propose a new algorithm that could improve the model’s adversarial robustness without adversarial training, called CSE. The CSE algorithm is described in Section III-C.

A. Symbol Definition and Problem Description

1) Empirical Risk Minimization: Given a training set with \( N \) samples \( D = \{ (x^{(i)}, y^{(i)}) \}_{i=1}^{N} \subset \mathbb{R}^d \times Y \) where \( Y = \{0, 1, \ldots, m-1\} \) and \( m \) is the number of classes, we can train a classifier by machine learning \( f : \mathbb{R}^d \rightarrow [0, 1]^m \), where \( \hat{y} = \arg \max_i f_i(x) \) represents the prediction result of the classifier on sample \((x, y) \in \{ (x^{(i)}, y^{(i)}) \}_{i=1}^{N} \). For the classification problem of \( m \) categories, the neural network is used to represent the classifier, and its training process can be described as the following optimization problem:

\[
\min_{(x, y) \sim D} \frac{1}{N} \sum_{i=1}^{N} L(f(x^{(i)}), y^{(i)})
\]

where \( L \) represents the loss function.

2) Adversarial Training: The set of adversarial examples concerning a sample \((x, y)\) is defined as \( B(x, y, f) = \{x_{adv} | \|x_{adv} - x\|_l \leq \epsilon \text{ and } \arg \max_j f_j (x_{adv}) \neq y\} \), where \( l \) represents \( l \)-norm. Then, the process of its adversarial training can be formulated as follows:

\[
\min_{(x, y) \sim D} \frac{1}{N} \max_{x_{adv} \in B(x, y)} L(f(x_{adv}), y).
\]

3) Cost-Sensitive Learning: The cost of categorization is usually represented by a cost matrix \( C \) as follows:

\[
C(i, j) = \begin{cases} 
\epsilon_{ij}, & i \neq j \\
0, & i = j
\end{cases}
\]

where \( \epsilon_{ij} \) represents the cost of misclassifying an example from class \( j \) to class \( i \).

For any sample \( x \) belonging to class \( j \), the optimal decision is equivalent to minimizing the following loss function:

\[
L(x, j) = \sum_i P(i|x) C(i, j).
\]
That is to say, $L(x, j)$ represents the cost expectation of class $j$ predicted by the model under the given sample $x$.

4) Proposed Problem Formulation: Given a training set with $N$ samples, $D = \{x^{(i)}, y^{(i)}\}_{i=1}^N$, where a special class, say the $p$th class we need to protect. Our goal is to train a classifier $f : X \rightarrow Y$, which can classify any clean sample $(x, y) \in \{x^{(i)}, y^{(i)}\}_{i=1}^N$ while has stronger robustness against adversarial examples $x_{adv} \in B(f, x_{p}, f)$ in the condition of the adversarial attack. In short, when improving the overall robustness of the classifier $f$, category $p$ is given priority to ensure the robustness.

B. Cost-Sensitive Adversarial Model

This section introduces cost-sensitive learning and adversarial training. Then, we propose a cost-sensitive adversarial model (hereinafter referred to as the CSA model for convenience). The CSA model can effectively improve the robustness of a certain class $p$ in the classification problem against adversarial attacks.

Note that we want to protect class $p$ and improve the robustness of the model regarding class $p$. To solve this problem, two subquestions need to be answered as follows.

1) How to improve the overall robustness of the model?
2) How to give priority to improve the robustness of class $p$ in the model?

Intuitively, we can improve the overall robustness of the model through adversarial training. Simultaneously, to particularly improve the robustness regarding class $p$, we add the cost matrix under the framework of the adversarial training and use the cost matrix to indicate which classes of the classifier need to be specially protected. In conclusion, the construction of the model misclassification of class $p$ into non-$p$ or non-$p$ into $p$ is higher than that of other misclassification. So, we define the following cost matrix:

$$C(i, j) = \begin{cases} 0, & i = j \\ c, & i = p \text{ or } j = p \\ 1, & \text{else} \end{cases}$$

(3)

where $p$ is the protected class number, $j$ represents the true label, and $i$ represents the class number predicted by the classifier. This matrix indicates that the cost is 0 when the classifier correctly identifies the sample. When a classifier misclassifies a $p$ class as a non-$p$ class or misclassifies a non-$p$ class as $p$ class, the cost is $c$. The cost is 1 in other conditions.

$$\min_{(x, y) \sim D} \left[ L(f(x_{adv}), y) + \sum_l f_l(x_{adv}) C(i, y) \right].$$

(4)

The first term in (4) is a traditional loss function, which can be the cross-entropy function, square deviation, or any existing loss function. It mainly plays a role in improving the overall performance of the model. The second term represents the expected output cost of the model. When $c = 1$, the CSA model degenerates back to a standard adversarial training model.

Equation (4) is a nonconvex problem. Thus, finding a solution is a bit of challenge. There is a natural explanation for adversarial training proposed in [2]; that is, an attack method is a solution to the internal max part of (4). In contrast, the external min part works on the entire dataset, making the model’s loss the least. For example, (5) is an attack method, called FGSM [3]

$$(x_{adv})_{FGSM} = x + \epsilon \text{sign}(\nabla L(f(x), y)).$$

(5)

Then, (4) can be solved as follows:

$$\min_{(x, y) \sim D} \left[ L(f(x_{adv}), y) + \sum_l f_l(x_{adv}) C(i, y) \right].$$

(6)

The sensitivity of the model to the protected category depends on the cost matrix $C$. Obviously, the cost of the protected category misclassified by the model is higher than that of the general category. When $C(i, j) = 1, i \neq j$, the model’s penalty for misclassification of any category is the same. The implementation of the CSA algorithm is presented in Algorithm 1.

