ReabsNet: Detecting and Revising Adversarial Examples

Jiefeng Chen, Zihang Meng, Changtian Sun, Wei Tang, Yinglun Zhu
University of Wisconsin - Madison
{jiefeng, zihangm, changtiansun, weit, yinglun}@cs.wisc.edu

Abstract

Though deep neural network has hit a huge success in recent studies and applications, it still remains vulnerable to adversarial perturbations which are imperceptible to humans. To address this problem, we propose a novel network called ReabsNet to achieve high classification accuracy in the face of various attacks. The approach is to augment an existing classification network with a guardian network to detect if a sample is natural or has been adversarially perturbed. Critically, instead of simply rejecting adversarial examples, we revise them to get their true labels. We exploit the observation that a sample containing adversarial perturbations has a possibility of returning to its true class after revision. We demonstrate that our ReabsNet outperforms the state-of-the-art defense method under various adversarial attacks.

1 Introduction

As a major technique used in machine learning, deep neural networks have shown good performance in many practical areas, especially in tasks involving pattern recognition and image classification [1], such as image tagging [2], face detecting and verification [3] and video classification [4]. Although employing neural networks is one of the most accurate machine learning approaches, it can be quite vulnerable to adversarial examples whose aim is to attack the classifiers [5]. The *adversarial perturbation* is defined as a test-time attack, which uses perturbed images that are visually the same as their unmodified versions, leading the neural network to generate a totally wrong labeling result. This problem has been shown to exist in almost all domains where the neural networks are used, strongly restricting the reliable applications of neural networks into many security-critical areas.

The research community has devoted much effort to building a neural network that is robust to adversarial examples. There are two major directions in the field of defending neural networks. One focuses on improving the model's robustness against adversarial perturbations itself. Another pays its attention to detecting adversarial examples and preventing them from attacking neural networks.

For the first direction, there exist theories and methods to improve the classification accuracy by strengthening the network structure and robustness (e.g., defensive distillation [6], dropout [7], convex outer adversarial polytope [8]), training the network with enhanced training data [9], etc. There is a possibility that these approaches would face difficulties, since systematic understandings of the network structures and adequate amount of adversarial training data are required [10].

For the second direction, several different tracks for recent research works including training the neural network with an 'adversarial' class [11], training an additional binary classifier [12, 13], detecting adversarial examples based on Maximum Mean Discrepancy (MMD) [11], observing the internal state of the classifier's late layers [10], statistical analysis on Convolution Neural Network (CNN) layers [14], and so on. Such 'detecting and dropping' strategy has a limitation: if the detector misclassifies the natural images as adversarial ones, then these images are dropped. What's more, according to the definition that adversarial examples are samples that are closed to the natural ones

Final project for CS 760 (Fall 2017), University of Wisconsin-Madison.

but misclassified by the neural network, they should have the same labels with the corresponding natural ones. Thus simply rejecting them is not appropriate.

To address these issues, we propose a reabsorption scheme and present a novel network called ReabsNet which can output a correct label for every input, even the adversarial ones. We build a *guardian network* which can efficiently detect adversarial examples, and a modifier to modify detected adversarial examples iteratively until they are considered 'natural' by the guardian network. In the end, we enter all the natural and modified examples to the master network [15] for classification. The whole process resembles the physiology term *reabsorption* [16] that nutrient substances are reabsorbed from the tubule into the peritubular capillaries even if they are dropped due to various reasons at first.

Our method focuses on developing an effective defense mechanism to adversarial examples, as well as building a strong classification system. Our main contributions are the following:

- We empirically prove the existence of a guardian network that can detect adversarial examples effectively.
- We propose an image modification method based on attacking mechanisms, which is able to turn adversarial examples to natural ones so that a master network could correctly classify them.
- We perform an extensive experimental evaluation, showing that our method can successfully defend some existing attacks and outperform the state-of-the-art defense method.

2 Background

Upon the demonstration of the idea *adversarial examples* [5], the study of these phenomena and the methods of dealing with this problem have been showing an uptrend. Given an input and its prospective label, after making perturbations to the input in a way where human eyes cannot tell the difference, it is possible that the classification result is different from the original label.

Currently, multiple kinds of attacks can achieve this goal to different extents, such as *Fast Gradient Sign Method* [5], *DeepFool* [17], L_2*, L_0 and L_∞ attack* [18] and so on. And these methods will be introduced briefly in section 3.2.

