Towards Robust Detection of Adversarial Examples

Tianyu Pang [1] Chao Du [1] Yinpeng Dong [1] Jun Zhu [1]

Abstract

Although the recent progress is substantial, deep learning methods can be vulnerable to the maliciously generated adversarial examples. In this paper, we present a novel training procedure and a thresholding test strategy, towards robust detection of adversarial examples. In training, we propose to minimize the reverse cross-entropy (RCE), which encourages a deep network to learn latent representations that better distinguish adversarial examples from normal ones. In testing, we propose to use a thresholding strategy as the detector to filter out adversarial examples for reliable predictions. Our method is simple to implement using standard algorithms, with little extra training cost compared to the common cross-entropy minimization. We apply our method to defend various attacking methods on the widely used MNIST and CIFAR-10 datasets, and achieve significant improvements on robust predictions under all the threat models in the adversarial setting.

1. Introduction

Deep learning (DL) has obtained unprecedented progress in various tasks, including image classification, speech recognition, and natural language processing (Goodfellow et al., 2016). However, a high-accuracy DL model can be vulnerable in the adversarial setting (Szegedy et al., 2014; Goodfellow et al., 2015), where adversarial examples are maliciously generated to mislead the model to output wrong predictions. Several attacking methods have been developed to craft such adversarial examples (Goodfellow et al., 2015; Kurakin et al., 2017a; Liu et al., 2017; Papernot et al., 2016a;b; Carlini & Wagner, 2017a; Dong et al., 2017). As DL is becoming ever more prevalent, it is imperative to improve the robustness, especially in safety-critical applications, e.g., self-driving cars, healthcare and finance.

Therefore, various defenses have been proposed attempting to correctly classify adversarial examples (Szegedy et al., 2014; Gu & Rigazio, 2014; Papernot et al., 2016c; Rozsa et al., 2016; Zheng et al., 2016). However, most of these defenses are not effective enough, which can be successfully attacked by more powerful adversaries (Carlini & Wagner, 2017a;b). Overall, as adversarial examples always exist for a fixed parametric model (thus fixed decision boundary), it is unlikely for such methods to solve the problem by preventing adversaries from generating adversarial examples, no matter how hard it is to find them.

Due to the difficulty, recent work on defense has turned to detecting adversarial examples instead. Grosse et al. (2017) introduce an extra class in classifiers solely for adversarial examples, and similarly Gong et al. (2017) train an additional binary classifier to decide whether an instance is adversarial or not. Metzen et al. (2017) detect adversarial examples via training a detection neural network, which takes input from intermediate layers of the classification network. Bhagoji et al. (2017) reduce dimensionality of the input image fed to the classification network, and train a fully-connected neural network on the smaller input. Li & Li (2016) build a cascade classifier where each classifier is implemented as a linear SVM acting on the PCA of inner convolutional layers of the classification network. However, these methods all require a large amount of extra computational cost, and some of them also result in loss of accuracy on normal examples. Feinman et al. (2017) propose a kernel density estimate method to detect the points lying far from the data manifolds in the final-layer hidden space, which does not change the structure of the classification network with little extra computational cost. They also combine with a Bayesian uncertainty estimate method which is only available on dropout neural networks (Li & Gal, 2017). However, Carlini & Wagner (2017b) show that each of these defense methods can be evaded by an adversary targeting at that specific defense, i.e., by a white-box adversary.

In this paper, we make contributions by presenting a new defense method that provides a robust way to detect adversarial examples. Specifically, we demonstrate that the white-box adversaries have to craft adversarial examples with macroscopic noises to successfully evade our defense, which means the crafted adversarial examples can be easily filtered out by a human observer. Our method consists of a novel training procedure and a thresholding test strategy.

[1]Dept. of Comp. Sci. & Tech., TNList Lab, State Key Lab for Intell. Tech. & Systems, CBICR Center, Tsinghua University. Correspondence to: Jun Zhu <dcszj@mail.tsinghua.edu.cn>.

In training, we propose to minimize a novel objective function, named as reverse cross-entropy (RCE), instead of minimizing the common cross-entropy (CE) loss (Goodfellow et al., 2016). By minimizing RCE, our training procedure encourages the classifiers to return a high confidence on the true class while a uniform distribution on false classes for each data point, and further makes the classifiers map the normal examples to the neighborhood of low-dimensional manifolds in the final-layer hidden space. The minimization of RCE is simple to implement using stochastic gradient descent methods, with little extra training cost, as compared to CE. Therefore, it can be easily applied to any deep networks.

In testing, we propose a thresholding strategy as the detector based on the kernel density (Feinman et al., 2017). By setting a proper threshold, the detector can filter out adversarial examples for robust predictions, namely, it makes the classifiers return meaningful predictions only when values of kernel density are higher than a given threshold, and otherwise refuse to predict.

We apply our method to defend various attacking methods on the widely used MNIST (LeCun et al., 1998) and CIFAR-10 (Krizhevsky & Hinton, 2009) datasets. We test the performance of our method under different threat models, i.e., *oblivious adversaries*, *white-box adversaries* and *black-box adversaries*. We choose the kernel density estimate method as our baseline, which has shown its superiority and versatility compared to other detection-based defense methods (Carlini & Wagner, 2017b). The results demonstrate that compared to the baseline, the proposed method improves the robustness against adversarial attacks under all the threat models, while maintaining state-of-the-art accuracy on normal examples.

2. Preliminaries

This section provides the notations and introduces the threat models and attacking methods.

2.1. Notations

A deep neural network (DNN) classifier can be generally expressed as a mapping function $F(X, \theta) : \mathbb{R}^d \to \mathbb{R}^L$, where $X \in \mathbb{R}^d$ is the input variable, θ denotes all the parameters and L is the number of classes (hereafter we will omit θ without ambiguity). Here, we focus on the DNNs with softmax output layers. For notation clarity, we define the softmax function $\mathbb{S}(z) : \mathbb{R}^L \to \mathbb{R}^L$ as $\mathbb{S}(z)_i = \exp(z_i)/\sum_{i=1}^{L}\exp(z_i), i \in [L]$, where $[L] := \{1, \cdots, L\}$. Let Z be the output vector of the penultimate layer, i.e., the final hidden layer. This defines a mapping function: $X \mapsto Z$ to extract data representations. Then, the classifier can be expressed as $F(X) = \mathbb{S}(W_s Z + b_s)$, where W_s and b_s are the weight matrix and bias vector of the softmax layer respectively. We denote the pre-softmax output $W_s Z + b_s$ as Z_{pre}, termed logits. Given an input x (i.e., an instance of X), the predicted label for x is denoted as $\hat{y} = \arg\max_{i\in[L]} F(x)_i$. The probability value $F(x)_{\hat{y}}$ is often used as the confidence score on this prediction (Goodfellow et al., 2016).

Let $\mathcal{D} := \{(x_i, y_i)\}_{i\in[N]}$ be a training set with N input-label pairs, where $y_i \in [L]$ is the true class of x_i. One common training objective is to minimize the cross-entropy (CE) loss, which is defined as:

$$\mathcal{L}_{CE}(x, y) = -1_y^\top \log F(x) = -\log F(x)_y,$$

for a single pair (x, y). Here, 1_y is the one-hot encoding of y and the logarithm of a vector is defined as taking logarithm of each element. The CE training procedure intends to minimize the average CE loss (under proper regularization) to obtain the optimal parameters $\theta^* = \arg\min_\theta \frac{1}{N}\sum_{i\in[N]}\mathcal{L}_{CE}(x_i, y_i)$, which can be efficiently done by stochastic gradient methods with back-propagation (Rumelhart et al., 1988).