C. Cost-Sensitive Adversarial Extension

This section first gives an explanation of the adversarial robustness of the CSA model, then gives an extension of the CSA model, called the CSE model. Compared with the CSA model, a significant feature is the CSE model does not need adversarial training to make the model robust.

1) Empirical Observations on CSA: To further study the principle of the CSA model, we analyze the difference between the CSA model and the ADV model based on the parameters of the convolutional layer (ADV represents the model that is training with standard adversarial learning. Turn to Section IV.

---

**Algorithm 1 CSA Algorithm**

1. Initialize parameters of network $\theta$ with random weights
2. Initialize dataset $D = \{x^{(i)}, y^{(i)}\}_{i=1}^N$
3. Initialize the batchsize $B$, epochs $T$
4. for $i=1$ to $T$ do
5. Initialize cost-sensitive matrix $C$
6. Sample $B$ examples $Q_1 = \{(x^{(i)}, y^{(i)})\}_{i=1}^B$
7. Get adversarial example $Q_2 = \{(x_{adv}, y) | (x, y) \in Q_1\}$
8. **Stage 1**: Update $\theta$ with clean examples $(x, y) \in Q_1$
9. Updated $\theta$ by minimizing loss function Eq. (4)
10. **Stage 2**: Update $\theta$ with adversarial examples $(x_{adv}, y) \in Q_2$
11. Updated $\theta$ by minimizing loss function Eq. (4)
12. end for
13. Output $\theta^*$
for details). Fig. 3(a) is the parameter distribution diagram of the convolution of the CSA model and the ADV model in the first layer. Fig. 3(b) arranges the convolution parameters of the CSA model and ADV model from left to right and top to bottom. Fig. 3(a) shows that in the first convolution parameter of the CSA model, most values are relatively small, and the number of values close to 0 is several times that of the ADV model. Fig. 3(b) shows that in both the CSA model and ADV model, part of the parameters in the first convolutional layer are relatively high, and the value of the CSA model is higher than that of the ADV model.

In order to further observe the influence of parameter distribution of the convolutional layer, we added the L2 norm term to the original CSA model, hoping that the absolute values of most parameters in the convolutional layer of the model tend to 0. The implementation of the CSE algorithm is presented in Algorithm 2.

Table 1 shows that in both the CSA model and ADV model, the number of values close to 0 is several times that of the ADV model. From Fig. 3(a) and (b), we find that the parameters of the first layer of CSA model, CSA+L2 model, and ADV model. From Fig. 4(a) and (b), we find that the parameters of the first layer of the model have the following features.

1) The number of parameter values approaching zero: ADV < CSA + L2 < CSA.

2) Among the many parameters, the absolute values of a small number of parameters are relatively large, and for the parameter values of the three models at the corresponding positions, we have ADV < CSA + L2 < CSA.

Therefore, we give the following two propositions based on the above experimental findings.

**Proposition 1:** The Min–Max property, that is, the absolute values of most parameters in the convolutional layer approach 0 while the absolute values of a few parameters are significantly larger than others, can help improve the adversarial robustness.

**Proposition 2:** When the model predicts sample, the parameters that have real implication on the prediction result are only a part.

Two above-mentioned observations and analysis give some explanations why the CSA model can improve the robustness of category $p$. The adversarial training gives the Min–Max property to the parameters in the model. The cost matrix can locate the parameters that play a decisive role in the prediction results of category $p$. When the adversarial training is combined with the cost matrix, the positioning effect of the cost matrix makes these parameters having the Min–Max property during the training to improve the robustness of the category $p$.

2) Extension of CSA Model: This section further mines the information of Proposition 1 and uses this information to design the CSE model.

Proposition 1 shows that if the model parameters have the Min–Max property, then the model has stringer adversarial robustness. Through the analysis in the previous section, we give two features of the Min–Max property.

**Feature 1:** Most of the parameters in the convolutional layer of the model tend to 0.

**Feature 2:** In the convolutional layer parameters of the model, some are very large relatively.

Now, we analyze Features 1 and 2 in convolutional layers from the viewpoint of uncertainty. Let $w_{conv}$ represent the parameters of convolutional layers. According to [18] and [49], the fuzziness vector can be defined as

$$\text{Fuzziness}(w_{conv}) = -\frac{1}{N} \sum_{i=1}^{N} [(\rho_i \log \rho_i + (1 - \rho_i) \log(1 - \rho_i))]$$

(8)

where $N$ is the size of vector $w_{conv}$. $\rho_i$ is defined as follows:

$$\rho_i = \frac{1}{1 + e^{w_i}}$$

(9)

where $w_i$ is the element in $w_{conv}$.

