HKU CS Tech Report TR-2018-02

DCN: Detector-Corrector Network Against Evasion Attacks on Deep Neural Networks

Jing Wen
Dept. of Computer Science
The University of Hong Kong
Email: jwen@cs.hku.hk

Lucas C.K. Hui
Hong Kong Applied
Science and Technology
Research Institute (ASTRI)
Email: lucashui@astri.org

Siu-Ming Yiu
Dept. of Computer Science
The University of Hong Kong
Email: smyiu@cs.hku.hk

Ruoqing Zhang
Dept. of Computer Science
The University of Hong Kong
Email: rqzhang2@cs.hku.hk

Abstract—**Deep neural networks are extensively used in image recognition. However, its integrity is compromised by evasion attacks. Attackers can easily craft adversarial examples that make DNNs unknowingly output the labels they want rather than the right labels. In a recent study, it was shown that existing detection methods are not effective in identifying these adversarial examples, i.e., it is a realistic threat to existing systems. Unlike the previous detection methods, we observe that the classification probability distributions of adversarial examples and those of untampered examples exhibit a big difference, which can be easily identified based on the output of a DNN without getting into the complicated DNN internal structure. Based on this new insight, we propose a new light-weight detection method by transforming the detection of adversarial examples into a binary classification problem. The detector we train achieves almost 100% accuracy on adversarial examples. Moreover, we propose a detector-corrector network that effectively reduces successful rate of existing state-of-the-art evasion attacks under three commonly used distance metrics. In particular, for the common L_2 attack, DCN mitigates 99% adversarial examples on MNIST and 95% on CIFAR-10. Our evaluation demonstrates that DCN is significantly more effective and efficient against various evasion attacks than existing methods.**

1. Introduction

Deep Neural Networks (DNNs) have become a popular methodology for handling various challenging AI problems such as speech recognition [1], playing games [2], [3], natural language processing [4] and image recognition [5]. Specifically, some models using DNNs in image recognition achieve near-human performance [5], [6].

However, recent work in machine learning, computer vision and security has revealed the vulnerability of DNNs to evasion attacks at the inference stage. Attackers can add noises onto an image then transform it into an *adversarial example* [7], which is incorrectly classified by a DNN. Often, the change on original image (called an *benign example*) is too small to be noticed by human. Therefore, attackers can take control of DNN model so that it unknowingly outputs what attackers want instead of the right label [8].

Evasion attacks limit the deployment of DNNs, especially in safety-critical applications such as self-driving systems. For instance, a stop sign is slightly modified such that it is still recognized as a stop sign by human but classified as a yield sign by an autonomous car. As a result, the self-driving car may not stop at the adversarial stop sign.

To defend against these attacks, adversarial training [9] and distillation network [10] that reinforce robustness of DNN model, were proposed. Meanwhile, Carlini et al. proposed a set of attacks [11] that achieved 100% success rate on DNNs even with defensive distillation. On the other hand, some research focused on detection of adversarial examples by examining the convolution layers [12] and PCA of image [13]. But [14] demonstrated that previous detection methods failed to detect their new adversarial examples.

Our work. In this paper, we propose a new detection method by transforming the detection of adversarial examples into a binary classification problem. In other words, we train a binary classifier to distinguish adversarial examples. Intuitively, the classifier learns the differences in classification probability distribution, which is the output of DNN. Especially for adversarial examples, the detection accuracy exceeds 99%. This indicates that the classification probability distributions of adversarial examples and benign examples have different characteristics and can be exploited to train a classifier. Based on the measurement result, we implement the *detector* using a two-layer neural network.

Besides, we adopt the idea of Region-based classifier (RC) [15] proposed by Can and Gong. Since this method is very cumbersome, we improve it by setting new parameters and transform it into the *corrector* to recover the right labels of those adversarial examples found by detector.

Finally, we propose a detector-corrector network (DCN) which not only accurately captures adversarial examples, but also efficiently recovers their right labels. We evaluate it against six types of the state-of-the-art evasion attacks and compare it with other defensive methods such as distillation and RC. Evaluation results in Sec. 5 shows our method is significantly more effective and efficient.