2.2. Threat Models

In the adversarial setting, an elaborate taxonomy of threat models is introduced in Carlini & Wagner (2017b):

- **Oblivious adversaries** are not aware of the existence of the detector D and generate adversarial examples based on the unsecured classification model F.
- **White-box adversaries** know the scheme and parameters of D, and can design special methods to attack both the model F and the detector D simultaneously.
- **Black-box adversaries** know the existence of the detector D with its scheme, but have no access to the parameters of the detector D or the model F.

2.3. Attacking Methods

Although DNNs have obtained substantial progress, adversarial examples can be easily identified to fool the network, even when its accuracy is high (Nguyen et al., 2015). Several attacking methods on generating adversarial examples have been introduced in recent years. Most of them can craft adversarial examples that are visually indistinguishable from the corresponding normal ones, and yet are misclassified by the target model F. Here we introduce some well-known and commonly used attacking methods.

Fast Gradient Sign Method (FGSM): Goodfellow et al. (2015) introduce an one-step attacking method, which crafts an adversarial example x^* as $x^* = x + \epsilon \cdot \text{sign}(\nabla_x \mathcal{L}(x, y))$, with the perturbation ϵ and the training loss $\mathcal{L}(x, y)$.

Basic Iterative Method (BIM): Kurakin et al. (2017a) propose an iterative version of FGSM, with the formula as $x_i^* = \text{clip}_{x,\epsilon}(x_{i-1}^* + \frac{\epsilon}{r} \cdot \text{sign}(\nabla_{x_{i-1}^*}\mathcal{L}(x_{i-1}^*, y)))$, where $x_0^* = x$, r is the number of iteration steps and $\text{clip}_{x,\epsilon}(\cdot)$ is a clipping function to keep x_i^* in its domain.

Iterative Least-likely Class Method (ILCM): Kurakin et al. (2017a) also propose a targeted version of BIM as $x_i^* = \text{clip}_{x,\epsilon}(x_{i-1}^* - \frac{\epsilon}{r} \cdot \text{sign}(\nabla_{x_{i-1}^*}\mathcal{L}(x_{i-1}^*, y_{ll})))$, where $x_0^* = x$ and $y_{ll} = \arg\min_i F(x)_i$. ILCM can avoid label leaking (Kurakin et al., 2017b), since it does not use the true label y when crafting adversarial examples.

Jacobian-based Saliency Map Attack (JSMA): Papernot et al. (2016b) propose another iterative method for targeted attack, which perturbs one feature x_i by a constant offset ϵ in each iteration step that maximizes the saliency map

$$S(x,t)[i] = \begin{cases} 0, \text{ if } \frac{\partial F(x)_y}{\partial x_i} < 0 \text{ or } \sum_{j \neq y} \frac{\partial F(x)_j}{\partial x_i} > 0, \\ (\frac{\partial F(x)_y}{\partial x_i}) \left| \sum_{j \neq y} \frac{\partial F(x)_j}{\partial x_i} \right|, \text{ otherwise.} \end{cases}$$

Compared to other methods, JSMA perturbs fewer pixels.

Carlini & Wagner (C&W): Carlini & Wagner (2017a) introduce an optimization-based method, which is one of the most powerful attacks. They define $x^* = \frac{1}{2}(\tanh(\omega) + 1)$ in terms of an auxiliary variable ω, and solve the problem $\min_\omega \|\frac{1}{2}(\tanh(\omega) + 1) - x\|_2^2 + c \cdot f(\frac{1}{2}(\tanh(\omega) + 1))$, where c is a constant that need to be chosen by modified binary search. $f(\cdot)$ is an objective function as $f(x) = \max(\max\{Z_{pre}(x)_i : i \neq y\} - Z_{pre}(x)_i, -\kappa)$, where κ controls the confidence on adversarial examples.

Note that both JSMA and C&W do not rely on the training loss $\mathcal{L}(x, y)$, which indicates that these attacks are independent of the training procedures of the target networks.

3. Methodology

In this section, we present a new method to improve the robustness of classifiers against adversarial examples. We first provide some insights on the existence of adversarial examples and the insufficiency of the confidence score as a thresholding metric, which guide us to the new method.

3.1. The Intrinsic Existence of Adversarial Examples

Previous work (Goodfellow et al., 2015; Papernot et al., 2016b) hypothesizes that the existence of adversarial examples is caused by certain defects in training. One main support is from the universal approximator theorem (Hornik et al., 1989) that DNNs are able to represent functions resisting adversarial examples. However, in practice any given DNN has an architecture of limited scale, thereby a limited representation capability, and the existence of adversarial examples is unavoidable. Namely, for a given DNN classifier, its decision boundary is fixed. Then, any pair of similar instances located on different sides of the decision boundary will be classified into different classes. Very often, such a pair of inputs are not distinguishable by human observers; thus, it sounds counterintuitive and irrational to have a jump on the predicted labels. Previous works that attempt to classify adversarial examples correctly (Szegedy et al., 2014; Gu & Rigazio, 2014; Papernot et al., 2016c) only result in the change of the distribution of decision boundary but the jump on the predicted labels nearby the decision boundary still exists. Therefore, a smart enough adversary can always find new adversarial examples to successfully attack the target classifier no matter how the decision boundary changes, as demonstrated by Szegedy et al. (2014).

3.2. The Insufficiency of Confidence and a New Metric

Given the intrinsic existence of adversarial examples, we design a defense method to detect them instead, which can help the classifiers distinguish adversarial examples from normal ones so as to filter them out for robust predictions.

A detection method relies on some metrics to decide whether an input x is adversarial or not for a given classifier $F(X)$. A potential candidate is the confidence $F(x)_{\hat{y}}$ on the predicted label $\hat{y}$, which inherently conveys the degree of certainty on a prediction and is widely used (Goodfellow et al., 2016). As F predicts incorrectly on adversarial examples, intuitively we should expect a higher confidence on a normal example than that on an adversarial one, which consequently allows us to distinguish them by the confidence values. However, a well-trained DNN classifier usually not only misclassifies adversarial examples but also gives high confidence on its predictions (Goodfellow et al., 2015; Nguyen et al., 2015), which renders the confidence unreliable in the adversarial setting. This is not surprising because in the normal setting without adversaries, the distribution of the test data is similar with that of the training data, then a high confidence implies a high probability to be a correct prediction. In the adversarial setting, adversaries can explore the points far from the data distribution (Li & Gal, 2017), and the predictions in these regions mostly result from extrapolation with high uncertainty. Therefore, a high confidence returned in these regions is unreliable and probably a false positive.

Instead of using the unreliable confidence as the sole detection metric in the adversarial setting, we construct another metric which is more pertinent and helpful to our goal. Namely, we define the metric of *non-ME*—the entropy of normalized non-maximal elements in $F(x)$, as:

$$\text{non-ME}(x) = -\sum_{i \neq \hat{y}} \hat{F}(x)_i \log(\hat{F}(x)_i), \quad (1)$$

where $\hat{F}(x)_i = {F(x)_i}/{\sum_{j \neq \hat{y}} F(x)_j}$ are the normalized non-maximal elements in $F(x)$. Hereafter we will consider the final hidden vector z of F given x, and use the notation $F(z)$ with the same meaning as $F(x)$ without ambiguity.

