# DeT: Defending Against Adversarial Examples via Decreasing Transferability

Changjiang Li[1], Haiqin Weng[1], Shouling Ji[1(✉)], Jianfeng Dong[2], and Qinming He[1]

[1] Zhejiang University, Hangzhou, China
{cj_li,hq_weng,sji,hqm}@zju.edu.com
[2] Zhejiang Gongshang University, Hangzhou, China
danieljf24@gmail.com

**Abstract.** Deep neural networks (DNNs) have made great progress in recent years. Unfortunately, DNNs are found to be vulnerable to adversarial examples that are injected with elaborately crafted perturbations. In this paper, we propose a defense method named DeT, which can (1) defend against adversarial examples generated by common attacks, and (2) correctly label adversarial examples with both small and large perturbations. DeT is a transferability-based defense method, which to the best of our knowledge is the first such attempt. Our experimental results demonstrate that DeT can work well under both black and gray box attacks. We hope that DeT will be a benchmark in the research community for measuring DNN attacks.

**Keywords:** Deep learning · Adversarial examples · Transferability

## 1 Introduction

Deep neural networks (DNNs) have been widely used in many challenging tasks, showing impressive performance. For some tasks, DNNs have already exceeded human beings, e.g., image recognition, text classification, and speech recognition [4,6,13,14,21,23,25]. Not surprisingly, however, researchers have found that DNNs are vulnerable to some specially perturbed examples called adversarial examples [3,5]. Such adversarial examples are so harmful since they are injected with elaborately crafted perturbations that are imperceptible to humans, even the model interpretation methods, while can fool DNNs [27].

To defend against adversarial examples, researchers have proposed various defense methods. Goodfellow et al. proposed adversarial training [5], which trains the model on the augmented dataset containing adversarial examples. Papernot et al. proposed the defensive distillation to improve the robustness of the model [20]. They first train one DNN model to predict soft labels for training examples, and then train another model on the same training examples with the predicted soft labels. Meng et al. proposed a novel method called Magnet which combines with detectors and reformers [16]. Detectors are designed to detect adversarial


J. Vaidya et al. (Eds.): CSS 2019, LNCS 11982, pp. 307–322, 2019.
https://doi.org/10.1007/978-3-030-37337-5_25

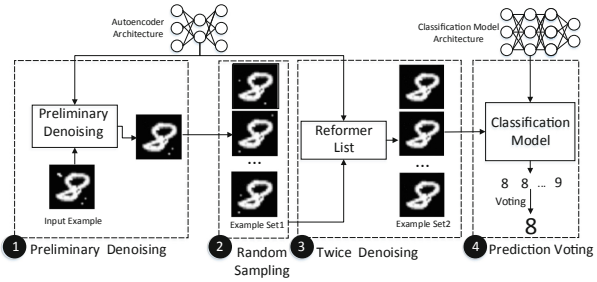



**Fig. 1.** Working flow of our proposed DeT.

examples and reformers are designed to reconstruct input examples. Cao et al. proposed a region-based classification which selects examples in a hypercube centered on the input example [1] and sends them to the classifier. The most frequent label is deemed as the final predicted label.

The above methods however have their own limitations. Adversarial training can only defend against specific adversarial examples similar to those in the training set. Defensive distillation cannot significantly improve the robustness of the classifier, and it is easy to be attacked [3]. Region-based methods work poorly to defend against adversarial examples with large perturbations. Magnet has been broken by Carlini&Warner leveraging the transferability of adversarial examples and requires extensive human labors [2]. In summary, the existing defenses have the following limitations: (1) it can only defend against specific adversarial examples [5] and adversarial examples with slight perturbations [1]; and (2) it needs extensive human labors [16]. Therefore, there is a long-term arms race between the attackers and the defenders [15].

To address the above limitations, we propose a robust defense method called DeT. It uses a list of reformers[1] to reduce the transferability of adversarial examples and thus mitigates the threat posed by the adversary. We propose a cross training algorithm to generate several complementary reformers, among which the transferability of adversarial examples is low. The defense ability of DeT is mainly based on the low transferability of adversarial examples among the reformers. DeT can (1) defend against adversarial examples generated by common attacks, and (2) still give the correct output for adversarial examples with large perturbations.