There are two general directions in tackling the problem. The first is to develop a network that is robust to one or several kinds of attacks. One example in this category is to use the convex outer adversarial polytope [8]. This approach provides a method to train classifiers based on linear programming and duality theory. It uses a dual problem technique to represent the linear program as a deep network similar to back-propagation network. A decent proof is provided to show that this algorithm is robust to any norm-bounded adversarial attack. However, a small flaw in this method is that although it can be guaranteed to detect all the adversarial examples, some non-adversarial examples may also be misclassified. In addition, it is difficult to adjust this solution to a more complicated network due to high complexity of the algorithm.

In Madry et al.'s work [19], they tried to propose a concrete *guarantee* that a robust model should satisfy, and adapt their training methods towards it. The major steps involve specifying an attack model and raising a natural saddle point formulation. The resulting trained networks are shown to be robust against a wide range of attacks. They demonstrated that the saddle point problem can be solved eventually, but might not be in a reasonable time duration.

Another way of defense is to attach additional protection scheme to existing neural network models. Metzen et al. came up with the idea of adding an additional subnetwork specialized in detecting adversarial examples [13] to ResNet [20]. Lu et al. designed a RBF-SVM based detector called SafetyNet with inputs of codes, of original/adversarial examples, in late ReLU layers of the main classification network, and this approach further enhances the detecting ability. [10] These are truly novel ideas in avoiding attacks by adversarial examples by using a detector.

Apart from attaching networks for detecting adversarial examples, preprocessing the data samples are adopted in a manifold scheme [21]. This scheme integrates protection into classification by using MCN or NCN model shells. Although further proof of the common manifold assumption is needed, this approach provides an advanced idea of preprocessing all the data points.

In our method, we adopted the second way of attaching additional protection scheme by using a guardian network. Different from the methods mentioned above, we make amendments for all the detected adversarial examples rather than simply dropping them.

3 Method

In this section, we briefly introduce the definition of adversarial examples and some adversarial attacks used in the experiment. Based on the defensive methods mentioned in Section 2, we further propose our novel scheme for correctly classifying adversarial perturbations.

3.1 Problem Definition and Notation

Suppose we have a classifier $f_\theta(x)$ with parameters θ and outputs of discrete class labels. For a natural example x and its ground truth label y, we want to train a proper θ that can make $f_\theta(x) = y$ stand with high probability. Further, it is reasonable to assume that $f_\theta(x + \epsilon) = f_\theta(x)$ when ϵ is small enough. However, many researchers [22, 5] have noticed small perturbations added to natural examples can confuse classifiers with high confidence and also in an incomprehensible way, specifically

$$f_\theta(x^{adv}) = y^{adv} \neq y \text{ s.t. } ||x^{adv} - x||_p \leq \epsilon \tag{1}$$

Here, x^{adv} is an adversarial example to the classifier $f_\theta(x)$. If the class y^{adv} is pre-specified by adversary, then we call x^{adv} a *targeted* adversarial example; otherwise, x^{adv} can be viewed as an *untargeted* adversarial example.

3.2 Adversarial Attacks

In this work, we consider three approaches for generating adversarial examples:

Fast Gradient Sign Method [5]: This method is based on the assumption that neural networks are too linear to resist linear adversarial perturbation. Thus, the perturbation can be created using the sign of the elements of the gradient of the cost function with respect to the input. Adversarial examples can be given by:

$$x^{adv} = x + \epsilon \text{sign}(\nabla_x J(\theta, x, y)) \tag{2}$$

where $J(\theta, x, y)$ as the cost function of the classifier, θ is the parameters of the model, x is the input, y is the expected label and ϵ is the step size.

CW L_2, L_0 and L_∞ attack [18]: It is a general way to quantify similarity with an L_p norm form distance metric when defining adversarial examples. Given an image x, the L_p attack aims to find a different image x^{adv} that is similar to x under L_p distance, yet is given a totally different label by the classifier. In this paper, L_2 and L_∞ attacks are used for experiment.

For L_2 attack, given a sample x and a target class t, we search for w that solves:

$$\begin{gathered} \text{minimize } \|\frac{1}{2}(tanh(w) + 1) - x\|_2^2 + c \cdot f(\frac{1}{2}(tanh(w) + 1)) \\ \text{with } f \text{ defined as } f(x') = \max(\max\{Z(x')_i : i \neq t\} - Z(x')_t, -\kappa) \end{gathered} \tag{3}$$

Here f is based on the best objective function and the value of κ is tried out for controlling the confidence.