To intuitively illustrate the ideas, Fig. 1a presents an example of classifier F in the hidden space, where $z \in \mathbb{R}^2$ and $L = 3$. Let $Z_{pre,i}, i \in [L]$ be the i-th element of the logits Z_{pre}. Then the decision boundary between each pair of classes i and j is the hyperplane $db_{ij} := \{z : Z_{pre,i} =$

$Z_{pre,j}\}$. In Fig. 1a, each db_{ij} corresponds to one of the three black lines. Let $DB_{ij} = \{Z_{pre,i} = Z_{pre,j} + C, C \in \mathbb{R}\}$ be the set of all parallel hyperplanes w.r.t. db_{ij}, which are parallel lines in a 2d space as in Fig. 1a. Furthermore, we denote the half space $Z_{pre,i} \geq Z_{pre,j}$ as db_{ij}^{+}. Then, we can formally represent the decision region of class $\hat{y}$ as $dd_{\hat{y}} = \bigcap_{i\neq\hat{y}} db_{\hat{y}i}^{+}$ and the corresponding decision boundary of this region as $\overline{dd_{\hat{y}}}$. Note that the output $F(z)$ has $L-1$ equal non-maximal elements for any points on the low-dimensional manifold $S_{\hat{y}} = (\bigcap_{i,j\neq\hat{y}} db_{ij}) \bigcap dd_{\hat{y}}$. For example, when $L = 3$ and $\hat{y} = 1$, S_1 is $db_{23} \bigcap dd_1$, which is one of the three black dashed lines in Fig. 1a.

With the above notations, we have Lemma 1 as below:

Lemma 1. *(Proof in Appendix A) In the decision region $dd_{\hat{y}}$ of class $\hat{y}$, $\forall i, j \neq \hat{y}, \widetilde{db_{ij}} \in DB_{ij}$, the value of* non-ME *for any point on the low-dimensional manifold $\bigcap_{i,j\neq\hat{y}} \widetilde{db_{ij}}$ is constant. In particular,* non-ME *obtains its global maximal value* $\log(L-1)$ *on and only on $S_{\hat{y}}$.*

When $S_{\hat{y}}$ is empty, the conclusion of Lemma 1 becomes trial. Thus we consider the non-trivial cases where $S_{\hat{y}} \neq \emptyset$. Lemma 1 tells us that in the decision region of class $\hat{y}$ if one moves a normal input along the low-dimensional manifold $\bigcap_{i,j\neq\hat{y}} \widetilde{db_{ij}}$, then its value of non-ME will not change, and vice verse. This conclusion leads to Theorem 1:

Theorem 1. *(Proof in Appendix A) In the decision region $dd_{\hat{y}}$ of class $\hat{y}$, $\forall i, j \neq \hat{y}, z_0 \in dd_{\hat{y}}$, there exists a unique $\widetilde{db_{ij}^{0}} \in DB_{ij}$, such that $z_0 \in Q_0$, where $Q_0 = \bigcap_{i,j\neq\hat{y}} \widetilde{db_{ij}^{0}}$. Let $Q_0^{\hat{y}} = Q_0 \bigcap \overline{dd_{\hat{y}}}$, then the solution set of the problem*

$$\arg\min_{z_0}(\max_{z\in Q_0^{\hat{y}}} F(z)_{\hat{y}})$$

is $S_{\hat{y}}$. Furthermore, $\forall z_0 \in S_{\hat{y}}$ there is $Q_0 = S_{\hat{y}}$, and $\forall z \in S_{\hat{y}} \bigcap \overline{dd_{\hat{y}}}$, $F(z)_{\hat{y}} = \frac{1}{L}$.

Let z_0 be the representation of a normal example with the predicted class $\hat{y}$. When crafting adversarial examples based on z_0, adversaries need to perturb z_0 across the decision boundary $\overline{dd_{\hat{y}}}$. Theorem 1 says that there exists a unique low-dimensional manifold Q_0 that z_0 lies on in the decision region of class $\hat{y}$. If we can somehow restrict adversaries changing the values of non-ME when they perturb z_0, then by Lemma 1, the adversaries can only perturb z_0 along the manifold Q_0. In this case, the nearest adversarial counterpart z^* for z_0 must be in the set $Q_0^{\hat{y}}$ (Moosavi-Dezfooli et al., 2016). Then the value of $\max_{z\in Q_0^{\hat{y}}} F(z)_{\hat{y}}$ is an upper bound of the prediction confidence $F(z^*)_{\hat{y}}$. This bound is a function of z_0. Theorem 1 further tells us that if $z_0 \in S_{\hat{y}}$, the corresponding value of the upper bound will obtain its minimum $\frac{1}{L}$, which leads to $F(z^*)_{\hat{y}} = \frac{1}{L}$. In a nutshell, under the restriction that the values of non-ME are invariant when crafting adversarial examples, the nearest adversarial counterpart for any normal example on $S_{\hat{y}}$ will have the lowest value $\frac{1}{L}$ of confidence. This makes the nearest adversarial counterpart be easily distinguished according to its confidence score, and adversaries have to perturb the normal example further to obtain a higher value of confidence.

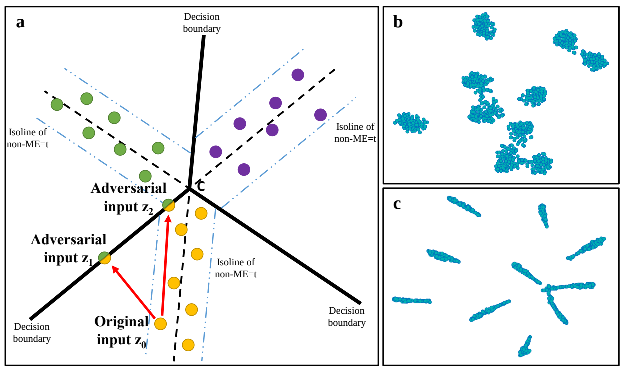


Figure 1. **a**, The three black solid lines are the decision boundary of the classifier, and each black line (both solid and dashed parts) is the decision boundary between two classes. The blue dot-dashed lines are the isolines of $\text{non-ME} = t$. **b**, t-SNE visualization of the final hidden vectors on CIFAR-10. The model is Resnet-32. The training procedure is CE. **c**, The training procedure is RCE.

In practice, the restriction can be implemented by a detector with the new metric of non-ME. Specifically, if the learned representation transformation: $X \mapsto Z$ in a DNN classifier can map the normal examples to the neighborhood of $S_{\hat{y}}$, then constructing the detector will filter out the examples that violate the restriction, where those examples are usually adversarial. In the case of Fig. 1a, all the points that locate on the set $S_{\hat{y}}$ (black dashed lines) have highest values of $\text{non-ME} = \log 2$. Assuming that the classifier F makes all the normal examples be mapped to the neighborhood of $S_{\hat{y}}$ by the representation transformation, which indicates that for any normal example z_0 the value of $\text{non-ME}(z_0)$ is higher than a threshold t. For implementation feasibility, we relax the restriction $z^* \in Q_0^{\hat{y}}$ to $\text{non-ME}(z^*) \geq t$. Therefore, given any unidentified input (e.g., a crafted adversarial example), if the input violates the restriction, i.e., its non-ME value is less than t, then the detector will immediately know that this input is not normal and filters it out.