Figure 1 illustrates the working flow of DeT. Given an input example, DeT first adds Gaussian noises on the input example and then sends the noised example to a specific reformer for preliminary denoising. After preliminary denoising, we can get a denoised example close to the original input. Then, we perform a non-differentiable random sampling operation on the denoised example. Under the gray box circumstance, this makes the attack method unable to perform the gradient-based optimization. By the sampling operation, we obtain an example set

[1] The details of reformers will be explained in Sect. 3.

where each example is very close to the original input. We use several reformers to reconstruct all the examples from the example set, and send the reconstruction results to the classifier for labeling. We select the most frequently appeared label in the predicted label collection as the output label.

The contributions of this paper are summarized as follows.

- We propose an algorithm that can significantly decrease the transferability of adversarial examples among different reformers. This algorithm can be used to defend against the gray box attack. To our knowledge, it is the first attempt to defend against adversarial examples through reducing the transferability of adversarial examples.
- We introduce a robust defense method DeT which can significantly increase the accuracy of the classifier under common attacks.
- DeT can give the correct output even if adversarial examples are with large adversarial perturbations.

## 2 Background

In this section, we first introduce the threat model considered in this paper, followed by reviewing the related work.

### 2.1 Threat Model

The attack situation can be divided into the following two situations based on the adversary's understanding of our defense methods.

**Black Box Attack.** The adversary only knows the structure and parameters of the classifier while does not know anything about the defense method.

**Gray Box Attack.** The adversary not only knows the structure and parameters of the classifier but also knows the defense method. In this paper, we assume a more powerful adversary who can get the structure and parameters of one reformer used in the Reformer List.

Below we briefly review the related work from four fields: deep neural networks, adversarial examples, common attacks, and defenses.

### 2.2 Deep Neural Networks

Deep neural networks (DNNs) have made great progress in recent years, especially the convolutional neural network (CNN) [12]. DNN is widely used in many fields, such as image recognition [10,23], speech recognition [6,8], medical treatments [26] and information security [9,25].

**Notation.** For simplicity, let $X = \{x_i |\ i = 1, 2, ...\}$ denote the set of input examples, where $x_i$ is an example. Let $X' = \{x'_i |\ i = 1, 2, ...\}$ denote the set of adversarial examples corresponding to $X$. Given $X$, let $C(x_i)$ denote the estimated label of $x_i$.

A DNN is a function model, denoted by $y = F(x) = f_L(f_{L-1}(...(f_1(x))))$, where $y$ represents the output vector and $y \in \mathbb{R}^m$. $f_i$ represents layer $i$ in DNN, and $f_L$ is often a $softmax$ layer which maps from logits (the result vector before the softmax layer) to the output vector. The output vector is in the range $[0, 1]$ that adds up to 1. The value of $F(x_i)_j$ is regarded as the probability of the input $x_i$ belonging to class $j$. The classifier will consider the label $C(x_i) = argmax_j F(x_i)_j$ as the predicted label.

### 2.3 Adversarial Examples

The concept of adversarial examples was first proposed by Szegedy et al. [24]. Those examples with elaborately crafted perturbations are called adversarial examples.

To measure the perturbations, researchers usually adopt three distance metrics: $L^0, L^2$, and $L^\infty$. If the distances are small enough, we can guarantee that the adversarial perturbations of adversarial examples are undetectable to humans.

Transferability is of key importance to adversarial examples. Transferability means that adversarial examples generated for one model can also lead to the misclassification of another, even if these two models have different structures and are trained on different datasets [19]. DeT significantly decreases the transferability of adversarial examples among reformers.

### 2.4 Common Attacks

**Fast Gradient Sign Method (FGSM)** utilizes the gradient information to generate adversarial examples [5]. Specially, the adversarial examples generated by FGSM can be formulated as $x' = x + \epsilon * sign(\nabla_x L(F(x), y))$, where $L(.,.)$ denotes the loss between the model output of $x$ and the ground truth label $y$ and $sign$ is a function that extracts the sign of its inputs. The generated adversarial examples maximize the loss function. The parameter $\epsilon$ controls the attack strength. An increase of $\epsilon$ can improve the success rate of the attack, though it might result in human-perceptible noises.

**Basic Iterative Method (BIM)** is an extension of FGSM [11]. Different from FGSM that takes one step to change the original examples, BIM takes multiple small steps with step-width $\alpha$. Compared with FGSM, we can see that BIM is designed as the multiple iterations version of FGSM.