For L_∞ attack, an iterative attack is used and in each iteration, we need to solve

$$\text{minimize } c \cdot f(x + \delta) + \sum_i [(\delta_i - \tau)^+] \tag{4}$$

Where δ_i is a penalization parameter in the ith iteration. After each iteration, if $\delta_i < \tau$, τ needs to be reduced to 0.9 of its current value; otherwise, the search should be terminated.

DeepFool [17]: This is an untarget attacking technique optimized for the L_p distance metric. When implementing DeepFool, one can imagine that the neural networks are completely linear. Then the optimal solution towards this simplified problem can be derived iteratively and adversarial examples are thereby generated. This searching algorithm of DeepFool is greedy and it stops once the adversarial example crosses the classification boundary. Usually, DeepFool will generate adversarial examples closest to the original ones.

3.3 Master Network

Master network is used to assign correct labels to input examples. The master network can just be a deep neural network. To get better performance in the master network, we also equip it with defensive distillation.

Here we briefly introduce the defensive distillation method [6]. Distillation [15] is a training procedure in which a softmax layer is added to the original DNN. This softmax layer considers the vector $Z(X)$ output by the last hidden layer and normalizes it into a probability vector $F(X)$ as the ultimate output of the whole network. The vector $F(X)$ assigns a probability to each class of the dataset for input X. Suppose there are N classes in total, the output of neuron i can be represented as the following:

$$F_i(X) = \left[\frac{e^{z_i(X)/T}}{\sum_{l=0}^{N-1} e^{z_l(X)/T}} \right]_{i \in \{0 \dots N-1\}} \tag{5}$$

Here, T is the *temperature* parameter. The probability vectors produced by the new DNN are used as a *soft labels*. The defensive distillation method just uses the same network architecture to train both the original network and the distilled network.

3.4 Guardian Network

We design a guardian network to detect adversarial examples from the natural ones. Before describing our guardian network elaborately, we first make the following assumption as the basis of the guardian network.

Assumption: Distributions of natural examples and adversarial ones are statistically different, and this difference can be learned by the guardian network.

We design a two-category deep neural network, with both natural and adversarial examples as input, to be our guardian network. Suppose x is an input example, we will get an output $G(x) \in [0,1]^2$ in the softmax layer showing the probability of classifying x as a natural example and an adversarial example. If the probability of an input being an adversarial example is higher, our guardian network will put it into a modifier that can turn it back to a 'natural' example. Otherwise, our guardian network will send it into the master network directly. Since it is very difficult for adversarial examples to be both misclassified by the master network and the guardian network, they together form a very reliable classifier which can classify not only natural examples, but also adversarial examples.

To train this guardian network, we first need to train our master network with natural examples. Subsequently, we generate adversarial data for each natural examples in the training dataset through one of the attacking methods discussed in Section 3.2. Finally, we train our guardian network with a balanced binary classification dataset of twice the size of the original one.

3.5 Revising Adversarial Examples

Our guardian network mentioned in Section 3.4 can detect many possible adversarial examples. In previous approaches (e.g., the SafetyNet [10]), people just prevent these adversarial examples from entering the main classification network. For improvement, we move forward by modifying the adversarial examples and then enter the *revised adversarial examples* into the master network.

With the assumption in Section 3.4, a good guardian network can be obtained. If the guardian network reckons an input as a natural example, the master network should be able to classify it correctly. Thus,