As depicted in Fig. 1a, z_0 is the original normal input. z_0 locates in the neighborhood of $S_{\hat{y}}$, where the neighborhood boundary consists of the isolines of $\text{non-ME} = t$ shown by the blue dot-dashed lines. When there is no detector, the nearest successful adversarial example is z_1 locating on the nearest decision boundary w.r.t. z_0. In contrast, when non-ME is used as the detection metric, z_1 will be easily filtered out by the detector because $\text{non-ME}(z_1)$ is less than t, and the nearest successful adversarial example becomes z_2 in this case, which locates on the nearest junction manifold of the neighborhood boundary and the decision boundary. Note that the central point C of the decision boundary in

Fig. 1a is actually the singleton set $S_{\hat{y}} \bigcap \overline{dd_{\hat{y}}}$ in this case, on which $F(z)$ has the lowest confidence $\frac{1}{L} = \frac{1}{3}$ as indicated by Theorem 1. It is easy to find that the minimal perturbation $\|z_0 - z_2\|$ is larger than $\|z_0 - z_1\|$, almost everywhere. This means that due to the existence of the detector, adversaries have to move the original example further to successfully generate a nearest adversarial example that can fool the detector. Furthermore, according to Theorem 1, the confidence at z_2 is also lower than it at z_1. Thus although z_2 can fool the non-ME detector, it will still be most likely distinguished since its low confidence score.

3.3. The Reverse Cross-Entropy Training Procedure

Based on the above analysis, we now design a new training objective to improve the robustness of DNN classifiers. The key is to enforce a DNN classifier to map all the normal instances to the neighborhood of the low-dimensional manifolds $S_{\hat{y}}$ in the final-layer hidden space. According to Lemma 1, this can be achieved by making the non-maximal elements of $F(x)$ be as equal as possible, thus having a high non-ME value for every normal input. Specifically, for a training data (x, y), we let R_y denote its reverse label vector whose y-th element is zero and other elements equal to $\frac{1}{L-1}$. One obvious way to encourage uniformity among the non-maximal elements of $F(x)$ is to apply the model regularization method termed label smoothing (Szegedy et al., 2016), which can be done by introducing a cross-entropy term between R_y and $F(x)$ in the CE objective:

$$\mathcal{L}_{CE}^{\lambda}(x, y) = \mathcal{L}_{CE}(x, y) - \lambda \cdot R_y^{\top} \log F(x), \tag{2}$$

where λ is a trade-off parameter. However, it is easy to show that minimizing $\mathcal{L}_{CE}^{\lambda}$ equals to minimizing the cross-entropy between $F(x)$ and the L-dimensional vector P^{λ}:

$$P_i^{\lambda} = \begin{cases} \frac{1}{\lambda+1}, & i = y, \\ \frac{\lambda}{(L-1)(\lambda+1)}, & i \neq y. \end{cases} \tag{3}$$

Note that $1_y = P^0$ and $R_y = P^{\infty}$. When $\lambda > 0$, let $\theta_{\lambda}^* = \arg\min_{\theta} \mathcal{L}_{CE}^{\lambda}$, then the prediction $F(x, \theta_{\lambda}^*)$ will tend to equal to P^{λ}, rather than the ground-truth 1_y. This makes the output predictions be biased. In order to have unbiased predictions that make the output vector $F(x)$ tend to 1_y, and simultaneously encourage uniformity among probabilities on untrue classes, we define another objective function based on what we call *reverse cross-entropy (RCE)* as

$$\mathcal{L}_{CE}^{R}(x, y) = -R_y^{\top} \log F(x). \tag{4}$$

Minimizing RCE is equivalent to minimizing $\mathcal{L}_{CE}^{\infty}$. Note that by directly minimizing $\mathcal{L}_{CE}^{R}$, i.e., $\theta_R^* = \arg\min_{\theta} \mathcal{L}_{CE}^{R}$, one will get a reverse classifier $F(X, \theta_R^*)$, which means that given an input x, the reverse classifier $F(X, \theta_R^*)$ will not only tend to assign the lowest probability to the true class but also tend to output a uniform distribution on other classes. This simple insight leads to our entire RCE training procedure which consists of two parts, as outlined below:

Reverse training: Given the training set $\mathcal{D}$, we train the DNN $F(X, \theta)$ to be a reverse classifier by minimizing the average RCE loss: $\theta_R^* = \arg\min_{\theta} \frac{1}{N} \sum_{d=1}^{N} \mathcal{L}_{CE}^{R}(x_d, y_d)$.

Reverse logits: We negate the final logits fed to the softmax layer as $F_R(X, \theta_R^*) = \mathbb{S}(-Z_{pre}(X, \theta_R^*))$ to get the outputs.

Then we will obtain the network $F_R(X, \theta_R^*)$ that returns ordinary predictions on classes, and $F_R(X, \theta_R^*)$ is referred as the network trained via *the RCE training procedure*.

Theorem 2. *(Proof in Appendix A) Let (x, y) be a given training data. Under the L_{∞}-norm, if there is a training error $\alpha \ll \frac{1}{L}$ that $\|\mathbb{S}(Z_{pre}(x, \theta_R^*)) - R_y\|_{\infty} \leq \alpha$, then we have bounds*

$$\|\mathbb{S}(-Z_{pre}(x, \theta_R^*)) - 1_y\|_{\infty} \leq \alpha(L-1)^2,$$

and $\forall j, k \neq y$,

$$|\mathbb{S}(-Z_{pre}(x, \theta_R^*))_j - \mathbb{S}(-Z_{pre}(x, \theta_R^*))_k| \leq 2\alpha^2(L-1)^2.$$

Theorem 2 demonstrates two important properties of the RCE training procedure. First, it is consistent and unbiased in the sense that when the training error $\alpha \to 0$, the output $F_R(x, \theta_R^*)$ converges to the one-hot label vector 1_y . Second, the upper bounds of the difference between any two non-maximal elements in outputs decrease as $\mathcal{O}(\alpha^2)$ w.r.t. α for RCE, much faster than the $\mathcal{O}(\alpha)$ for CE and label smoothing. These two properties make the RCE training procedure meet our requirements as described above.

3.4. The Thresholding Test Strategy

Given a classifier $F(X)$ trained via the CE or RCE training procedure, we implement a detector by a thresholding test strategy for robust prediction. After presetting a metric and a threshold T, the detector classifies the input as normal and decides to return the predicted label if the value of metric is larger than T, or classifies the one as adversarial and returns NOT SURE otherwise. We introduce three candidate metrics below, and separately test them in Section 4.

Confidence: Given an input x, the corresponding confidence score is calculated as $\text{Confidence}(x) = F(x)_{\hat{y}}$.

non-ME: According to Eq. (1), the non-ME score is calculated as $\text{non-ME}(x) = -\sum_{i \neq \hat{y}} \hat{F}(x)_i \log(\hat{F}(x)_i)$.