**Deepfool** iteratively searches the minimal $L^2$ adversarial perturbations for a given example [17]. In the iterative process, it adds small perturbations to the given example at a time until the example crosses the decision boundary.

**Carlini&Wagner Attacks (C&W)** include three attacks that can almost reach an incredible success rate of 100% with human imperceptible perturbations [3]. The attacks included in C&W can be classified as the targeted and non-targeted attacks. The targeted attack makes the output to meet the target requirements (e.g., misclassify the input example to the specified target class) while the non-targeted attack only requires the target model to misclassify the

input examples. Due to the space limitation, we refer interesting authors to [3] for more details.

In this paper, we focus on non-targeted attacks since they are more easy to be achieved for the adversary. As $L^2$-based C&W attack is the strongest attack method, we use it to evaluate our proposed defense method.

### 2.5 Common Defenses

**Adversarial Training** aims to train a more robust classification model by adding extra adversarial examples with ground truth label to the training dataset [24]. This method requires a great deal of computational cost to generate adversarial examples as well as retraining the model. The effectiveness of adversarial training depends on whether the attack methods being available.

**Region-based Classification** randomly selects a number of examples in the hypercube centered on the original input example [1]. These selected examples are then fed to the model. The model gives all labels of these selected examples, and it considers the most frequent label as the final output. The basic idea of this method is to assume that *the adversarial example is close to the decision boundary.*

**Magnet** is a framework for defending against adversarial examples [16]. Upon receiving an example, Magnet determines whether this example is adversarial or not. Magnet then refuses to classify the example if the example is detected as adversarial, otherwise Magnet reconstructs the input example to reduce its potential threat and sends the reconstructed example to the classifier.

Adversarial training can only defend against known attacks, and therefore it is not within our scope of comparison. In our experimental evaluation, we compare our method with Region-based method and Magnet.

## 3 Our Defense Method

In this paper, we propose a robust defense method called DeT to defend against adversarial examples. Figure 1 shows the overall architecture of DeT. DeT has four processes: *preliminary denoising*, *random sampling*, *twice denoising* and *prediction voting.*

Below, we introduce the four processes of DeT in detail.

### 3.1 Preliminary Denoising

We first briefly introduce the details of the reformer used in the process of preliminary denoising.

**Reformer.** The reformer is a function that reconstructs the input example throughout denoising it. The reconstructed example is very close to the original example only with several details lost (these details are regarded as noise by the reformer). Usually, the reformer is built upon the autoencoder.

Given an input example $x_t$, DeT first adds Gaussian noises to the example for obtaining a noised example $x'_t$. This process is helpful for the reformer to reconstruct examples, since it amplifies noises to the same level as the noises used to train our reformer. Then, we use the reformer to preliminarily denoise $x'_t$ for obtaining a relatively clean example $x''_t$. This operation can eliminate potential threats.

### 3.2 Random Sampling

After preliminary denoising, we obtain a relatively clean example $x''_t$. However, we cannot guarantee that $x''_t$ is not an adversarial example. There still exists the possibility that $x''_t$ is near the decision boundary. Therefore, inspired by the Region-based method, we randomly sample a set of examples in a hypercube centered at $x''_t$ and can get an example set $X_t = \{x''_i | i = 1, 2, ...\}$. The generation of $x''_i$ can be formulated as follows:

$$x''_i = Clip(x''_t + \epsilon * Noise), \tag{1}$$

where $Noise$ denotes the Gaussian noise, and $\epsilon$ is a parameter that controls the side length of the hypercube.

### 3.3 Twice Denoising

To further improve the effectiveness of DeT, we again use several other reformers to form a Reformer List to further denoising. A single reformer can be easily attacked by the adaptive attack. DeT uses several reformers that can prevent the adversarial examples that fool one reformer from fooling the other reformer, since the transferability of adversarial examples among these reformers is relatively low. We believe that the diversity of the reformer architectures in the Reformer List will help reduce the transferability of the adversarial examples and thus enhance the defense capabilities of DeT. Therefore, we employ three autoencoders with different architectures as our reformers. The architectures of the three autoencoders used in this paper are based on the open source implementations[2]. For the ease of reference, we denote the $Reformer\ i$ as $R_i$ in the paper. We also propose an algorithm called cross training to further reduce the transferability of adversarial examples among reformers, as shown in Algorithm 1.