By minimizing (8), we have

$$w_i \to 0 \Rightarrow \rho_i \to 1$$

$$w_i \to \infty \Rightarrow \rho_i \to 0.$$

Therefore, we can train a model with Features 1 and 2 by minimizing (8). Then, we have

$$\min E_{(x,y)\sim D} \left[ L(f(x), y) + \sum_{i} f_i(x) C(i, y) + \gamma \text{Fuzziness}(w_{conv}) \right]$$

(10)

where $\gamma$ is a hyperparameter. Compared with (4), (10) omits the calculation of the Max function but adds a regular term Fuzziness($w_{conv}$). The implementation of the CSE algorithm is presented in Algorithm 2.

In a word, the CSE is a novel method of defense against adversarial attacks. The difference between the CSE model and the traditional adversarial defense methods (such as adversarial training) is that the CSE model does not need to calculate the Max function inside the optimization formula, which greatly reduces the training time of the model.
Algorithm 2 CSE Algorithm
1: Initialize parameters of network \( f = (x; \theta = [\theta_{\text{conv}}, \theta_{\text{other}}]) \) with random weights
2: Initialize dataset \( D = \{x(i), y(i)\}_{i=1}^{N} \)
3: Initialize the batchsize \( B \), learning rate \( \alpha \), epochs \( T \), gamma \( \gamma \)
4: for \( i=1 \) to \( T \) do
5: Initialize cost-sensitive matrix \( C \)
6: Sample \( B \) examples \( Q = \{x(i), y(i)\}_{i=1}^{B} \)
7: Calculate entropy loss \( l_1 = \sum_{i=1}^{B} L(f(x), y) \)
8: Calculate cost-sensitive loss \( l_2 = \sum_{i=1}^{B} L_s(f(x))C(i, y) \)
9: \( l_3 = \gamma \text{Fuzziness}(\theta_{\text{conv}}) \)
10: Then, \( L = l_1 + l_2 + l_3 \)
11: Update parameters of network \( \theta \leftarrow \theta - \alpha \nabla L \)
12: end for

D. Min–Max Property

The Min–Max property is first discovered by Shen et al. [18]. Shen et al. proposed that the neural-network models with Min–Max property have stronger adversarial robustness. The Min–Max property means that, after using a combination between L1 and L2 normalizations as the loss function, the training process will result in such a phenomenon that the weights in convolutional layers will tend toward 0 (the minimum) or a maximum value. This article considers more complex architectures of neural networks to enhance the representation ability and advocate to measure Min–Max property by minimizing fuzziness of convolutional layers.

It is worth noting that the Min–Max property is similar to an off-center technique proposed in metric learning [50]. The off-center technique is achieved by a representation in feature space based on such an idea that, given the center being 0.5, the similarity after transformation is required to tend toward 0 (the minimum) or 1 (the maximum) if the similarity before the transformation is less than or more than 0.5, respectively. It is observed that the parameters are close to 0 or far further away from 0 after several rounds of adversarial training in LeNet. It is an interesting observation which confirms from the angle of off-center that convolution can indeed summarize the features from low to high levels while high-level features have more representative abilities.

The process of training neural networks such that the weights in convolutional layer weights going to extremes can be implemented in different ways, for example, by minimizing the uncertainty of the convolutional layer [18] or by adding the regularization term in the loss function. Furthermore, it is found that a DNN with strong robustness against adversarial examples may not have the Min–Max property. However, based on the observation from a considerable number of simulations, a DNN with the Min–Max property usually has strong robustness against adversarial examples where the adversarial robustness means a tolerance to adversarial noise.

We recall some results in [18] where two neural-network models with simple architectures are considered.

1) Model#1: \( L_a(x; W_a) = -y^T \log(\text{softmax}(W_a x)) \), where \( W_a = (a_{ij}) \) and \( a_{ij} \sim U(0, 1) \).
2) Model#2: \( L_b(x; W_b) = -y^T \log(\text{softmax}(W_b x)) \), where \( W_b = (b_{ij}) \) and \( b_{ij} \sim B(1, p_1) \). \( B(1, p_1) \) is the binomial distribution. \( p_1 \) represents the probability of \( b_{ij} = 1 \) where \( p_1 > 0, p_1 << 1 \).

Based on both models, a theorem regarding the normal and bi-nominal distributions of weight parameters was given as follows.

Theorem 1: Suppose \( x \) is a vector and \( x_i \) is the \( i \)th subscript in \( x \) \forall i, the following inequality holds:

\[
\left| \frac{\partial E_{B_{ij} \sim B(1, p_1)}L(x; W_b)}{\partial x_i} \right| \leq \left| \frac{\partial E_{a_{ij} \sim U(0,1)}[L_a(x; W_a)]}{\partial x_i} \right|
\]

According to this theorem, we have the following inequality:

\[
|L_b(x; W_b) - L_b(x + \Delta; W_b)| 
\leq |L_a(x; W_a) - L_a(x + \Delta; W_a)|
\]

which indicates that Model#2 (with Min–Max property) is really having the adversarial robustness stronger than Model#1 (without the Min–Max property).

IV. EXPERIMENTS

In this section, we use Python3.6 and Jupyter Notebook to implement our algorithms. Using Advertorch [51], we adopt some adversarial attacks (FGSM [3], PGD [6], CW [7], MIA [52], L2BIA [52], and LinfBIA [52]) to evaluate the adversarial robustness of the model. We compare the performance of the following four models.