Kernel density: Because of the relatively better robustness and versatility of the kernel density estimate method (Feinman et al., 2017; Carlini & Wagner, 2017b), we choose kernel density (K-density) as a candidate metric. The K-density is calculated in the final-layer hidden space. Given the predicted label $\hat{y}$, K-density is defined as $KD(x) = \frac{1}{|X_{\hat{y}}|} \sum_{x_i \in X_{\hat{y}}} k(z_i, z)$, where $X_{\hat{y}}$ represents the set of train-

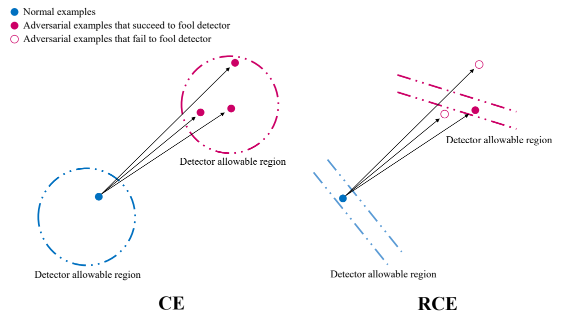

Normal examples
Adversarial examples that succeed to fool detector
Adversarial examples that fail to fool detector
Detector allowable region
Detector allowable region
Detector allowable region
Detector allowable region
CE
RCE

Figure 2. Practical attacks on the trained networks. The blue regions are of the original classes for normal examples, and the red regions are of the target classes for adversarial examples.

ing points with label $\hat{y}$, z_i and z are the corresponding final-layer hidden vectors, $k(z_i, z) = \exp(-\left\|z_i - z\right\|^2 / \sigma^2)$ is the Gaussian kernel with the bandwidth σ treated as a hyper-parameter. K-density can be regarded as some combination of the first two metrics, since it can simultaneously convey the information about them. As shown in Section 4.3, K-density is more reliable in the adversarial setting.

Carlini & Wagner (2017b) show that previous methods on detecting adversarial examples can be evaded by white-box adversaries. Besides, in practice adversaries do not have to fool the detector (i.e., satisfies the restriction in Section 3.2) on all the points it explores when crafting adversarial examples. Namely, adversaries only need to keep the crafted adversarial examples in the detector allowable regions when finally feeding them into the classifiers. However, our method can defend the white-box attacks effectively. This is because the RCE training procedure conceals normal examples on low-dimensional manifolds in the final-layer hidden space, as shown in Fig. 1b and Fig. 1c. Then the detector allowable regions can also be set low-dimensional as long as the regions contain all normal examples. Therefore the white-box adversaries who intend to fool our detector have to generate adversarial examples with preciser calculations and larger noises. This is intuitively illustrated in Fig. 2, where the adversarial examples crafted on the networks trained by CE are easier to locate in the detector allowable regions than those crafted on the networks trained by RCE. This illustration is experimentally verified in Section 4.4.

4. Experiments

We now present the experimental results to demonstrate the effectiveness of our method on improving the robustness of DNN classifiers in the adversarial setting.

4.1. Setup

We use the two widely studied datasets—MNIST (LeCun et al., 1998) and CIFAR-10 (Krizhevsky & Hinton, 2009). MNIST is a collection of handwritten digits in classes 0 to 9. It has a training set of 60,000 images and a test set of 10,000 images. CIFAR-10 consists of 60,000 color images in 10 classes with 6,000 images per class. There are 50,000 training images and 10,000 test images. The pixel values of images in both datasets are scaled to be in the interval $[-0.5, 0.5]$. The normal examples in our experiments refer to all the ones in the training and test sets. In the adversarial setting, the baseline we use is the kernel density estimate method, i.e., CE as the training objective and K-density as the thresholding metric, which has shown its superiority and versatility compared to other detection-based defense methods (Feinman et al., 2017; Carlini & Wagner, 2017b).

Table 1. Classification error rates (%) on test sets.

Method	MNIST	CIFAR-10
DropConnect (Wan et al., 2013)	0.57	9.32
Maxout (Goodfellow et al., 2013)	0.45	9.38
NiN (Lin et al., 2014)	0.47	8.81
FitNet (Romero et al., 2015)	0.51	8.39
DSN (Lee et al., 2015)	0.39	7.97
R-CNN (Liang & Hu, 2015)	0.31	7.09
Resnet-32 (CE)	0.38	7.13
Resnet-32 (RCE)	**0.29**	**7.02**
Resnet-56 (CE)	0.36	**6.49**
Resnet-56 (RCE)	**0.32**	6.60

4.2. Classification on Normal Examples

We first evaluate in the normal setting, where we implement Resnet-32 and Resnet-56 (He et al., 2016a) on both datasets. For each network, we use both the CE and RCE as the training objectives, trained by the same settings as He et al. (2016b). The number of training steps for both objectives is set to be 20,000 on MNIST and 90,000 on CIFAR-10. Hereafter for notation simplicity, we will indicate the training procedure used after the model name of a trained network, e.g., Resnet-32 (CE). Similarly, we indicate the training procedure and omit the name of the target network after an attacking method, e.g., FGSM (CE).

Table 1 shows the test error rates, where the thresholding test strategy is disabled and all the points receive their predicted labels. We can see that the performance of the networks trained by RCE is as good as and sometimes even better than those trained by the traditional CE procedure. Note that we apply the same training hyperparameters (e.g., learning rates and decay factors) for both the CE and RCE procedures, which suggests that RCE is easy to optimize and does not require much extra effort on tuning hyperparameters.

To verify that the RCE procedure tends to map all the normal inputs to the neighborhood of $S_{\hat{y}}$ in the hidden space, we apply the t-SNE technique (Maaten & Hinton, 2008) to visualize the distribution of the final hidden vector z on the test set. Fig. 1b and Fig. 1c give the 2-D visualization results. For clarity, we show the results on 1,000 test examples of CIFAR-10. We can see that the networks trained by RCE can successfully map the test examples to the neighborhood of low-dimensional manifolds in the final-layer hidden space.

Table 2. AUC-scores (10^{-2}) and average distortions of adversarial examples on MNIST and CIFAR-10. The model of target networks is Resnet-32. Values are calculated on the examples which are correctly classified as normal examples and then misclassified as adversarial counterparts. Bandwidths used when calculating K-density are $\sigma_{CE}^2 = 1/0.26$ and $\sigma_{RCE}^2 = 0.1/0.26$. **Boldface** indicates the better combination w.r.t. the objective, and asterisk * indicates the best combination under certain attack.

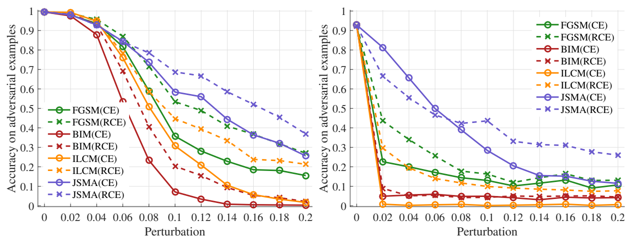
Attack	Objective	MNIST				CIFAR-10			
		Distortion	AUC-scores (10^{-2})			Distortion	AUC-scores (10^{-2})		
			Confidence	non-ME	K-density		Confidence	non-ME	K-density
FGSM	CE	26.92	79.7	66.8	98.8	29.80	71.5	66.9	**99.7***
	RCE	26.63	**98.8**	**98.6**	**99.4***	29.81	**92.6**	**91.4**	98.0
BIM	CE	6.80	88.9	70.5	90.0	6.29	0.0	64.6	**100.0***
	RCE	7.15	**91.7**	**90.6**	**91.8***	5.46	**0.7**	**70.2**	**100.0***
ILCM	CE	6.57	98.4	50.4	96.2	2.51	16.4	37.1	84.2
	RCE	6.60	**100.0***	**97.0**	**98.6**	2.06	**64.1**	**77.8**	**93.9***
JSMA	CE	12.1	98.6	60.1	97.7	1.53	99.2	27.3	85.8
	RCE	12.87	**100.0***	**99.4**	**99.0**	1.40	**99.5***	**91.9**	**95.4**
C&W	CE	8.63	98.6	64.1	99.4	0.65	99.5	50.2	95.3
	RCE	11.04	**100.0***	**99.5**	**99.8**	0.77	**99.6***	**94.7**	**98.2**
C&W-hc	CE	13.70	0.0	40.0	91.1	0.93	0.0	28.8	75.4
	RCE	23.20	**0.1**	**93.4**	**99.6***	1.81	**0.2**	**53.6**	**91.8***

4.3. Performance under the Oblivious Attack

We test the performance of the trained Resnet-32 networks on MNIST and CIFAR-10 under the oblivious attack, where we investigate the attacking methods as in Sec. 2.3. We first disable the thresholding test strategy and make classifiers return all predictions to study the networks ability of correctly classifying adversarial examples. We use the gradient-based attacking methods: FGSM, BIM, ILCM and JSMA, and we calculate the classification accuracy of networks on crafted adversarial examples w.r.t. the perturbation ϵ.