**Cross Training Algorithm.** The input of the cross training algorithm includes the training epoch $e$, the reformer set $R$ with $n$ already trained reformers, the batch size $s$, the training dataset $X$ and the noise length $l$. The output is a reformer set $R'$ corresponding to $R$. Note that all the reformers in the input reformer set are already trained and can reconstruct examples well.

In the following, we show several examples, based on two re-trained reformers $R'_1$ and $R'_2$

[2] https://github.com/Trevillie/MagNet.

**Algorithm 1:** Cross Training

**Input**: e: Training epoch.
s: The size of a batch of data.
X: Training dataset.
R: $[R_1,R_2,...,R_n]$
l: The noise strength added to the clean example.
**Output**: R'=$[R'_1,R'_2,...,R'_n]$

$sub\_list = Combinations(0, n-1, 2)$;
**while** *epoch less than e* **do**
  $train_x, train_y = GenetrateData(X)$;
  $temp_x = Clip(train_x + l * random(s))$;
  **for** *p in sub_list* **do**
    $p0 = P[0], p1 = p[1]$;
    $X_1 = R[p0](temp_x)$;
    $X_{1\_2} = R[p1](X_1)$;
    $Loss1 = EuclideanDistance(X_{1\_2}, train_x)$;
    $Train1 = Optimizer.min(Loss1, R[p1])$;
    $Run(Train1)$;
    $X_2 = R[p1](temp_x)$;
    $X_{2\_1} = R[p0](X_2)$;
    $Loss2 = EuclideanDistance(X_{2\_1}, train_x)$;
    $Train2 = Optimizer.min(Loss2, R[p0])$;
    $Run(Train2)$;
  **end**
  $epoch+ = 1$
**end**
**Result**: $R'=[R'_1,R'_2,...,R'_n]$

Figure 2 illustrates reconstructed examples on MNIST. From Fig. 2, we can see that $R'_1$ makes the outline of the input example thicker while $R'_2$ makes the outline of the input example thinner, which are equivalent to expansion and corrosion operations in image processing, respectively. Intuitively, $R'_1$ and $R'_2$ deal with the image in two opposite ways, and thus the adversarial perturbations for one reformer can be removed by the other.

Figure 3 also shows two adversarial examples reconstructed by our two other reformers trained on CIFAR. As we can see from Figs. 2 and 3, the cross training algorithm can re-train reformers in two different directions.

We do not explicitly specify the change directions in the cross training. Interestingly, it is the cross training algorithm that automatically searches for two different directions of the change.

We select several reformers from $R'$ to form the Reformer List. Assume there are $k$ re-trained reformers in the Reformer List, $R'_1$, $R'_2$,..., and $R'_k$. We use $R'_i$ $(i = 1, 2, ..., k)$ to reconstruct all the examples in $X_t$ and get a reconstructed example set denoted by $X_{ti}$. Up to now, we get $k$ reconstructed example sets, $X_{t1}$, $X_{t2}$, ..., and $X_{tk}$.

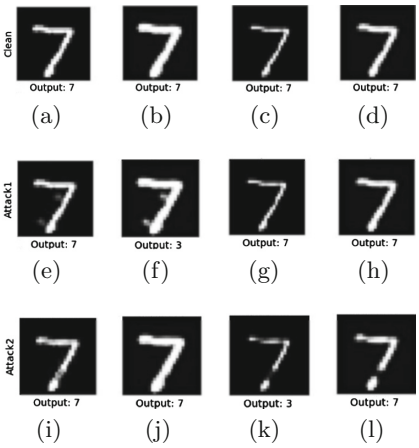



**Fig. 2.** Examples reconstructed by $R_1'$ and $R_2'$. (a)–(d) indicate that the input example is a clean example, (e)–(h) indicate that the input example is an adversarial example for $R_1'$, and (i)–(l) indicate that the input example is an adversarial example for $R_2'$. (a), (e), where (i) presents the input example itself; (b), (f), and (j) present the input examples processed by the $R_1'$; (c), (g), and (k) present the input examples processed by the $R_2'$; (d), (h), and (l) present the examples processed by the $R_1$. The number below each picture presents the classification result.

**Fig. 3.** The first image is a clean image, and the latter two are the images reconstructed by the two reformers re-trained on CIFAR.