1) Standard Model (STD): This model is trained with clean examples by adopting cross-entropy as the loss function.
2) Extension of the Cost-Sensitive Adversarial Model (CSE): This is a model trained with clean examples by adopting a loss function (10).
3) Cost-Sensitive Adversarial Model (CSA): This is an adversarial training model trained with (4). The adversarial examples are generated by PGD.
4) A Model Combined CSE and Adversarial Training (CSE+ADV): This is a adversarial training model trained with (10). The adversarial examples are generated by PGD.

We evaluate these models on three standard datasets:

1) MNIST [53]; 2) CIFAR10 [54]; and 3) CIFAR100 [54].

1) The MNIST dataset consists of 60,000 training samples and 10,000 samples, each of which is a 28 \times 28 pixel handwriting digital image.
2) The CIFAR10 dataset consists of 60,000 32 \times 32 color images in ten classes, with 6000 images per class. There are 50,000 training images and 10,000 testing images.
3) CIFAR100 is just like the CIFAR10, except it has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. Some examples are shown in Fig. 2.

In the experiments, all the adversarial perturbations are limited to a \( L_\infty \)-norm ball. Let \( M_p \) represent the model \( M \) which particularly protects category \( p \). Before training network, samples from datasets will be regularized. The preprocessing is
described as follows:

\[ x = \frac{x - \mu}{\sigma} \]

where \( \mu \) and \( \sigma \) in a different dataset are shown in Table I.

### A. MINST

For the MNIST dataset, we adopt LeNet [55] as our network structure. LeNet network is composed of two convolutional layers, two pooling layers, and two full connection layers. Two convolutional layers contain 5 and 16 convolution kernels, respectively. The convolutional layer is followed by a pooling layer, the padding is set to 2. The number of two full connection layer hidden nodes are 120 and 84, respectively. RELU [56] function is adopted as the activation function in the network. During the training stage, the batch size is 256, and the learning rate is 0.01, the epoch is 20, and the momentum is 0.95. For the CSA model and CSE+ADV model, the first ten epochs are trained with clean samples, and the last ten epochs are trained with adversarial examples generated by PGD. Maximum perturbation of the attack \( \epsilon = 0.3 \). We set the constant \( c = 10 \).

We, respectively, conduct ten experiments for CSA, CSE, and CSE+ADV to protect each class (0–9). Table II is the result of various models on the testing dataset, where \( \epsilon = 0.3 \). We only show the accuracy of the particularly protected class. For example, \( O_1, p = 1 \) represents the accuracy of class “one” in a model that is trained under protecting class “one.” However, the STD model, as a baseline, does not adopt any protection strategy. It can be seen from the table that the CSA, CSE and CSE+ADV can effectively improve the adversarial robustness of the model than STD. Compared the CSE with STD model, we find that CSE can achieve high adversarial robustness although both models do not use adversarial training. When adding adversarial training, it is worth-noted that CSE+ADV achieves better performance than CSA in most scenarios. These results experimentally validated that the Min–Max property of the model can improve the model’s adversarial robustness. Moreover, it is an effective way to reduce the fuzziness of parameters in convolutional layers.

Fig. 5(a)–(c) shows the change trend of the loss function, fuzziness of parameters in convolutional layer, and the robustness. The protected label is set to the first class. We find that CSE is harder to converge than STD. It would take more time for training CSE than STD. However, shown in Fig. 5(b), as the number of iterations increases, the fuzziness of the convolutional layer decreases gradually in CSE but remaining stable in STD. This phenomenon demonstrates that our method can effectively reduce the uncertainty of parameters, which gives a model Min–Max property which reflects in the result that the adversarial robustness of CSE is stronger than STD as shown in Fig. 5(c). Besides, we note that the fuzziness of convolutional layer in the STD and CSE is around 0.7 at the beginning. This is due to the fact that the parameters are initialized randomly in a same way.

### B. CIFAR10

In the experiment of CIFAR10, the neural network still is a LeNet network. The difference with the network on MNIST is that the first layer convolution has three channels, and it does not need padding operation. The hyperparameters are shown as follows: the epochs are 50, the batch size is 256, the optimizer is Adam, and the learning rate is 0.01. Every ten epochs, the learning rate will drop by half. When training the CSA and CSE+ADV, both clean examples and adversarial examples are used as inputs every iteration. The adversarial examples are generated by PGD.

In the test of the model’s adversarial robustness, assuming the adversarial perturbation is limited in \( l_\infty \)-norm ball with \( \epsilon = 8/255 \), we evaluate the model’s performance under FGSM, PGD, CW, MIA, L2BIA, and LinfBIA attack methods. The results are shown in Table III. To convenient for showing protecting performance, we only show the accuracy of the protected class in each scenario as same as that on MNIST. Under most adversarial attack methods except for CW, CSE has stronger adversarial robustness than STD. The CSA and CSE+ADV models have much stronger adversarial robustness than STD.

However, it is worth noting that all the models are vulnerable to adversarial examples generated by CW. Actually, CW is a strong attack method which does not depend on gradients. It has been found that adversarial training would cause gradient obfuscation which gives one a false sense that the model has adversarial robustness [15]. Therefore, we conclude that the Min–Max property would cause gradient obfuscation in another way.

The CIFAR10 dataset is much more complex than MNIST. Therefore, the number of iterations may be much more than that of MNIST, as shown in Fig. 5(a) and (d). When the number of iterations is sufficient, both STD and CSE can converge eventually. This is not obvious when training CSE on MNIST (so it does not require too many iterations). Similar to the MNIST experiments, with the increase of iterations, the fuzziness of the convolutional layer decreases gradually, for example, the Min–Max property of convolution becomes more obvious. Thus, the model becomes more adversarial robust.