Fig. 3 shows the results. We can see that Resnet-32 (RCE) has higher accuracy scores than Resnet-32 (CE) under all the four attacks on both datasets. This happens because when applying gradient-based methods to attack the networks trained by RCE, the gradients used by the attacking methods are less accurate, which leads to less accurate attacks. However, the adversarial examples crafted by optimization-based methods like C&W can still make Resnet-32 (RCE) misclassify with an 100% success rate, since these methods iterate thousands of times and rely much less on the accuracy of gradients. Note that the attacks on CIFAR-10 are much more efficient than those on MNIST in the sense of requiring smaller perturbations to achieve same attacking success rates, but when adversaries impose large enough perturbations, attacks can still have high success rates on both datasets. That is why the thresholding test strategy is necessary to detect and filter out adversarial examples.

We activate the thresholding test strategy and separately test the performance of three candidate metrics, i.e., *Confidence*, *non-ME* and *K-density*. We construct simple binary classifiers to decide whether an example is adversarial or not by thresholding with different metrics, and then calculate the AUC-scores of ROC curves on these binary classifiers. Table 2 shows the AUC-scores calculated under different combinations of training procedures and thresholding metrics on both datasets. We also calculate the average distortions (Szegedy et al., 2014) as $(\sum_{i=[d]}(x_i^* - x_i)^2/d)^{\frac{1}{2}}$, where x^* is the generated adversarial example and each pixel feature is rescaled to be in the interval $[0, 255]$. Moreover, to make our investigation more convincing, we introduce the high-confidence version of the C&W attack (abbr. to C&W-hc) that sets the parameter κ in the C&W attack to be 10 in our experiments. The C&W-hc attack can generate adversarial examples with the confidence higher than 0.99, and previous work has shown that the adversarial examples crafted by C&W-hc are stronger and more difficult to detect than those crafted by C&W (Carlini & Wagner, 2017a;b).



(a) Results on MNIST (b) Results on CIFAR-10

Figure 3. Classification accuracy on adversarial examples. The model of target networks is Resnet-32.

From Table 2, we can see that our method with K-density as the thresholding metric performs better in almost all cases, and non-ME itself is also a pretty reliable metric, although not as good as K-density. Besides, the C&W attack and its variants (e.g., C&W-hc) need much larger minimal distortions to successfully attack on the networks trained by RCE than on those trained by CE. The similar phenomenon is also observed under the white-box attack. Another noteworthy phenomenon shown in Table 2 is that the classifiers trained by RCE return more reliable confidence, which verifies the conclusion in Theorem 1. Furthermore, we also show that our method can better distinguish between noisy examples and adversarial examples, as demonstrated in Appendix B.3.

4.4. Performance under the White-Box Attack

We test our method under the white-box attack, which is the most difficult threat model and no effective defense exits yet. Due to the prominent performance under the oblivious attack, we select K-density as our thresholding metric. We apply the white-box version of the C&W attack introduced in Carlini & Wagner (2017b), which is constructed specially to fool the K-density detectors. We refer to this attack as C&W-wb. Compared to the C&W attack, C&W-wb introduces a new loss term $f_2(x^*)$ that penalizes the adversarial example x^* being detected by the K-density detectors. The loss $f_2(x^*) = \max(-\log(KD(x^*)) - \eta, 0)$, where η is set to be the median of $-\log(KD(\cdot))$ on the training set.

In practice, we adopt the two-phase implementation procedure of C&W-wb (Carlini & Wagner, 2017b). Table 3 shows the average distortions and the ratios of $f_2(x^*) > 0$ on the adversarial examples crafted by C&W-wb, where a higher ratio indicates that the detector is more robust and harder to be fooled. We find that nearly all the adversarial examples generated on Resnet-32 (CE) have $f_2(x^*) \leq 0$, which means that the values of K-density on them are greater than half of the values on the training data. This result is consistent with previous work (Carlini & Wagner, 2017b).

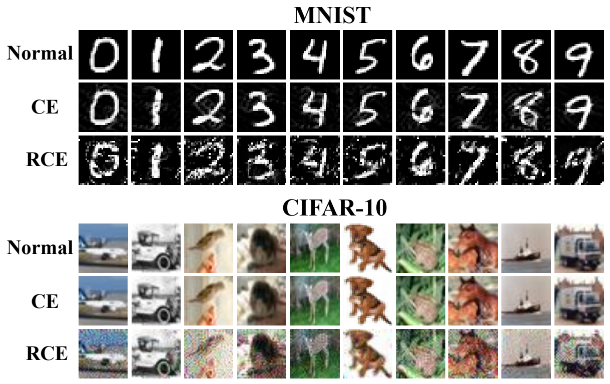


Figure 4. Some normal and adversarial examples from MNIST and CIFAR-10. Adversarial examples are generated by C&W-wb with minimal distortions. The normal test images are termed as *Normal*, and adversarial examples generated on Resnet-32 (CE) and Resnet-32 (RCE) are separately termed as *CE / RCE*.

However, note that applying C&W-wb on our method has a much higher ratio and results in a much larger minimal distortion. Fig. 4 shows some adversarial examples crafted by C&W-wb with the corresponding normal ones. We find that the adversarial examples crafted on Resnet-32 (CE) are indistinguishable from the normal ones by human eyes. In contrast, those crafted on Resnet-32 (RCE) have macroscopic noises, which are not strictly adversarial examples since they are visually distinguishable from normal ones. The inefficiency of the most aggressive attack C&W-wb under our defense verifies our illustration in Fig. 2. Moreover, in C&W-wb we set κ at 0, which means that even though it can fool the K-density detectors, it cannot simultaneously fool the confidence metric. More details on the limitation of C&W-wb are in Appendix B.4. Iterative-based attacking methods cannot effectively fool the K-density detectors even if there are additional special losses in their objective functions, since fooling detectors requires much preciser calculations than fooling classifiers, which can only be efficiently done by optimization-based attacks.

Table 3. The ratios (%) of $f_2(x^*) > 0$ and average distortions of the adversarial examples generated by C&W-wb on MNIST and CIFAR-10. The model is Resnet-32 and the metric is K-density.

Objective	**MNIST**		**CIFAR-10**	
	Ratio	Distortion	Ratio	Distortion
CE	1	17.12	0	1.26
RCE	**77**	**31.59**	**12**	**3.89**

4.5. Performance under the Black-Box Attack

For complete analysis, we investigate the robustness under the black-box attack. The success of the black-box attack is based on the transferability of adversarial examples among different models (Goodfellow et al., 2015). We set the trained Resnet-56 networks as the target models. Adversaries intend to attack them but do not have access to their parameters. Thus we set the trained Resnet-32 networks to be the substitute models that adversaries actually attack on and then feed the crafted adversarial examples into the target models. Since adversaries know the existence of the K-density detectors, we apply the C&W-wb attack. We find that the adversarial examples crafted by the C&W-wb attack have poor transferability, where less than 50% of them can make the target model misclassify on MNIST and less than 15% on CIFAR-10. Table 4 shows the AUC-scores in four different cases of the black-box attack on CIFAR-10, and the AUC-scores in the same cases on MNIST are all higher than 95%. Note that in our experiments the target models and the substitute models have very similar structures, and the C&W-wb attack becomes ineffective even under this quite 'white' black-box attack.