### 3.4 Prediction Voting

For a reconstructed set $X_{ti}$ $(i = 1, 2, ..., k)$, we feed all the examples in it to the classifier. The most frequent label is selected as the predicted $y_i$ label for this set. After this, we can get a label set $Y = [y_1, y_2, ..., y_k]$. Finally, the most frequent label in $Y$ is deemed as the final output. If several labels in $Y$ have the same highest frequency, we randomly select one from them as the final output.

## 4 Evaluation

In this section, we first describe the experimental setup. After that, we evaluate the effectiveness of cross training, test DeT against black and gray box attacks, and compare DeT with the state-of-the-arts.

### 4.1 Setup

**Datasets:** We employ two commonly used datasets: MNIST and CIFAR. MNIST, includes 55,000 training examples, 5,000 validation examples, and 10,000 testing examples. CIFAR, includes 45,000 training examples, 5,000 validation examples, and 10,000 testing examples.

**Implementation:**

- **Classifier:** On MNIST, we train a classification model (a seven-layer CNN [3]) with the accuracy of 99.43%. On CIFAR, we train a classification model (a Residual Neural Network [7]) with the accuracy of 92.43%.
- **Reformer:** For each of the datasets, we train three reformers using three different structures. Note that we train all the reformers on the noise-added examples and the reconstruction error (i.e., the $L^2$ distance between the reconstructed example and the clean example) is used to optimize the reformer.
- **Cross Training:** We re-train the reformers according to the cross training algorithm. After the re-training process, we get nine reformers for each dataset. Then, we select a few reformers to form the Reformer List. Our selection strategy are according to: (1) the selected reformer does not have much influence on the accuracy of the classifier after the input reconstruction; and (2) the transferability of adversarial examples among the reformers should be as low as possible.

**Metrics:** To evaluate DeT and compare it with other defense methods, we adopt the following evaluation metric named *Accuracy*.

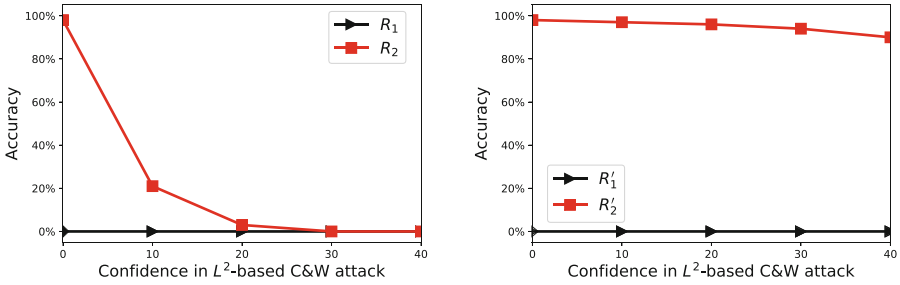
$$Accuracy = \frac{Number\ of\ correctly\ classified\ examples}{Total\ number\ of\ input\ examples},$$

*Accuracy* measures the proportion of correctly classified examples in the input examples.

All experiments in this paper are run on a server with 2 Intel Xeon E5-2640 V4 GPUs, 64 GB memory, 4 TB HDD and 1 GeForce GTX-1080TI GPU.

### 4.2 Effectiveness of Cross Training

Figure 4 shows the classification accuracy comparison of the classifier equipped with the re-trained and the original reformers under $L^2$-based C&W attack. As can be seen from Fig. 4(a), the adversarial examples fool the classifier with $R_1$ perfectly. While feeding these examples to the classifier with $R_2$, the classification accuracy decreases from 99% to 0% as the $Confidence$ increases from 0 to 40. It can be seen from Fig. 4(a) that the transferability of adversarial examples becomes stronger as the $Confidence$ increases. Even when $Confidence$ is greater than 30, adversarial examples can also attack $R_2$ perfectly. Figure 4(b) shows the classification accuracy of the classifier equipped with re-trained reformers. From this figure, we can see that when $Confidence$ is 40, the classifier equipped with $R_2'$ still achieves the classification accuracy of 89%. The above results suggest that our algorithm can significantly decrease the transferability of adversarial examples among reformers.




(a) Without re-trianed (b) With re-trianed

**Fig. 4.** The classification accuracy on MNIST under attack, where the adversarial example in (a) is generated for $R1$, and the adversarial example in (b) is generated for $R1'$.