### C. CIFAR100

In this section, we will test our proposed methods on a larger, more realistic dataset. On the CIFAR100 dataset, the simple LeNet network will no longer be applicable. Therefore, ResNet18 is adopted as the structure of the model. The following is the introduction of some hyperparameters: the learning rate is 0.1 at begin, and it will be dropped by half every 60 epochs.

| Dataset | \( \mu \)       | \( \sigma \)     |
|---------|----------------|-----------------|
| MNIST   | [0.1307]       | [0.3081]        |
| CIFAR10 | [0.4914, 0.4822, 0.4465] | [0.2023, 0.1994, 0.2010] |
| CIFAR100| [0.5070, 0.4865, 0.4409] | [0.2673, 0.2564, 0.2761] |
epochs; epochs are 180; batch size is 256; optimizer is SGD; and the constant in cost matrix is 10. When training CSA and CSE + ADV, alternate training modes of clean examples and adversarial examples are adopted. All adversarial examples in training are generated by PGD.

Since CIFAR10 has 100 classes, it would be a waste of time to train 100 × 3 models (ADV, CSA, and CSE) if each class is protected once. Therefore, as shown in Table IV, we only show partial results since CIFAR10 contains 100 classes. Nevertheless, we find our methods improve insignificantly adversarial robustness of the model on deeper neural networks and more complicated datasets. To find the failure reason, we further explore the convolutional layer’s fuzziness in ResNet18. Fig. 6 shows the fuzziness of convolutional layer parameters in ResNet18 on CIFAR10. For the CSE model, the protected label is 21. We discover that the fuzziness of convolutional layer parameters is almost hard to become small on CIFAR10. The fuzziness of each convolutional layer in the STD model is almost identical to that of the CSE model. We think this is due to the depth of the network. The proposed

Table II: Accuracy of protected categories under various attack methods. P represents the category of protection. Oi represents the accuracy of category i for the data in the table. Dataset is MNIST.

| Attacks | Training | Models | O0, p=0 | O1, p=1 | O2, p=2 | O3, p=3 | O4, p=4 | O5, p=5 | O6, p=6 | O7, p=7 | O8, p=8 | O9, p=9 |
|---------|----------|--------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| FGSM No | STD | 0.8771 | 0.9491 | 0.7667 | 0.8401 | 0.7230 | 0.8737 | 0.8381 | 0.6053 | 0.8227 | 0.6830 |
|        | CSA | 0.9448 | 0.9674 | 0.9109 | 0.8796 | 0.8849 | 0.8574 | 0.8424 | 0.9052 | 0.9048 | 0.7099 |
|        | Yes | 0.9566 | 0.9792 | 0.9426 | 0.9229 | 0.9013 | 0.9311 | 0.9464 | 0.9406 | 0.8953 |
|        | CSA+ADV | 0.7157 | 0.7749 | 0.4019 | 0.6155 | 0.2730 | 0.5787 | 0.6061 | 0.1425 | 0.3767 | 0.2096 |
|        | CSA | 0.8740 | 0.7954 | 0.7341 | 0.6293 | 0.7497 | 0.7284 | 0.5535 | 0.7187 | 0.5696 | 0.2096 |
|        | Yes | 0.8336 | 0.9066 | 0.9792 | 0.8023 | 0.7398 | 0.7094 | 0.8664 | 0.7941 | 0.8023 | 0.7404 |
|        | CSA+ADV | 0.8508 | 0.9553 | 0.9099 | 0.8667 | 0.9225 | 0.8961 | 0.9367 | 0.9137 | 0.8805 | 0.9069 |
| PGD No | STD | 0.5837 | 0.9036 | 0.4403 | 0.6052 | 0.5519 | 0.6539 | 0.5590 | 0.5189 | 0.2413 | 0.3317 |
|        | CSA | 0.7497 | 0.9118 | 0.6141 | 0.4539 | 0.6105 | 0.6614 | 0.4996 | 0.7477 | 0.4196 | 0.3337 |
|        | Yes | 0.4524 | 0.6934 | 0.6194 | 0.8362 | 0.7059 | 0.6969 | 0.7442 | 0.7296 | 0.4248 | 0.4982 |
|        | CSA+ADV | 0.7070 | 0.8541 | 0.7428 | 0.6548 | 0.5446 | 0.6247 | 0.7448 | 0.7926 | 0.5427 | 0.4781 |
| MIA No | STD | 0.7409 | 0.8342 | 0.4107 | 0.6717 | 0.3211 | 0.6280 | 0.6180 | 0.1835 | 0.4031 | 0.2629 |
|        | CSA | 0.8455 | 0.3360 | 0.6385 | 0.5440 | 0.7150 | 0.6397 | 0.4971 | 0.6592 | 0.5031 | 0.2629 |
|        | Yes | 0.8403 | 0.9171 | 0.8085 | 0.8271 | 0.7706 | 0.7162 | 0.8656 | 0.8089 | 0.8294 | 0.7680 |
|        | CSA+ADV | 0.8771 | 0.9542 | 0.9049 | 0.8737 | 0.9305 | 0.8964 | 0.9462 | 0.9141 | 0.8811 | 0.9073 |
| L2BIA No | STD | 0.9926 | 0.7975 | 0.9647 | 0.9552 | 0.9467 | 0.9504 | 0.9872 | 0.9613 | 0.9477 | 0.9729 |
|        | CSA | 0.9738 | 0.9927 | 0.9762 | 0.9859 | 0.9760 | 0.9789 | 0.9633 | 0.9810 | 0.9827 | 0.9729 |
|        | Yes | 0.9834 | 0.9934 | 0.9918 | 0.9879 | 0.9935 | 0.9936 | 0.9890 | 0.9827 | 0.9884 | 0.9916 |
|        | CSA+ADV | 0.9893 | 0.9955 | 0.9881 | 0.9854 | 0.9819 | 0.9852 | 0.9911 | 0.9872 | 0.9925 | 0.9913 |
| LinFBIA No | STD | 0.8845 | 0.8519 | 0.7626 | 0.6562 | 0.2400 | 0.5700 | 0.5774 | 0.1223 | 0.3371 | 0.1734 |
|        | CSA | 0.8375 | 0.8404 | 0.7771 | 0.8029 | 0.7197 | 0.7129 | 0.8551 | 0.7704 | 0.7952 | 0.7182 |
|        | Yes | 0.8388 | 0.9553 | 0.9031 | 0.8670 | 0.9229 | 0.8668 | 0.9550 | 0.9142 | 0.8650 | 0.9097 |