Table 4. AUC-scores (10^{-2}) on CIFAR-10. Resnet-32 is the substitute model and Resnet-56 is the target model.

	Resnet-32 (CE)	Resnet-32 (RCE)
Resnet-56 (CE)	75.0	90.8
Resnet-56 (RCE)	89.1	84.9

5. Conclusions

We present a novel method to improve the robustness of deep learning models by reliably detecting and filtering out adversarial examples, which can be implemented using standard algorithms with little extra training cost. Our method performs well on both the MNIST and CIFAR-10 datasets under all threat models and various attacking methods, while maintaining state-of-the-art accuracy on normal examples.

References

Bhagoji, Arjun Nitin, Cullina, Daniel, and Mittal, Prateek. Dimensionality reduction as a defense against evasion attacks on machine learning classifiers. *arXiv preprint arXiv:1704.02654*, 2017.

Carlini, Nicholas and Wagner, David. Towards evaluating the robustness of neural networks. *IEEE Symposium on Security and Privacy*, 2017a.

Carlini, Nicholas and Wagner, David. Adversarial examples are not easily detected: Bypassing ten detection methods. *ACM Workshop on Artificial Intelligence and Security*, 2017b.

Dong, Yinpeng, Liao, Fangzhou, Pang, Tianyu, Su, Hang, Hu, Xiaolin, Li, Jianguo, and Zhu, Jun. Boosting adversarial attacks with momentum. *arXiv preprint arXiv:1710.06081*, 2017.

Feinman, Reuben, Curtin, Ryan R, Shintre, Saurabh, and Gardner, Andrew B. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

Gong, Zhitao, Wang, Wenlu, and Ku, Wei-Shinn. Adversarial and clean data are not twins. *arXiv preprint arXiv:1704.04960*, 2017.

Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Goodfellow, Ian J, Warde-Farley, David, Mirza, Mehdi, Courville, Aaron, and Bengio, Yoshua. Maxout networks. *International Conference on Machine Learning (ICML)*, 2013.

Goodfellow, Ian J, Shlens, Jonathon, and Szegedy, Christian. Explaining and harnessing adversarial examples. *The International Conference on Learning Representations (ICLR)*, 2015.

Grosse, Kathrin, Manoharan, Praveen, Papernot, Nicolas, Backes, Michael, and McDaniel, Patrick. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.

Gu, Shixiang and Rigazio, Luca. Towards deep neural network architectures robust to adversarial examples. *Conference on Neural Information Processing Systems (NIPS) Workshops*, 2014.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016a.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Identity mappings in deep residual networks. In *European Conference on Computer Vision (ECCV)*, pp. 630–645. Springer, 2016b.

Hornik, Kurt, Stinchcombe, Maxwell, and White, Halbert. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Krizhevsky, Alex and Hinton, Geoffrey. Learning multiple layers of features from tiny images. Technical report, 2009.

Kurakin, Alexey, Goodfellow, Ian, and Bengio, Samy. Adversarial examples in the physical world. *The International Conference on Learning Representations (ICLR) Workshops*, 2017a.

Kurakin, Alexey, Goodfellow, Ian, and Bengio, Samy. Adversarial machine learning at scale. *The International Conference on Learning Representations (ICLR)*, 2017b.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lee, Chen-Yu, Xie, Saining, Gallagher, Patrick, Zhang, Zhengyou, and Tu, Zhuowen. Deeply-supervised nets. In *Artificial Intelligence and Statistics*, pp. 562–570, 2015.

Li, Xin and Li, Fuxin. Adversarial examples detection in deep networks with convolutional filter statistics. *arXiv preprint arXiv:1612.07767*, 2016.

Li, Yingzhen and Gal, Yarin. Dropout inference in bayesian neural networks with alpha-divergences. *International Conference on Machine Learning (ICML)*, 2017.

Liang, Ming and Hu, Xiaolin. Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3367–3375. IEEE, 2015.

Lin, Min, Chen, Qiang, and Yan, Shuicheng. Network in network. *The International Conference on Learning Representations (ICLR)*, 2014.

Liu, Yanpei, Chen, Xinyun, Liu, Chang, and Song, Dawn. Delving into transferable adversarial examples and black-box attacks. *The International Conference on Learning Representations (ICLR)*, 2017.

Maaten, Laurens van der and Hinton, Geoffrey. Visualizing data using t-sne. *Journal of Machine Learning Research (JMLR)*, 9(Nov):2579–2605, 2008.

Metzen, Jan Hendrik, Genewein, Tim, Fischer, Volker, and Bischoff, Bastian. On detecting adversarial perturbations. *The International Conference on Learning Representations (ICLR)*, 2017.

Moosavi-Dezfooli, Seyed-Mohsen, Fawzi, Alhussein, and Frossard, Pascal. Deepfool: a simple and accurate method to fool deep neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2574–2582, 2016.

Nguyen, Anh, Yosinski, Jason, and Clune, Jeff. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 427–436, 2015.

Papernot, Nicolas, McDaniel, Patrick, Goodfellow, Ian, Jha, Somesh, Celik, Z Berkay, and Swami, Ananthram. Practical black-box attacks against deep learning systems using adversarial examples. *arXiv preprint arXiv:1602.02697*, 2016a.

Papernot, Nicolas, McDaniel, Patrick, Jha, Somesh, Fredrikson, Matt, Celik, Z Berkay, and Swami, Ananthram. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pp. 372–387. IEEE, 2016b.

Papernot, Nicolas, McDaniel, Patrick, Wu, Xi, Jha, Somesh, and Swami, Ananthram. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pp. 582–597. IEEE, 2016c.

Romero, Adriana, Ballas, Nicolas, Kahou, Samira Ebrahimi, Chassang, Antoine, Gatta, Carlo, and Bengio, Yoshua. Fitnets: Hints for thin deep nets. *The International Conference on Learning Representations (ICLR)*, 2015.

Rozsa, Andras, Rudd, Ethan M, and Boult, Terrance E. Adversarial diversity and hard positive generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 25–32, 2016.

Rumelhart, David E, Hinton, Geoffrey E, and Williams, Ronald J. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

Szegedy, Christian, Zaremba, Wojciech, Sutskever, Ilya, Bruna, Joan, Erhan, Dumitru, Goodfellow, Ian, and Fergus, Rob. Intriguing properties of neural networks. *The International Conference on Learning Representations (ICLR)*, 2014.

Szegedy, Christian, Vanhoucke, Vincent, Ioffe, Sergey, Shlens, Jon, and Wojna, Zbigniew. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2016.

Wan, Li, Zeiler, Matthew, Zhang, Sixin, Le Cun, Yann, and Fergus, Rob. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning (ICML)*, pp. 1058–1066, 2013.