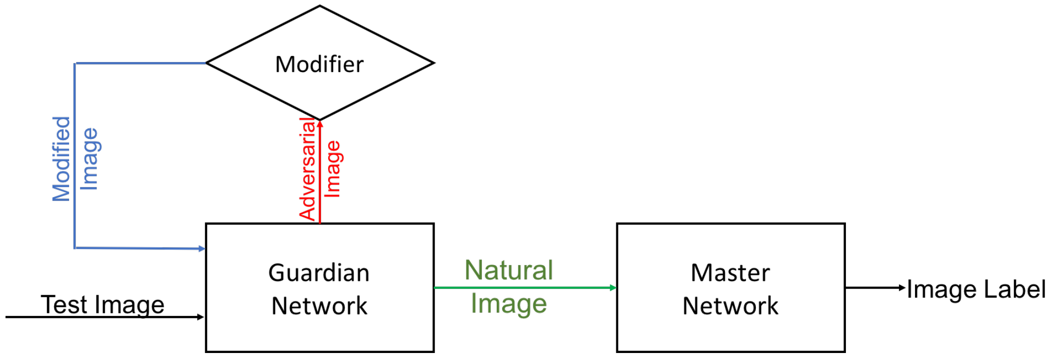

Figure 1: Structure of ReabsNet: there are three parts in our ReabsNet, the guardian network, the modifier and the master network. When a test image is received by ReabsNet, it is first examined by the guardian network: if it is classified as an adversarial image, it will be sent to the modifier; if it is classified as natural image, it will be sent to the master network for classification. The job of modifier is, for each adversarial image it received from guardian network, it amends the image iteratively until the modified image can pass the guardian network, i.e., be classified as natural image by the guardian network.

we can slightly modify the adversarial example so that the guardian network may classify it as a natural example. Afterwards, it will be entered into the master network. This modification method can be the one of the attacks described in Section 3.2. We show below why this method can work.

Suppose x is the original natural example, and x^{adv} is the corresponding adversarial example detected by the guardian network. With the property of attack under p norm, we have

$$||x - x^{adv}||_p \leq \epsilon_1 \tag{6}$$

We further suppose x^{ame} is the amendment of the adversarial example x^{adv}, which means that x^{ame} is viewed as natural example by the guardian network. Based on the assumption, it is likely that the master network can classify x^{ame} correctly. Besides, we have

$$||x^{ame} - x^{adv}||_p \leq \epsilon_2 \tag{7}$$

Combining equation (6) and equation (7), we have

$$||x^{ame} - x||_p \leq ||x^{ame} - x^{adv}||_p + ||x - x^{adv}||_p \leq \epsilon_2 + \epsilon_1 \tag{8}$$

In equation (8), although ϵ_1 is determined by the attackers, we can choose our modification strategy to control ϵ_2. In order to control the distance between x^{ame} and x in worst case, DeepFool [17] can be used here. In this way, after amending the adversarial example, we obtain a natural example lying very close to the original one. Thus the label of x^{ame} should be the same as x. Hence, through revising the adversarial with respect to the guardian network, we find a way to correctly classify adversarial examples, and this can be viewed as a novel defending mechanism.

3.6 Reabsorption Network

We name our entire model as *Reabsorption Network* (shortened as *ReabsNet*). Because our classification process resembles the renal physiological process *Reabsorption* in which nutrient substances like glucose and amino acids are reabsorbed after they are dropped at first. In our method, we actually revise adversarial examples into natural ones and then re-classify them rather than simply 'drop them out of the body'.

The structure of our model is demonstrated in figure 1. The model consists of a guardian network, an image modifier and a master network that is used for classification. If the guardian network concludes an image as an adversarial image, this sample is then modified to become 'natural'. The image can be sent to the master network for classification only when it seems natural to the guardian network.

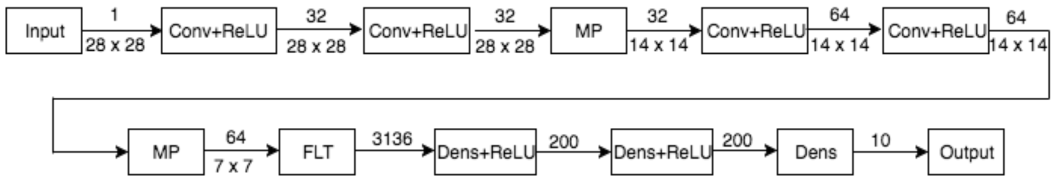

Figure 2: Structure of Master Network. In this graph, 'Conv+ReLU' stands for a convolutional layer with a ReLU activation layer. 'MP' stands a for Max-Pooling layer. 'Dens' denotes a fully-connected layer, and 'Dens+ReLU' stands for a fully-connected layer with a ReLU activation layer. 'FLT' denotes a flattening operation. The numbers on top of arrows denote the numbers of feature maps, and those below arrows denote spatial resolution.