Fig. 5. Changing trend of the loss function, fuzziness and robustness on MNIST and CIFAR10. (a) Loss on MNIST. (b) Fuzziness of first convolutional layer on MNIST. (c) Robustness on MNIST. Evaluation under PGD. (d) Loss on CIFAR10. (e) Fuzziness of first convolutional layer on CIFAR10. (f) Robustness on CIFAR10. Evaluation under PGD.
TABLE III

| Attacks       | ADV Training | Models        | $O_{p=0}$ | $O_{p=1}$ | $O_{p=2}$ | $O_{p=3}$ | $O_{p=4}$ | $O_{p=5}$ | $O_{p=6}$ | $O_{p=7}$ | $O_{p=8}$ | $O_{p=9}$ |
|---------------|--------------|---------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| FGSM          | No           | STD           | 0.393     | 0.328     | 0.159     | 0.069     | 0.116     | 0.156     | 0.287     | 0.317     | 0.332     | 0.283     |
|               | Yes          | CSSE          | 0.4269    | 0.4399    | 0.2212    | 0.0862    | 0.1325    | 0.2136    | 0.3580    | 0.4029    | 0.4566    | 0.3748    |
|               |              | CSSE+ADV      | 0.5382    | 0.5904    | 0.3423    | 0.1982    | 0.2817    | 0.3539    | 0.5937    | 0.5497    | 0.6284    | 0.5260    |
| PGD           | No           | STD           | 0.2190    | 0.2584    | 0.0918    | 0.0270    | 0.0439    | 0.1065    | 0.1804    | 0.2586    | 0.2424    | 0.2144    |
|               | Yes          | CSSE          | 0.3628    | 0.4059    | 0.1975    | 0.0682    | 0.0810    | 0.1770    | 0.3118    | 0.3881    | 0.4039    | 0.3561    |
|               |              | CSSE+ADV      | 0.5677    | 0.5602    | 0.3186    | 0.1904    | 0.2770    | 0.3431    | 0.5850    | 0.5509    | 0.6115    | 0.5104    |
| CW            | No           | STD           | 0.0000    | 0.0000    | 0.0000    | 0.0000    | 0.0000    | 0.0000    | 0.0000    | 0.0000    | 0.0000    | 0.0000    |
|               | Yes          | CSSE          | 0.0000    | 0.0000    | 0.0000    | 0.0000    | 0.0000    | 0.0000    | 0.0000    | 0.0000    | 0.0000    | 0.0000    |
|               |              | CSSE+ADV      | 0.0000    | 0.0000    | 0.0000    | 0.0000    | 0.0000    | 0.0000    | 0.0000    | 0.0000    | 0.0000    | 0.0000    |
| MIA           | No           | STD           | 0.2041    | 0.2546    | 0.0988    | 0.0322    | 0.0404    | 0.1049    | 0.1752    | 0.2446    | 0.2255    | 0.2051    |
|               | Yes          | CSSE          | 0.3702    | 0.4069    | 0.1961    | 0.0618    | 0.0756    | 0.1815    | 0.3011    | 0.3703    | 0.3837    | 0.3463    |
|               |              | CSSE+ADV      | 0.5636    | 0.5791    | 0.3101    | 0.1849    | 0.2851    | 0.3367    | 0.5845    | 0.5501    | 0.6105    | 0.5116    |
| L2BIA         | No           | STD           | 0.6796    | 0.7365    | 0.4834    | 0.4315    | 0.5427    | 0.4984    | 0.6877    | 0.6666    | 0.7128    | 0.6816    |
|               | Yes          | CSSE          | 0.7046    | 0.7411    | 0.4936    | 0.3661    | 0.5114    | 0.4915    | 0.6854    | 0.6039    | 0.7761    | 0.6783    |
|               |              | CSSE+ADV      | 0.7041    | 0.7439    | 0.4935    | 0.3987    | 0.5277    | 0.5175    | 0.7424    | 0.6842    | 0.7612    | 0.7137    |
| LinfBIA       | No           | STD           | 0.4128    | 0.4447    | 0.2070    | 0.0819    | 0.1175    | 0.2010    | 0.3329    | 0.3940    | 0.4555    | 0.3709    |
|               | Yes          | CSSE          | 0.5407    | 0.5871    | 0.3393    | 0.2026    | 0.3009    | 0.3536    | 0.5586    | 0.5548    | 0.5995    | 0.5586    |
|               |              | CSSE+ADV      | 0.5699    | 0.5830    | 0.3425    | 0.1870    | 0.2902    | 0.3556    | 0.5903    | 0.5504    | 0.6289    | 0.5136    |