### 4.3 Evaluation Against Black Box Attacks

**Table 1.** Classification accuracy.

| Dataset | Accuracy (no defense) | Accuracy (with DeT) |
|---|---|---|
| MNIST | 99.43% | 98.6% |
| CIFAR | 92.43% | 86.7% |

**Performance of DeT.** Table 1 shows the decrease of the classification accuracy, which is within our tolerance range. Then, we adopt four common attacks, i.e., Deepfool, BIM, FGSM, and C&W, to evaluate DeT. Among them, Deepfool, BIM and FGSM are implemented based on the Cleverhans library [18], and C&W is implemented based on the open source code [3].

On MNIST, the classification performance with and without DeT against common attacks are shown in Table 2. It is obvious that the classifier protected by DeT outperforms that without any protection. Especially, the classification accuracy of the classifier without defense is 0.0% while it can still obtain a 96%+ classification accuracy with DeT's protection.

On the CIFAR dataset, the classification accuracy with and without DeT against common attacks are shown in Table 2. We can observe that the accuracy of the classifier with DeT has been significantly improved compared to that without any protection.

### 4.4 Evaluation Against the Gray Box Attack

Following Meng et al. [16], in this section, we evaluate DeT against gray box attacks under $L^2$-based C&W attack.

**Table 2.** The classification accuracy of the classifier without defense v.s. with DeT.

| Attack | No defense | with DeT |
|---|---|---|
| *MNIST* | | |
| FGSM $L^\infty$ $\epsilon = 0.1$ | 90.0% | 97.95% |
| BIM $L^\infty$ $\epsilon = 0.05$ | 93.05% | 98.45% |
| BIM $L^\infty$ $\epsilon = 0.1$ | 64.85% | 97.85% |
| DeepFool $L^\infty$ | 0.85% | 97.8% |
| C&W $L^2$ $Confidence = 0$ | 0.0% | 98.44% |
| C&W $L^2$ $Confidence = 20$ | 0.0% | 96.79% |
| *CIFAR* | | |
| FGSM $L^\infty$ $\epsilon = 0.01$ | 29.2% | 78.8% |
| BIM $L^\infty$ $\epsilon = 0.005$ | 12.95% | 82.1% |
| BIM $L^\infty$ $\epsilon = 0.01$ | 5.15% | 78.2% |
| DeepFool $L^\infty$ | 4.55% | 86.1% |
| C&W $L^2$ $Confidence = 0$ | 0.0% | 86.4% |
| C&W $L^2$ $Confidence = 10$ | 0.0% | 85.6% |

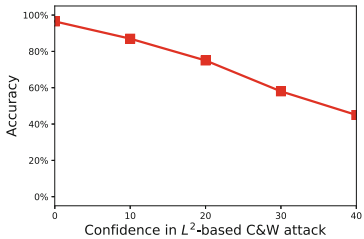



**Fig. 5.** The accuracy of DeT under the gray-box attack.

Figure 5 shows the gray box attack results, from which we can see that DeT can efficiently defend against gray box attack. Even when the adversarial examples are generated with $Confidence = 40$, the classifier with DeT still achieves 43.7% accuracy. The decrease of transferability in this experiment is not as significant as that in Fig. 4. We speculate that it may because we first send the adversarial example to the preliminary denoising. How to improve this is an interesting future work of DeT.

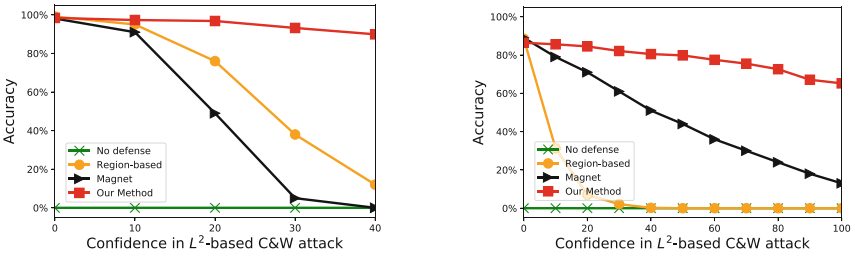
### 4.5 Comparison with the State-of-the-Art Defense

Since $L^2$-based C&W attack is one of the strongest attacks at present, we compare with other defenses under this attack. We conduct comparative experiments on the MNIST and the CIFAR datasets. Figure 6 illustrates the comparison between DeT and the state-of-the-art defenses.