Contributions. In summary, this paper makes following contributions:

- We perform a measurement study and provide new insights into characterizing adversarial examples.
- We propose a new efficient and accurate detection method to distinguish adversarial examples with extremely high accuracy and low false negative.
- We propose a detector-corrector network that can recover the right label of adversarial examples to mitigate evasion attacks. The evaluation results demonstrate that our DCN: 1) achieves the same accuracy on benign examples with standard DNN, 2) is more robust to the state-of-the-art evasion attacks than existing methods, 3) is significantly more efficient than existing effective defensive methods.

2. Background and related work

2.1. Deep neural networks

A deep neural network can be simply regarded as a function $C(\boldsymbol{x}) = L$ that accepts $\boldsymbol{x} \in \mathbb{R}^m$ as input and predicts the label $L \in \mathbb{N}$, that refers to the classification. For example, MNIST and CIFAR-10 are both 10-class problems. Therefore label L is an integer from 0 to 9. Furthermore, a DNN consists of an input layer, several hidden layers, and an output layer. For a k-class classification problem, after the input layer and hidden layers, DNN computes a k-dimensional vector denoted as $\boldsymbol{y} = H(\boldsymbol{x}), \boldsymbol{y} \in \mathbb{R}^k$, called *logits*. Since output layer is often a softmax layer, which normalizes $\boldsymbol{y}$ into another k-dimensional vector $\boldsymbol{p} = softmax(\boldsymbol{y})$ of values ranging from 0 to 1 that add up to 1. We donate the ith value in $\boldsymbol{p}$ as $\boldsymbol{p}_i$, which satisfies $0 \leq \boldsymbol{p}_i \leq 1, \sum_{i=1}^{k} \boldsymbol{p}_i = 1$. $\boldsymbol{p}_i$ represents the probability that $\boldsymbol{x}$ belongs to the ith class. The label L of $\boldsymbol{x}$ is supposed to be the one that has largest probability: $L = argmax_i(softmax(H(\boldsymbol{x}))_i)$. Therefore, $\boldsymbol{p}$ can be regarded as the probability distribution of input $\boldsymbol{x}$ over k classes. Since softmax is a monotonically increasing normalized function, the largest value in $\boldsymbol{p}$ has the same index with the largest value in logits $\boldsymbol{y}$. We can regard logits also as probability distributions without normalization.

In this paper, we treat logit $\boldsymbol{y}$ and softmax $\boldsymbol{p}$ as the output of DNN since they both are k-dimensional vectors and convey the classification probability distribution.

2.2. Evasion attacks

Evasion attacks transform inputs into adversarial examples by adding small amount of noises so that it will be misclassified by DNN. Szegedy et al. [7] first noticed the existence of adversarial examples in DNN: given a benign input $\boldsymbol{x}$, whose right label is $l = C(\boldsymbol{x})$, a similar input $\boldsymbol{x}'$ such that $C(\boldsymbol{x}') = t, t \neq l$ can be easily found. Given the specific $\boldsymbol{x}$ and t, $\boldsymbol{x}'$ is called *targeted* adversarial example.

There exists another less powerful attack named *untargeted* attack. Instead of targeting at t, it only searches for an example such that $C(\boldsymbol{x}') \neq l$ and $\boldsymbol{x}$, $\boldsymbol{x}'$ are very close. Carlini et al. [11] proposed a more efficient but less accurate strategy to conduct untargeted attacks: choosing the closest one among adversarial examples generated by targeted attacks.

Theoretically, targeted evasion attacks can be transformed into the following optimization problem [7]:

$$\begin{array}{c} minimize:\ D(\boldsymbol{x}, \boldsymbol{x}') \\ subject\ to:\ C(\boldsymbol{x}') = t \\ \boldsymbol{x}' \in [0,1]^m \end{array} \tag{1}$$

where t is the targeted label and D is the distance metric defining the distance between a pair of inputs.