V. Conclusion

A great deal of work has been done to improve the overall adversarial robustness of a model. However, in some specific problems, the cost of each class of adversarial attack often varies greatly. Therefore, we consider protecting the adversarial robustness of specific categories. While improving the overall adversarial robustness of a model, we prioritize improving the adversarial robustness of specific classes. Experimentally, we showed that the cost-sensitive training can effectively protect specific categories from adversarial attacks. Moreover, we found that the robustness of the model is closely related to the convolutional layer parameter distribution in LeNet networks. Also, we experimentally found that the more obvious Min–Max property of the convolutional layer parameter in LeNet, the stronger the adversarial robustness of the model will be.

There exist cases in which our method may not be successful in improving the adversarial robustness of the model when we apply the proposed method to a more complicated dataset. These cases may need deeper network structures to be designed. How to effectively control the uncertainty of
convolutional layer parameters in a deep network, which significantly has impact on adversarial robustness, remains to be studied further.

REFERENCES

[1] M. Cheng, Q. Lei, P.-Y. Chen, I. S. Dhillon, and C.-J. Hsieh, “CAT: Customized adversarial training for improved robustness,” 2020, arXiv:2002.06789.

[2] C. Szegedy et al., “Intriguing properties of neural networks,” in Proc. 2nd Int. Conf. Learn. Represent. (ICLR), Banff, AB, Canada, Apr. 2014.

[3] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” in Proc. 3rd Int. Conf. Learn. Represent. (ICLR), San Diego, CA, USA, May 2015.

[4] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, “The limitations of deep learning in adversarial settings,” in Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P), 2016, pp. 372–387.

[5] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, “Deepfool: A simple and accurate method to fool deep neural networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 2574–2582.

[6] A. Athalye, N. Carlini, and D. A. Wagner, “Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples,” in Proc. 35th Int. Conf. Learn. Mach. Learn. (ICML), vol. 80, Stockholm, Sweden, Jul. 2018, pp. 5283–5292.

[7] C. Elkan, “The foundations of cost-sensitive learning,” in Proc. Int. Joint Conf. Artif. Intell., vol. 17, 2001, pp. 973–978.

[8] Z.-H. Zhou and X.-Y. Liu, “On multi-class cost-sensitive learning,” Comput. Intell., vol. 26, no. 3, pp. 232–257, 2010.

[9] S. Pan, R. Hu, X.-F. Feng, G. Long, J. Jiang, and C. Zhang, “Learning graph embedding with adversarial training methods,” IEEE Trans. Cybern., vol. 50, no. 6, pp. 2475–2487, Jun. 2020.

[10] Z. Lu, C. Long, J. Zhang, M. Xu, and P. Lv, “TriATNE: Tripartite adversarial training for network embeddings,” IEEE Trans. Cybern., early access, Mar. 17, 2021, doi: 10.1109/TCYB.2021.3061771.

[11] W. Zhao et al., “Leveraging long and short-term information in content-aware movie recommendation via adversarial training,” IEEE Trans. Cybern., vol. 50, no. 11, pp. 4680–4693, Nov. 2020.

[12] M. Abbasi and C. Gagné, “Robustness to adversarial examples through an ensemble of specialists,” in Proc. 5th Int. Conf. Learn. Represent. (ICLR), Toulon, France, 2017.

[13] Y. Liu, X. Chen, C. Liu, and D. Song, “Delving into transferable adversarial examples and black-box attacks,” in Proc. 5th Int. Conf. Learn. Represent. (ICLR), Toulon, France, 2017.

[14] Z. Zhao, D. Dua, and S. Singh, “Generating natural adversarial examples,” in Proc. 6th Int. Conf. Learn. Represent. (ICLR), Vancouver, BC, Canada, 2018.

[15] N. Papernot, P. McDaniel, and I. Goodfellow, “Transferability in machine learning: From phenomena to black-box attacks using adversarial samples,” 2016, arXiv:1605.07277.

[16] R. Huang, B. Xu, D. Schuurmans, and C. Szepesvári, “Learning with a strong adversary,” 2015, arXiv:1511.03034.

[17] A. Kurakin, I. J. Goodfellow, and S. Bengio, “Adversarial machine learning at scale,” in Proc. 5th Int. Conf. Learn. Represent. (ICLR), Toulon, France, 2017.

[18] G. Liu, I. Khalil, and A. Khreich, “Using single-step adversarial training to defend iterative adversarial examples,” in Proc. CODASPY 11th ACM Conf. Data Appl. Secur. Privacy, 2021, pp. 17–27.

[19] A. Shafahi et al., “Adversarial training for free!” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 3358–3369.