**Magnet.** We find that the defense capability of Magnet depends heavily on the detectors. Magnet can detect an adversarial example with slightly large adversarial perturbations. It will not send the example to the classifier, and therefore, will not produce the correct classification. Magnet's classification accuracy is shown in the black lines of Fig. 6. It can be found that with the increase of $Confidence$, the classification accuracy of Magnet is significantly reduced. Notably, according to Fig. 6(a), on the MNIST dataset, when $Confidence = 40$, the classification accuracy of Magnet reduced to near 0% while DeT can still achieve the accuracy of around 84%. According to Fig. 6(b), on the CIFAR dataset, Magnet can only reach the accuracy of 44% when the $Confidence = 50$, and achieve the accuracy of about 13% when $Confidence = 100$. On the other hand, DeT can reach the accuracy of 80% when $Confidence = 50$, and we can still achieve the accuracy of about 65% when $Confidence = 100$. These results show that our classification accuracy has improved significantly compared to Magnet.

**Region-Based Classification.** For Region-based classification defense methods [1], the basic assumption is that *the adversarial examples exist near the decision boundary*. Our experimental results show that this assumption is not necessarily true.

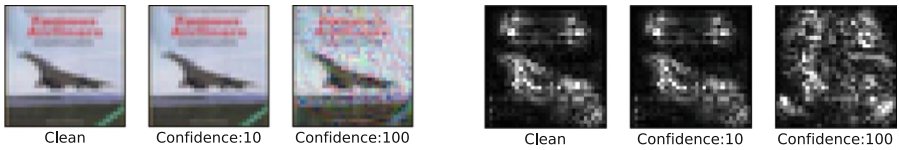
**Exploring DeT's Limitation.** As discussed above, the increase of adversarial perturbations leads to the decrease of DeT's performance. We conjecture that such decrease might be resulted from the key pixel changes caused by large adversarial perturbations. In this section, we use Saliency Maps [22] to experimentally verify our conjecture.




(a) The accuracy on MNIST datatset (b) The accuracy on CIFAR datatset

**Fig. 6.** Classification accuracy comparison of DeT and other defenses against the $L^2$-based C&W attack.

Figure 7(b) shows our experimental results, where the ground truth label is "Airplane". We can see that (1) key pixels of the clean example are the outline of the airplane, and (2) key pixels of the adversarial example with $Confidence = 10$ are very close to the former. Therefore, our proposed method, DeT, removes the adversarial perturbations through the reformer. However, for the adversarial example with $Confidence = 100$, we can see from the third image in Fig. 7(b)




(a) Original Image and Adversarial Images (b) Saliency Maps of the Imgae

**Fig. 7.** (a) One clean example and two adversarial examples with different perturbations generated by the $L^2$-based C&W attack. (b) Saliency Maps corresponding to the images in (a).

that the outline of the aircraft is completely invisible. We have sufficient reasons to believe that many key pixels have already been changed, and it is difficult to classify it correctly.

**Table 3.** The proportion of the adversarial examples with human imperceivable perturbations and the estimated best accuracy under each $Confidence$. The classification accuracy of DeT on clean examples is 86.7%.

| Confidence | Proportion | Best accuracy |
|---|---|---|
| 0.0 | 99.7% | 86.4% |
| 10.0 | 99.2% | 86.0% |
| 20.0 | 98.7% | 85.5% |
| 30.0 | 97.4% | 84.4% |
| 40.0 | 95.5% | 82.8% |
| 50.0 | 93.7% | 81.2% |
| 60.0 | 91.2% | 79.0% |
| 70.0 | 89.1% | 77.2% |
| 80.0 | 85.3% | 73.9% |
| 90.0 | 80.4% | 69.7% |
| 100.0 | 78.4% | 67.9% |

Then we do another statistical experiment to count the adversarial examples with human imperceivable perturbations under different $Confidence$. We first randomly sample 1000 examples from the CIFAR dataset. Then, we invite an expert in adversarial machine learning to label how many examples with human imperceivable perturbations under each $Confidence$. Table 3 lists the proportion of the adversarial examples with human imperceivable perturbations under each $Confidence$.