Distance metrics. There are 3 widely-used distance metrics:

1) L_0 distance reflects the number of pixels that have been changed in an image. It counts the number of i with $x_i \neq x'_i$ no matter how much these values differ.
2) L_2 distance is the standard Euclidean distance between $\boldsymbol{x}$ and $\boldsymbol{x}'$. It sums up root-mean-square of each pixel pair. Unlike L_0, L_2 distance can remain small even many pixels are changed.
3) L_∞ distance measures the maximum change among the changed pixels. Its value is $max\{|x_1 - x'_1|, \cdots, |x_m - x'_m|\}$ no matter how many pixels change.

We summarize some popular evasion attacks in Tab.1 according to the distance matrix they use:

TABLE 1. EXISTING EVASION ATTACKS UNDER 3 DISTANCE METRICS.

	L-BFGS	FGSM	IGSM	JSMA	Deepfool	CW
L_0				✓		✓
L_2	✓				✓	✓
L_∞		✓	✓			✓

L-BFGS [7] uses box-constrained L-BFGS to solve the problem in Eq.1 with an extra loss function. Deepfool [16] is an untargeted attack under L_2 distance metric. Fast Gradient Sign method (FGSM) [9] calculates the gradient of the loss function to determine the direction towards targeted adversarial label in every step. Similarly, Iterative gradient sign method (IGSM) [17] improved FGSM by taking an iteration of smaller step size and clipping the current result. Papernot et al. proposed the Jacobian-based Saliency Map Attack (JSMA) [18] under L_0 distance metric. JSMA is a greedy algorithm that makes use of saliency map to pick pixels to alter. They use both gradients of $H(\boldsymbol{x})_t$ and $softmax(H(\boldsymbol{x}))_t$ under different configurations [10].

CW attacks. Carlini et al. [11] proposed a set of attacks under the three distance metrics:

L_2: They first set the adversarial example $\boldsymbol{x}' = \frac{1}{2}(tanh(w) + 1)$. Since $-1 \leq tanh(w_i) \leq 1$, $\boldsymbol{x}'$ is automatically set in range $[0,1]^m$. Given input $\boldsymbol{x}$ and target label t, it minimizes $||\boldsymbol{x} - \boldsymbol{x}'||^2 + \ c \cdot f(\boldsymbol{x}')$, where $f(\boldsymbol{x}') = max(max\{H(\boldsymbol{x}')_i : i \neq t\} - H(\boldsymbol{x}')_t, -\kappa)$, and κ controls the confidence that $\boldsymbol{x}'$ is classified as label t.

L_0: They also adopted the idea of iterative algorithms. In each iteration, they make use of their L_2 attack and compute

the gradient of $f(\boldsymbol{x}')$ to determine the less important pixels for the classification t then cut them until obtaining a small subset of pixels that can be altered to produce an adversarial example classified as label t.

L_∞: They donate $\boldsymbol{x}' = \boldsymbol{x} + \delta$, where δ is noises to be minimized. Then transform Eq. 1 into: $c \cdot f(\boldsymbol{x} + \delta) + \sum_i max\{(\delta_i - \tau), 0\}$ and search for proper c and τ.

2.3. Defensive methods

Defensive distillation. Papernot et al. [10] proposed the defensive distillation to train another DNN with the soft labels. They first train standard DNN but replace the softmax by $\frac{1}{T}H(\boldsymbol{x})$, which is smoother than softmax so-called soft labels. To distill this DNN, they using the same training data and soft labels, train the second network which behaves like the first model. Actually, Carlini et al. [11] found that distillation cannot remove adversarial examples. The attacks still achieve 100% success rates in their datasets.

Feature squeezing. Xu et al. [19] proposed a detection method by comparing a DNN model's prediction on the original images with the squeezed image by reducing the color bit depth or spatial smoothing. Single detective method can not recover the right label of adversarial examples. We suppose that it requires human to recognize the adversarial examples thus the whole system would be inefficient.

MagNet. Meng and Chen [20] proposed a framework against adversarial examples by training separate detector networks and reformer networks. The detector learns the difference between benign and adversarial examples by approximating the manifold of benign examples. While reformer moves the adversarial examples towards the manifold of benign examples to correctly classify them.