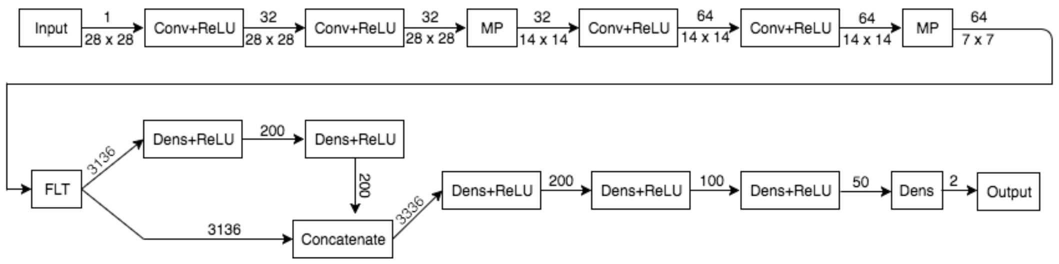

Figure 3: Structure of Guardian Network. 'Concatenate' is a step that concatenates features from two steps together. The other symbols and notations have the same representations as figure 2

4 Experiment

4.1 Dataset

The dataset that we use is MNIST. MNIST [23] is a popular benchmark dataset for computer vision tasks. It consists of a training set of $60,000$ examples, and a testing set of $10,000$ examples, where each 28×28 image belongs to one of ten handwritten digits (from 0 to 9). In our experiments, we use $55,000$ examples in the training set to train models and use the remaining $5,000$ examples to validate models. We also normalize each pixel's value to the range of $[-0.5, 0.5]$ before feeding the image into the network.

4.2 Implementing Details

The structure of the master network is showed in figure 2. For the defensive distillation training, we set $T = 100$. The structure of the guardian network is showed in figure 3. We use DeepFool method to attack the master network and generate adversarial examples. Then we use them to train the guardian network. Afterwards, DeepFool method is used again to attack the guardian network so as to turn the adversarial examples back to the natural images. In consideration of the computing time, we generate 1000 possible adversarial examples with FGSM, DeepFool, CW L_2 attack (targetedly/untargetedly) respectively, and generate 100 possible adversarial examples using CW L_∞ attack (targetedly/untargetedly) to calculate the defense success rate under these attacks.

4.3 Results

To evaluate the performance of our network against adversarial perturbations, we conduct experiments on MNIST and report results under several common effective adversarial attack methods.

4.3.1 Performance of Guardian Network

Before discussing the whole network's performance, we first report the performance of our guardian network. Our guardian network can detect adversarial examples with high success rate on MNIST.