We conclude the DeT can correctly classify most of the adversarial examples. The decrease in accuracy is may due to the fact that those adversarial perturbations are perceptible to the humans while not meeting the definition of adversarial examples.

## 5 Conclusion

In this paper, we propose DeT, which obtains high accuracy in both scenarios of classifying adversarial examples with small and large perturbations. We also propose an algorithm that can significantly reduce the transferability of adversarial examples among different reformers. In the future work, we will consider to improve the robustness of the model by further reducing the transferability of adversarial examples.

Our method also has some limitations. The first is the efficiency problem, which can be solved by adding hardware at this moment. In addition, our reformer is trained under Gaussian noises. For some of the adversarial noises that differ greatly from Gaussian noises, they cannot be denoised by the reformer, which also needs to be resolved in the future.

In general, our method significantly improves the robustness of a classifier, and we hope that DeT will be used as a benchmark in the research community for measuring neural network attacks.

**Acknowledgements.** This work was partly supported by NSFC under No. 61772466 and U1836202, the Zhejiang Provincial Natural Science Foundation for Distinguished Young Scholars under No. LR19F020003, and the Provincial Key Research and Development Program of Zhejiang, China under No. 2017C01055.

## References

1. Cao, X., Gong, N.Z.: Mitigating evasion attacks to deep neural networks via region-based classification. In: Proceedings of the 33rd Annual Computer Security Applications Conference, pp. 278–287. ACM (2017)
2. Carlini, N., Wagner, D.: MagNet and efficient defenses against adversarial attacks are not robust to adversarial examples. arXiv preprint arXiv:1711.08478 (2017)
3. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 39–57. IEEE (2017)
4. Du, T., Ji, S., Li, J., Gu, Q., Wang, T., Beyah, R.: SirenAttack: generating adversarial audio for end-to-end acoustic systems. arXiv preprint arXiv:1901.07846 (2019)
5. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. Computer Science (2014)
6. Graves, A., Mohamed, A.R., Hinton, G.: Speech recognition with deep recurrent neural networks. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6645–6649. IEEE (2013)

7. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 630–645. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_38
8. Hinton, G., et al.: Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Process. Mag. **29**(6), 82–97 (2012)
9. Javaid, A., Niyaz, Q., Sun, W., Alam, M.: A deep learning approach for network intrusion detection system. In: Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS), pp. 21–26. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering) (2016)
10. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
11. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533 (2016)
12. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
13. Li, J., Ji, S., Du, T., Li, B., Wang, T.: TextBugger: generating adversarial text against real-world applications. In: NDSS (2019)
14. Li, X., et al.: Adversarial examples versus cloud-based detectors: a black-box empirical study. arXiv preprint arXiv:1901.01223 (2019)
15. Ling, X., et al.: DEEPSEC: a uniform platform for security analysis of deep learning model. In: IEEE S&P (2019)
16. Meng, D., Chen, H.: MagNet: a two-pronged defense against adversarial examples. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 135–147. ACM (2017)
17. Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: DeepFool: a simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2574–2582 (2016)
18. Papernot, N., et al.: Cleverhans v2. 0.0: an adversarial machine learning library. arXiv preprint arXiv:1610.00768 (2016)
19. Papernot, N., McDaniel, P., Goodfellow, I.: Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. arXiv preprint arXiv:1605.07277 (2016)
20. Papernot, N., McDaniel, P., Wu, X., Jha, S., Swami, A.: Distillation as a defense to adversarial perturbations against deep neural networks. In: 2016 IEEE Symposium on Security and Privacy (SP), pp. 582–597. IEEE (2016)
21. Shi, C., et al.: Adversarial captchas. arXiv preprint arXiv:1901.01107 (2019)
22. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034 (2013)
23. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
24. Szegedy, C., et al.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
25. Xu, L., Zhang, D., Jayasena, N., Cavazos, J.: HADM: hybrid analysis for detection of malware. In: Bi, Y., Kapoor, S., Bhatia, R. (eds.) IntelliSys 2016. LNNS, vol. 16, pp. 702–724. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-56991-8_51

26. Xu, Y., et al.: Deep learning of feature representation with multiple instance learning for medical image analysis. In: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1626–1630. IEEE (2014)
27. Zhang, X., Wang, N., Shen, H., Ji, S., Luo, X., Wang, T.: Interpretable deep learning under fire. In: USENIX Security (2020)