Region-based classifier. Cao and Gong [15] proposed a method called Region-based Classification (RC), which ensembles information in a hypercube centered at the example. Specifically, RC regards input space as a hyper space and classifier divides the space into different regions. RC samples some points in the hypercube centered at testing example and re-uses DNN to predict the label for each sampled data point. Afterward, it takes a majority vote among all labels as the label for the testing example.

3. Our detection method

First, we characterize benign and adversarial examples by exploring the reason why a classifier makes different decisions on these similar images.

Characteristics of adversarial examples. Different from the previous detection approaches such as using the content of image itself [13] and looking the inner convolution layers [12] of the network, our detection method focuses on *logits* from last hidden layer. Actually, it is the maximum value in logit that determines the classification.

Fig.1 shows the great difference in logits between benign 7 referring the first row, and adversarial examples generated from 7 in the last nine rows. The first column contains their labels DNN model predicted. An attack succeeds when DNN model predicts wrong label rather than 7 even though human recognizes all the 10 images as 7 displayed in the second column. The maximum in each vector, which determines the classification, is marked in red.

7			-7.771	0.485	1.720	3.543	1.609	-3.061	-14.77	**29.17**	-5.72	5.019
0			**1.603**	-1.699	1.580	1.454	-1.907	-1.742	-1.334	1.601	1.501	-3.106
1			-5.737	**5.074**	5.070	2.716	-0.263	-4.035	-4.098	5.067	-2.278	-3.615
2			-2.756	-0.146	**11.67**	1.155	-2.381	-6.354	-7.830	11.66	-3.838	-2.927
3			-11.32	-4.095	-5.638	**17.47**	-1.079	4.020	-14.76	17.46	-2.408	11.05
4			-4.557	-0.793	-4.342	2.639	**5.521**	-0.165	-4.780	5.517	-0.472	5.473
5			-8.449	-3.305	-8.148	10.70	-0.951	**10.71**	-10.30	10.70	-1.403	7.981
6			0.835	-1.637	0.678	0.843	-2.107	-1.735	**0.861**	0.838	0.577	-1.759
8			-3.741	-4.118	-0.712	4.756	-1.902	-0.280	-6.591	4.731	**4.781**	4.777
9			-11.57	-6.348	-7.836	12.89	3.571	3.996	-15.15	17.70	-3.334	**17.71**

Figure 1. Label, image, noises, logit of benign and adversarial examples using CW-L_2 attacks with confidence parameter $\kappa = 0$.

The advantages of using logits are threefold: 1) logits is the direct output from the last hidden layer. There is no need to know the complex internal structure of the DNN itself; 2) the dataset of logits is rather small, which improves the efficiency of training detector and prediction; 3) logits provide detector with enough information to distinguish adversarial examples from benign examples. Logits convey the probability distribution on 10 classes. The index of the maximum value in logit directly determines the classification of this image and also reflects the confidence.

In Fig.1, for example, the adversarial 0 has a totally different probability distribution from original data labeled with 7. Although two images both look like 7. Furthermore, for the logit of benign 7, the index of maximum is 7 with confidence more than 20, assuming that the index of first value is 0. In contrast for the adversarial example labeled with 0, the index of maximum is 0 with confidence only 1, and index 7 with less confidence but very close to the the confidence of adversarial index.

Detector. Our detector is an extremely light-weight neural network containing 2 fully connected layers. For training and testing detector, we randomly choose benign examples and generated adversarial example for each of them. More details refers to Sec. 5.

4. Detector-Corrector Network

We propose Detector-Corrector network (DCN), consisting of two independent parts: 1) a detector to distinguish adversarial examples from benign examples, 2) a corrector to correct the wrong label of adversarial example. There is no modification of the original DNN even if it is vulnerable to evasion attack. Together with our Detector-Corrector network, DNN is able to maintain its accuracy of benign examples and mitigate adversarial examples with a little overhead.