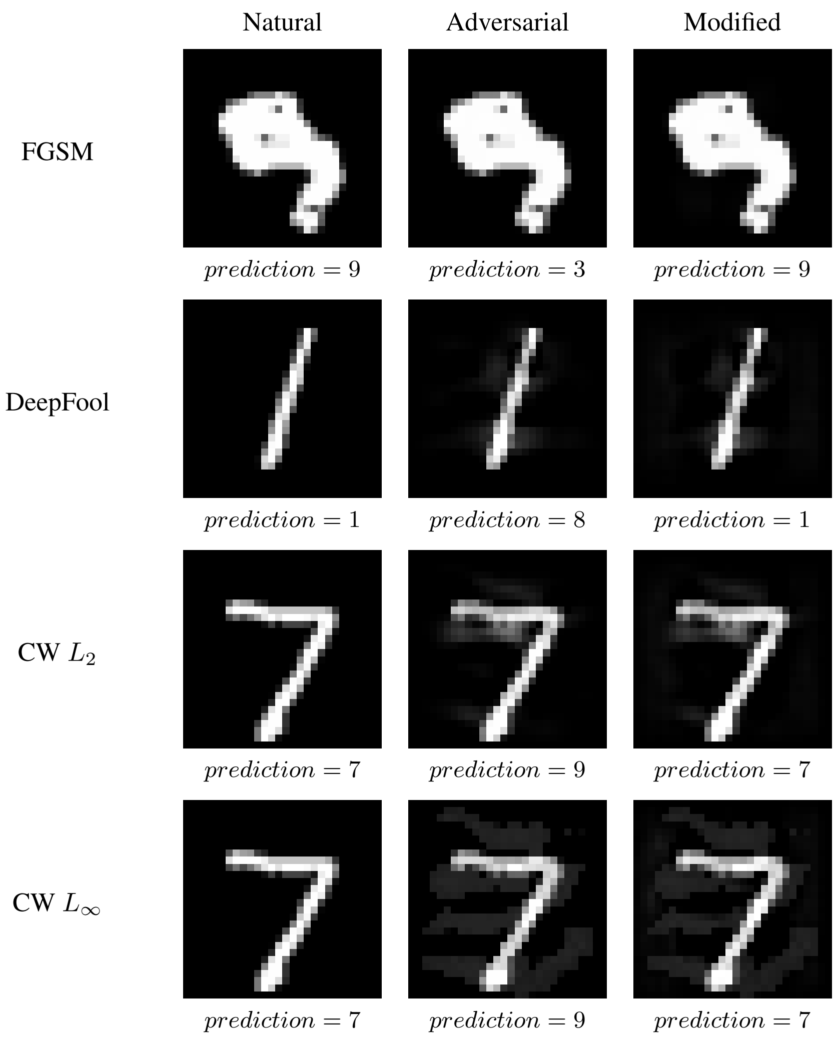

Figure 4: Examples of natural, adversarial and modified-adversarial MNIST images. Adversarial images are generated by FGSM, DeepFool, $CW\ L_2$ and $CW\ L_\infty$ respectively. The prediction of each image by our ReabsNet is marked at the bottom of the image. After our image modification step, the modified-adversarial examples can be predicted correctly again.

Table 1 shows the guardian network's performance on various attack methods and under a non-attack situation.

The experimental results show that our guardian network is able to learn the boundary between natural examples and adversarial ones and testify that the distributions of these two are different. In addition, the efficiency of our guardian network is critical to the further modification and classification steps.

Attack Method	Non-Attack	DeepFool	L_2 Untargeted	L_2 Targeted	L_∞ Untargeted	L_∞ Targeted
Detect Success Rate	0.9926	0.980	0.983	0.999	0.99	1.0

Table 1: Guardian net's success rate on detecting adversarial images generated from different attack methods. Note: we don't include the result of FGSM attack because, we only have 7 adversarial test images generated from FGSM, which means the resulted success rate might not be accurate.

Attack Method	**Defense Success Rate**	
	Master Network Only	**ReabsNet**
FGSM	0.998	1.0
DeepFool	0.0	0.709
CW L_2 Untargeted	0.001	0.983
CW L_2 Targeted	0.001	0.962
CW L_∞ Untargeted	0.0	0.99
CW L_∞ Targeted	0.0	0.95

Table 2: Defense success rate on adversarial images generated from different methods. We don't set a distance bound here because the attack algorithms will try to find the minimum distance between the natural image and the adversarial one, and the distance metrics that different attack methods use may vary. The defense success rate is evaluated on six different attack methods: FGSM, DeepFool, CW L_2 Untargeted, CW L_2 Targeted, CW L_∞ Untargeted and CW L_∞ Targeted. And the table shows the resulted defense success rates under the cases of using the master network only and using the whole ReabsNet, respectively.

4.3.2 Performance of ReabsNet

Our network achieves high classification accuracy under several common effective attack methods on MNIST. Both our master network and ReabsNet can achieve 99.05% classification accuracy on natural images. Table 2 shows that our ReabsNet still achieves high classification accuracy under various attacks and is prominently better than the model (our master network) with only defensive distillation training. Some examples of natural, adversarial and modified adversarial images can be seen in figure 4.

The results testify that an adversarial example which is misclassified by the master network has the possibility of returning to the true class after certain modification. Since we use DeepFool to generate adversarial examples for training the detector and also as the modification method under all the different attack methods, it demonstrates the generalization ability of our network across different attacks. In this way, at training time, we don't have to know what attacks will occur at test time, which is exactly the case in reality.

4.3.3 Comparing with Madry's Model

We further use a state-of-the-art defense method described in Madry's paper [19] as our baseline to evaluate the performance of ReabsNet.

Madry et al. trained a robust network on the MNIST dataset based on the method described in their paper [19]. They also posted a MNIST Adversarial Examples Challenge [24] to allow others to attack their model. From the leaderboards, we can see their model is very robust to adversarial attacks. However, their network is trained against an iterative adversary that is allowed to perturb each pixel value (in the range of $[-0.5, 0.5]$) by at most $\epsilon = 0.3$. In consideration of fairness, when comparing the results, we also set a L_∞ distance bound in the attack algorithms: if the L_∞ distance between the original image and the adversarial image found is larger than ϵ, we say the attack algorithms fail to find a valid adversarial example and replace the adversarial image with the original one.