[20] D. Zhang, T. Zhang, Y. Lu, Z. Zhu, and B. Dong, “You only propagate once: Accelerating adversarial training via maximal principle,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 227–238.

[21] E. Wong, L. Rice, and J. Z. Kolter, “Fast is better than free: Revisiting adversarial training,” 2020, arXiv:2001.03994.

[22] C. Qin et al., “Adversarial robustness through local linearization,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 13824–13833.

[23] S. Dhar and V. Cherkassky, “Development and evaluation of cost-sensitive universal-SVM,” IEEE Trans. Cybern., vol. 45, no. 4, pp. 806–818, Apr. 2015.

[24] A. Riccardi, F. Fernández-Navarro, and S. Carloni, “Cost-sensitive adaboost algorithm for ordinal regression based on extreme learning machine,” IEEE Trans. Cybern., vol. 44, no. 10, pp. 1898–1909, Oct. 2014.

[25] M. Kukar and I. Kononenko, “Cost-sensitive learning with neural networks,” in Proc. ECAL, vol. 98, pp. 445–449.

[26] N. Abe, B. Zadrozny, and J. Langford, “An iterative method for multi-class cost-sensitive learning,” in Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2004, pp. 3–11.

[27] L. Jiang, C. Li, and S. Wang, “Cost-sensitive bayesian network classifiers,” Pattern Recognit., vol. 45, pp. 211–216, Aug. 2012.

[28] P. M. Domingos, “MetaCost: A general method for making classifiers cost-sensitive,” in Proc. 5th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 1999, pp. 155–164.

[29] W.-Y. Loh, “Fifty years of classification and regression trees,” in Proc. Adv. Neural Inf. Process. Syst., 2010, pp. 425–432.

[30] K. Asif, W. Xing, S. Behpour, and B. D. Ziebart, “Adversarial cost-sensitive classification,” in Proc. UAI, 2015, pp. 92–101.

[31] J. Basak, R. K. De, and S. K. Pal, “Unsupervised feature selection using a neuro-fuzzy approach,” Pattern Recognit. Lett., vol. 19, no. 11, pp. 997–1006, 1998.

[32] D. Yan, X. Zhou, X. Wang, and R. Wang, “An off-center technique: Learning a feature transformation to improve the performance of clustering and classification,” Int. J. Knowl. Discov. Data Mining, vol. 503, pp. 635–651, Nov. 2019.
[51] G. W. Ding, L. Wang, and X. Jin, “AdverTorch v0.1: An adversarial robustness toolbox based on pytorch,” 2019, arXiv:1902.07623.

[52] Y. Dong et al., “Boosting adversarial attacks with momentum,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 9185–9193.

[53] L. Deng, “The MNIST database of handwritten digit images for machine learning research [best of the Web],” IEEE Signal Process. Mag., vol. 29, no. 6, pp. 141–142, Nov. 2012.

[54] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” J. Mach. Learn. Res., vol. 1, no. 4, p. 7, 2009.

[55] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[56] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” in Proc. 14th Int. Conf. Artif. Intell. Stat., 2011, pp. 315–323.

Haojing Shen received the B.S. degree in mathematics from Shenzhen University, Shenzhen, China, in 2019, where he is currently pursuing the M.Sc. degree with the College of Computer Science and Software Engineering.

His current research interests include adversarial example and machine learning.

Sihong Chen received the B.S. degree in mathematics from Shenzhen University, Shenzhen, China, in 2019, where he is currently pursuing the M.Sc. degree with the College of Computer Science and Software Engineering.

His current research interests include adversarial example and adversarial robustness.

Rao Wang (Member, IEEE) received the B.Eng. degree in computer science from the College of Information Science and Technology, Beijing Forestry University, Beijing, China, in 2009, and the Ph.D. degree from the Department of Computer Science, City University of Hong Kong, Hong Kong, in 2014.

From 2014 to 2016, she was a Postdoctoral Researcher with the Department of Computer Science, City University of Hong Kong. She is currently an Associate Professor with the College of Mathematics and Statistics, Shenzhen University, Shenzhen, China. Her current research interests include pattern recognition, machine learning, fuzzy sets and fuzzy logic, and their related applications.

Xizhao Wang (Fellow, IEEE) received the Doctoral degree in computer science from the Harbin Institute of Technology, Harbin, China, in 1998.

From 1998 to 2001, he was with the Department of Computing, Hong Kong Polytechnic University, Hong Kong, as a Research Fellow. From 2001 to 2014, he served with Hebei University, Baoding, China, as a Professor and the Dean of the School of Mathematics and Computer Sciences. Since 2014, he has been a Professor with the Big Data Institute, Shenzhen University, Shenzhen, China. He has edited over ten special issues and published three monographs, two textbooks, and over 200 peer-reviewed research papers. His current research interests include uncertainty modeling and machine learning for big data.

Dr. Wang was a recipient of the IEEE SMCS Outstanding Contribution Award in 2004 and the IEEE SMCS Best Associate Editor Award in 2006. He is the previous BoG Member of IEEE SMC Society, the Chair of IEEE SMC Technical Committee on Computational Intelligence, the Chief Editor of the International Journal of Machine Learning and Cybernetics, and an associate editor for a couple of journals in the related areas. He is the General Co-Chair of the 2002–2017 International Conferences on Machine Learning and Cybernetics, cosponsored by IEEE SMCS. He was a Distinguished Lecturer of IEEE SMCS.