Corrector. Our corrector makes use of RC [15] to recover the right label of an example classified as adversarial by detector. Comparing with the DNN model that makes a prediction based on only one input, corrector makes a prediction based on the area around the input data in order to avoid bias. Simply speaking, corrector makes m times prediction $L_1, L_2, ..., L_m$ about points $P_1, P_2, ..., P_m$ located within a hypercube $HC(r, x)$ of radius r centered at input data x. Among the m labels, corrector chooses the mode that appears most repeatedly as the label. Based on the DNN classifier $L = C(x)$, correct calculates: $Label = Mode(L_1, L_2, ..., L_m), \{L_i = C(x_i) | x_i \in HC(r, x), i = 1, 2, ..., m\}$.

In Fig.3, once a successful adversarial example was generated, it must have been moved to another region but as close as possible to the original region so as to add smaller distortion. A proper hypercube centered at adversarial example could intersect the most with the region of original label. By uniformly sampling points inside the hypercube with proper radius, corrector approximates the region intersecting the most with the hypercube to recover the original label.

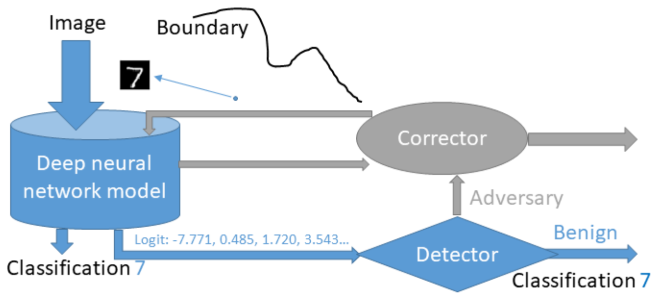

Figure 2. Workflow of processing benign 7.

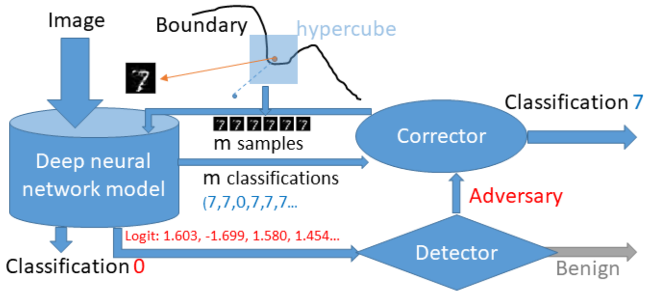

Figure 3. Workflow of processing adversarial 0 with original label 7.

DCN workflow. Fig.2 shows the workflow of DCN if detector reports a benign input. First of all, DNN model predicts label 7 and outputs logit as usual. Then the well-trained detector makes a benign judgment according to this logit. Immediately we get the right label only through two predictions without corrector participation. Since detector itself is very light-weighted, almost no overheads are added.

Workflow of processing adversarial example is more complicated due to the activation of corrector. In Fig.3, human recognize the image as 7 while DNN model produces the wrong label 0. Based on the unusual probability distribution in logit, detector immediately notices this adversarial example and passes it to corrector. Then corrector randomly samples a bunch of images located within the hypercube and obtains their predicted labels from DNN model. Finally corrector decides on the mode 7 as the classification.

All in all, for both benign and adversarial examples, our detector-corrector network predicts the right label.

5. Evaluation

In this section, we evaluate the accuracy and efficiency on two datasets. Moreover, we compared our approach with previous defensive methods against six types of state-of-art evasion attacks. We used open-source CW attacks from authors and implemented our DCN and RC by ourselves in Python 3.6.2 using Keras [21] and Tensorflow [22]. All the running times below are in seconds and measured on a Windows computer with a 3.4GHz Intel Core i5 processor and 8GB RAM without GPU acceleration.

5.1. Experimental setup

Datasets. We use: MNIST [23] and CIFAR-10 [24] as benchmark. MNIST is a dataset of handwritten digits formatted as 28×28 gray images. CIAFR-10 collected 10 kinds of colored images formatted as $32 \times 32 \times 3$. For our convenience, data was normalized into real number between -0.5 and 0.5. Carlini and Wagner [11] also used the same normalization to conduct their attacks.

Evasion attacks. We use a set of targeted CW attacks [11] since they include three distance metrics. In addition, we transform the targeted attacks into corresponding untargeted attacks by means of the strategy proposed in Sec 2.