	Defense Success Rate											
	$\epsilon < 0.3$						$\epsilon < 0.6$					
	FGSM	**DF**	L_2 **U**	L_2 **T**	L_∞ **U**	L_∞ **T**	**FGSM**	**DF**	L_2 **U**	L_2 **T**	L_∞ **U**	L_∞ **T**
Our Model	1.0	0.92	0.984	0.996	0.99	0.95	1.0	0.744	0.983	0.973	0.99	0.95
Madry's Model	1.0	0.968	0.962	0.997	0.93	0.98	1.0	0.946	0.582	0.226	0.15	0.43

Table 3: Compare Defense Success Rate with Madry's Model, using different attack methods and under perturbation scales $\epsilon < 0.3$ and $\epsilon < 0.6$. In this table, DF stands for the DeepFool attack, $L_2\ U$ stands for the CW L_2 norm untarget attack, $L_2\ T$ stands for the CW L_2 norm Targeted attack, $L_\infty\ U$ stands for the CW L_∞ norm untarget attack, and $L_\infty\ T$ stands for the CW L_∞ norm Targeted attack.

The results of experiments on our ReabsNet and Madry's model under various attacks appear in Table 3. Our model is comparable with Madry's model under the restricted condition they specify, and tends to be consistently better when the perturbation scale is relaxed beyond 0.3, where the classification task becomes harder.

5 Discussion

In this paper, we have described a novel network, ReabsNet, with high classification ability that can correctly classify both natural and adversarial examples. It can detect and classify the adversarial examples from attacking methods not seen in the previous training process. Despite good performance on the attacks to master networks, we have not addressed the situation where the attacker also knows the architecture and parameters of the guardian network and then attack the master network and guardian network at the same time. We leave this to future work. Finally, we believe that our technique of leveraging the guardian network and modifier could be applied to help understand the distributions of natural and adversarial examples, which could be an interesting direction for the future work.

Acknowledgments

We thank Prof. Yingyu Liang for his insightful instruction and providing us with such a great opportunity to explore more in the field of machine learning.

References

[1] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 427–436.

[2] X. Li, T. Uricchio, L. Ballan, M. Bertini, C. G. Snoek, and A. D. Bimbo, "Socializing the semantic gap: A comparative survey on image tag assignment, refinement, and retrieval," *ACM Computing Surveys (CSUR)*, vol. 49, no. 1, p. 14, 2016.

[3] O. M. Parkhi, A. Vedaldi, A. Zisserman *et al.*, "Deep face recognition." in *BMVC*, vol. 1, no. 3, 2015, p. 6.

[4] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4694–4702.

[5] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[6] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 2016, pp. 582–597.

[7] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," *arXiv preprint arXiv:1703.00410*, 2017.

[8] J. Z. Kolter and E. Wong, "Provable defenses against adversarial examples via the convex outer adversarial polytope," *arXiv preprint arXiv:1711.00851*, 2017.

[9] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae, and S. Ishii, "Distributional smoothing with virtual adversarial training," *arXiv preprint arXiv:1507.00677*, 2015.

[10] J. Lu, T. Issaranon, and D. Forsyth, "Safetynet: Detecting and rejecting adversarial examples robustly," *arXiv preprint arXiv:1704.00103*, 2017.

[11] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE, 2016, pp. 372–387.

[12] Z. Gong, W. Wang, and W.-S. Ku, "Adversarial and clean data are not twins," *arXiv preprint arXiv:1704.04960*, 2017.

[13] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," *arXiv preprint arXiv:1702.04267*, 2017.

[14] X. Li and F. Li, "Adversarial examples detection in deep networks with convolutional filter statistics," *arXiv preprint arXiv:1612.07767*, 2016.

[15] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[16] "Reabsorption." [Online]. Available: https://en.wikipedia.org/wiki/Reabsorption

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

[18] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," *arXiv preprint arXiv:1608.04644*, 2016.

[19] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[21] X. Wu, U. Jang, L. Chen, and S. Jha, "Manifold assumption and defenses against adversarial perturbations," *arXiv preprint arXiv:1711.08001*, 2017.

[22] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[23] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[24] Madry, Aleksander and Makelov, Aleksandar and Schmidt, Ludwig and Tsipras, Dimitris and Vladu, Adrian, "MNIST Challenge," https://github.com/MadryLab/mnist_challenge.