Other defensive methods. We compare our DCN with the following classifier and defensive methods:

Standard DNN. We train DNN model according to the network architectures and parameters presented in [11]. Our models achieved 99.3% accuracy on MNIST and 78.7% on CIFAR-10, both comparable to the accuracy in [11]. Standard DNN is used as a baseline.

Distillation. We use distillation [10] to re-train standard DNN by setting temperature $T = 100$, which was found to be most effective in [10]. Afterwards, when using distilled network to do classification, we use temperature $T = 1$.

Region-based classifier. According to [15], we transform standard DNN classifier into region-based DNN classifier with set $r = 0.3$ for MNIST,$r = 0.02$ for CIFAR-10 and $m = 1,000$, which are reported most effective [15].

5.2. DCN setup

Detector. For training detector, we randomly select 1,000 benign examples from MNIST that standard DNN correctly classifies and generate adversarial examples for them. Noted that we only use adversarial examples generated by CW-L_2 attack to train our detector but test it with other CW attacks. Specifically, for each benign example, we generate 9 adversarial examples since MNIST and CIFAR-10 both are 10-class problems. Finally, we obtain datasets composed $10,000$ examples in total. For CIFAR-10 we reduce the dataset size to half, which contains 5,000 examples in all.

For testing, we randomly select 1,000 benign data except those using for training and generate 9, 000 adversarial examples since CW-attacks do have 100% successful rate on standard DNN. We report the accuracy in Tab.2.

TABLE 2. FALSE RATE OF DETECTOR

	False negative	False positive
MNIST	3.7%	0.31%
CIFAR-10	21.5%	0.91%

False negatives imply that benign examples are classified to be adversarial and activate corrector. Although activations of corrector bring extra workload. False negative is *acceptable* because corrector can recover not only adversarial examples but also benign examples with minimum overheads.

False positives mean that adversarial examples are classified as benign examples then fail to activate corrector thus we get wrong label. False positive should be as *low* as possible so that all adversarial examples could activate corrector to recover right labels.

Corrector. In Sec.4, we mention that there are two parameters to be set: hypercube radius r and quantity of samples m. We adopt the parameter r [15] but set a smaller $m = 50$, since in Fig.4 we find m has little effect on accuracy but is roughly proportional to the running time. Setting smaller m could improve efficiency without compromising accuracy.

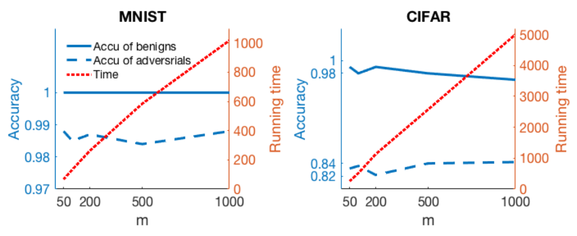

Figure 4. Accuracies and running time of corrector for different m.

5.3. Comparisons

We analyze robustness and efficiency of DCN compared with distillation and RC, against the state-of-the-art targeted and untargeted evasion attacks under L_0, L_2 and L_∞ norms.

Performance with benign examples. We randomly sample 1,000 benign examples from MNIST and 500 from CIFAR-10. Since RC is too inefficient to run enormous examples. Tab.3 shows the accuracies and overall running times. First, DCN achieves the same accuracy with the baseline. As described in Sec.4, detector acts as a filter that allows benign examples to pass through it with label DNN predicts. Even a benign example activates the corrector, we still get the right label. In contrast, distillation achieves lower accuracy than DCN and RC performs unstably. Its accuracy for MNIST is the lowest while for CIFAR is highest. As for efficiency, DCN consumes only 3 seconds and 55 seconds respectively on MNIST and CIFAR comparable with baseline and distillation, while RC takes thousands of seconds. This is because, for every input, RC wastes massive resource and time for meaningless sampling and predictions.

Although the differences between accuracies are smaller, DCN does not sacrifice accuracy on benign examples. While others sacrifice accuracy or efficiency.

TABLE 3. CLASSIFICATION ACCURACY ON BENIGN EXAMPLES

	Standard	Distillation	RC	Our DCN
MNIST	99.4%	99.3%	99.1%	99.4%
Overall time	0.783	0.735	1017.07	3.356
CIFAR-10	78.2%	77.0%	78.6%	78.4%
Overall time	1.507	1.483	5002.75	55.317

Robustness to adversarial examples. Tab.4 and 5 show the successful rates of six types of evasion attacks. For standard DNN and distillation network, we report that the attack succeeds if the attack produces an adversarial example that is misclassified. For RC and our DCN, we report an attack failure if the right label of adversarial example is recovered. Lower successful rate indicates that the defensive method is more effective and robust.

TABLE 4. SUCCESSFUL RATE OF EVASION ATTACKS ON MNIST

	Targeted			Untargeted		
	L_0	L_2	L_∞	L_0	L_2	L_∞
DNN	100%	100%	100%	100%	100%	100%
Distillation	100%	100%	100%	100%	100%	100%
RC	57.11%	9.22%	9.67%	49%	8%	9%
Our DCN	56.11%	1.89%	0.89%	44%	0%	0%

TABLE 5. SUCCESSFUL RATE OF EVASION ATTACKS ON CIFAR-10

	Targeted			Untargeted		
	L_0	L_2	L_∞	L_0	L_2	L_∞
DNN	100%	100%	100%	100%	100%	100%
Distillation	100%	100%	100%	100%	100%	100%
RC	33.89%	5.33%	18.67%	63%	5%	34%
Our DCN	35.22%	5.33%	18.22%	36%	5%	32%

Since CW attacks are inefficient, we randomly sample 100 benign examples that standard DNN correctly classifies and generate adversarial examples for each of them using different attacks. Specifically, for each benign examples, we generate 9 adversarial examples using targeted evasion attack, and choose only one adversarial example with lowest distortion as untargeted adversarial examples.

Our evaluation indicates the following: first, successful rate is the lowest with DCN than other defensive methods. This is because DCN is able to detect adversarial examples accurately and recover the right labels of them. Second, DNN performs better against L_2 and L_∞ attack than L_0, which also happens on RC. Actually, detector detects almost all adversarial examples but corrector fails to recover some of them. The reason is that adversarial examples generated under L_0 metrics may be further away from the boundary thus the hypercube intersects less with the space of right label. But L_0 distance only counts how many pixels that have been changed no matter how much each pixel changes. As a result, L_0 attacks often add *spots* on images, which human may perceive easily. Third, our DCN reduces the successful rate of all sorts of evasion attacks. Comparing with RC, our DCN is more effective since we improve the parameters of corrector, which contributes not only efficacy but also efficiency.

Efficiency. Tab.6 lists the running time of our DCN for 100 random examples containing different percentages of adversarial examples and the running time of RC for the same 100 examples. Since distillation is no longer effective to reduce successful rate of evasion attacks. We only compare our DCN with less effective RC.

TABLE 6. THE RUNNING TIME FOR 100 DATA WITH DIFFERENT ADVERSARIAL EXAMPLES PERCENTAGE.

	MNIST		CIFAR-10	
	Our DCN	RC	Our DCN	RC
0%	0.46	143.49	5.13	557.12
1%	0.47	148.60	5.23	555.78
10%	1.08	147.42	7.17	574.45
20%	1.81	148.89	9.13	565.56
50%	3.97	147.44	15.15	557.54
75%	5.64	146.25	20.13	571.12
100%	7.51	146.11	25.79	545.57

Our result shows that: 1) the running time of our DCN increases with the percentage of adversarial examples since adversarial examples activate the corrector, which consumes more time than detection. 2) our DCN is significantly more efficient than RC, which samples and predicts 1,000 times for every input even when there is no adversarial example at all. If an attacker just mix several adversarial examples into dataset, for the overall robustness, RC has to sample and predict repeatedly for each input. Therefore, RC spends almost the same amount of time in each test. We average the running time of RC and draw Fig. 5 with *logarithmic* axis for the running time to highlight the efficiency difference between our DCN and RC.

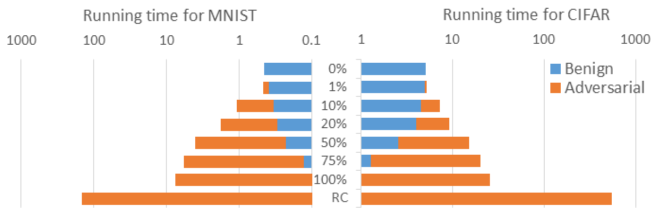

Figure 5. The running time for 100 data with different adversarial examples percentage using our DCN and RC.

6. Discussions

Other correctors. Our work demonstrates the high accuracy of detector. An accurate corrector is of great importance to improve the robustness of whole network if it could accurately recover more right labels, especially for L_0 adversarial examples.

Adaptive CW attack against our DCN. Another interesting work is to explore new adaptive evasion attack to fool our DCN. Increasing the attack confidence parameter κ will generate more confident adversarial examples with the expense of adding more noises. Those examples are more likely to be noticed by human. On the other hand, adaptive method can attempt to construct new loss function to bypass the detection network, as CW attacks only accepts one network.

DCN against other evasion attacks. Our preliminary experiment demonstrates that DNC is a promising method to mitigate adversarial examples generating by CW attacks. We start to evaluate it with other evasion attacks such as FGSM, JSMA and DeepFool implemented by CleverHans [25].

7. Conclusions

In this paper, we propose a new detection method and detector-corrector network to mitigate adversarial examples efficiently. First of all, we characterize adversarial examples according to the differences in logits. Second, based on the characteristics, we train detector to distinguish adversarial examples, which achieves almost 100% accuracy on adversarial examples with low false negative. Third, we propose DCN based on DNN that effectively reduce successful rate of the existing state-of-the-art evasion attacks under different distance metrics. Our evaluation demonstrates the efficacy and efficiency of DCN agianst various evasion attacks.

Future work includes new correcting method and adaptive evasion attacks. We encourage researchers who develop new evasion attacks to evaluate their attacks against DCN. To enable other researchers to test with our work in an easy manner, all of our implementation are available at: https://github.com/joy8023/DCN

Acknowledgement

We would like to thank the anonymous reviewers for their valuable feedback and comments. This project is partially supported by the NSFC/RGC Joint Research Scheme (N_HKU729/13) and the RGC CRF grant (CityU C1008-16G).

References

[1] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, “Mastering the game of go with deep neural networks and tree search,” *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[4] D. Andor, C. Alberti, D. Weiss, A. Severyn, A. Presta, K. Ganchev, S. Petrov, and M. Collins, "Globally normalized transition-based neural networks," *arXiv preprint arXiv:1603.06042*, 2016.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[6] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Computer vision and pattern recognition (CVPR), 2012 IEEE conference on*. IEEE, 2012, pp. 3642–3649.

[7] 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.

[8] I. Stoica, D. Song, R. A. Popa, D. Patterson, M. W. Mahoney, R. Katz, A. D. Joseph, M. Jordan, J. M. Hellerstein, J. E. Gonzalez *et al.*, "A berkeley view of systems challenges for ai," *arXiv preprint arXiv:1712.05855*, 2017.

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

[10] 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.

[11] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 2017, pp. 39–57.

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

[13] D. Hendrycks and K. Gimpel, "Early methods for detecting adversarial images," 2017.

[14] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM, 2017, pp. 3–14.

[15] X. Cao and N. Z. Gong, "Mitigating evasion attacks to deep neural networks via region-based classification," in *Proceedings of the 33rd Annual Computer Security Applications Conference*. ACM, 2017, pp. 278–287.

[16] S. M. Moosavi Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, no. EPFL-CONF-218057, 2016.

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

[18] 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.

[19] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," *arXiv preprint arXiv:1704.01155*, 2017.

[20] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 135–147.

[21] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[22] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

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

[24] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[25] N. Papernot, N. Carlini, I. Goodfellow, R. Feinman, F. Faghri, A. Matyasko, K. Hambardzumyan, Y.-L. Juang, A. Kurakin, R. Sheatsley *et al.*, "cleverhans v2. 0.0: an adversarial machine learning library," *arXiv preprint arXiv:1610.00768*, 2016.