Zheng, Stephan, Song, Yang, Leung, Thomas, and Goodfellow, Ian. Improving the robustness of deep neural networks via stability training. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

A. Proof

Lemma 1. *In the decision region $dd_{\hat{y}}$ of class $\hat{y}$, $\forall i, j \neq \hat{y}, \widetilde{db_{ij}} \in DB_{ij}$, the value of* non-ME *for any point on the low-dimensional manifold $\bigcap_{i,j\neq\hat{y}} \widetilde{db_{ij}}$ is constant. In particular,* non-ME *obtains its global maximal value* $\log(L-1)$ *on and only on $S_{\hat{y}}$.*

Proof. $\forall i, j \neq \hat{y}$, we take a hyperplane $\widetilde{db_{ij}} \in DB_{ij}$. Then according to the definition of the set DB_{ij}, it is easily shown that $\forall z \in \widetilde{db_{ij}}$, $Z_{pre,i} - Z_{pre,j} = constant$, and we denote this corresponding constant as C_{ij}. Thus given any $k \neq \hat{y}$, we derive that $\forall z \in \bigcap_{i,j\neq\hat{y}} \widetilde{db_{ij}}$

$$\begin{aligned}
\hat{F}(z)_k &= \frac{F(z)_k}{\sum_{j\neq\hat{y}} F(z)_j} \\
&= \frac{\exp(Z_{pre,k})}{\sum_{j\neq\hat{y}} \exp(Z_{pre,j})} \\
&= \frac{1}{\sum_{j\neq\hat{y}} \exp(Z_{pre,j} - Z_{pre,k})} \\
&= \frac{1}{\sum_{j\neq\hat{y}} \exp(C_{jk})} \\
&= constant,
\end{aligned}$$

and according to the defination of the non-ME value non-ME$(z) = -\sum_{i\neq\hat{y}} \hat{F}(z)_i \log(\hat{F}(z)_i)$, we can conclude that non-ME$(z) = constant, \forall z \in \bigcap_{i,j\neq\hat{y}} \widetilde{db_{ij}}$.

In particular, according to the property of entropy in information theory, we know that non-ME $\leq \log(L-1)$, and non-ME achieve its maximal value if and only if $\forall k \neq \hat{y}, \hat{F}_k = \frac{1}{L-1}$. In this case, there is $\forall i, j \neq \hat{y}, Z_{pre,i} = Z_{pre,j}$, which is easy to show that the conditions hold on $S_{\hat{y}}$. Conversely, $\forall z \notin S_{\hat{y}}$, there must $\exists i, j \neq \hat{y}$, such that $Z_{pre,i} \neq Z_{pre,j}$ which leads to $\hat{F}_i \neq \hat{F}_j$. This violates the condition of non-ME achieving its maximal value. Thus non-ME obtains its global maximal value $\log(L-1)$ on and only on $S_{\hat{y}}$.

□

Theorem 1. *In the decision region $dd_{\hat{y}}$ of class $\hat{y}$, $\forall i, j \neq \hat{y}, z_0 \in dd_{\hat{y}}$, there exists a unique $\widetilde{db^0_{ij}} \in DB_{ij}$, such that $z_0 \in Q_0$, where $Q_0 = \bigcap_{i,j\neq\hat{y}} \widetilde{db^0_{ij}}$. Let $Q_0^{\hat{y}} = Q_0 \bigcap \overline{dd_{\hat{y}}}$, then the solution set of the problem*

$$\arg\min_{z_0}(\max_{z^*\in Q_0^{\hat{y}}} F(z^*)_{\hat{y}})$$

is $S_{\hat{y}}$. Furthermore, $\forall z_0 \in S_{\hat{y}}$ there is $Q_0 = S_{\hat{y}}$, and $\forall z^* \in S_{\hat{y}} \bigcap \overline{dd_{\hat{y}}}$, $F(z^*)_{\hat{y}} = \frac{1}{L}$.

Proof. It is easy to show that given a point and a normal vector, one can uniquely determine a hyperplane. Thus $\forall i, j \neq \hat{y}, z_0 \in dd_{\hat{y}}$, there exists unique $\widetilde{db^0_{ij}} \in DB_{ij}$, such that $z_0 \in \bigcap_{i,j\neq\hat{y}} \widetilde{db^0_{ij}} = Q_0$.

According to the proof of Lemma 1, we have $\forall i, j \neq \hat{y}, z^* \in Q_0^{\hat{y}}$, there is $Z_{pre,i} - Z_{pre,j} = C_{ij}$, and $\exists k \neq \hat{y}$, s. t. $Z_{pre,\hat{y}} = Z_{pre,k}$. Thus we can derive

$$\begin{aligned}
F(z^*)_{\hat{y}} &= \frac{\exp(Z_{pre,\hat{y}})}{\sum_i \exp(Z_{pre,i})} \\
&= \frac{1}{1 + \sum_{i\neq\hat{y}} \exp(Z_{pre,i} - Z_{pre,\hat{y}})} \\
&= \frac{1}{1 + \exp(Z_{pre,k} - Z_{pre,\hat{y}})(1 + \sum_{i\neq\hat{y},k} \exp(Z_{pre,i} - Z_{pre,k}))} \\
&= \frac{1}{2 + \sum_{i\neq\hat{y},k} \exp(C_{ik})}.
\end{aligned}$$

Let $M = \{i : C_{ij} \geq 0, \forall j \neq \hat{y}\}$, there must be $k \in M$ so M is not empty, and we have

$$\begin{aligned}
\max_{z^*\in Q_0^{\hat{y}}} F(z^*)_{\hat{y}} &= \max_{l\in M} \frac{1}{2 + \sum_{i\neq\hat{y},l} \exp(C_{il})} \\
&= \frac{1}{2 + \min_{l\in M} \sum_{i\neq\hat{y},l} \exp(C_{il})} \\
&= \frac{1}{2 + \sum_{i\neq\hat{y},\widetilde{k}} \exp(C_{i\widetilde{k}})},
\end{aligned}$$

where $\widetilde{k}$ is any element in M. This equation holds since $\forall k_1, k_2 \in M$, there is $C_{k_1k_2} \geq 0$, $C_{k_2k_1} \geq 0$ and $C_{k_1k_2} = -C_{k_2k_1}$, which leads to $C_{k_1k_2} = C_{k_2k_1} = 0$. Therefore, $\forall l \in M$, $\sum_{i\neq\hat{y},l} \exp(C_{il})$ has the same value.

This equation consequently results in

$$\begin{aligned}
\arg\min_{z_0}(\max_{z^*\in Q_0^{\hat{y}}} F(z^*)_{\hat{y}}) &= \arg\min_{z_0} \frac{1}{2 + \sum_{i\neq\hat{y},\widetilde{k}} \exp(C_{i\widetilde{k}})} \\
&= \arg\max_{z_0} \sum_{i\neq\hat{y},\widetilde{k}} \exp(C_{i\widetilde{k}}).
\end{aligned}$$

From the conclusion in Lemma 1, we know that the value $\sum_{i\neq\hat{y},\widetilde{k}} \exp(C_{i\widetilde{k}})$ obtains its maximum when $C_{i\widetilde{k}} = 0, \forall i \neq \hat{y}, \widetilde{k}$. Thus the solution set of the above problem is $S_{\hat{y}}$. Furthermore, we have $\forall z^* \in S_{\hat{y}} \bigcap \overline{dd_{\hat{y}}}$, $F(z^*)_{\hat{y}} = \frac{1}{2+L-2} = \frac{1}{L}$.

□

Theorem 2. *Let (x, y) be a given training data. Under the L_∞-norm, if there is a training error $\alpha \ll \frac{1}{L}$ that $\|\mathbb{S}(Z_{pre}(x, \theta_R^*)) - R_y\|_\infty \leq \alpha$, then we have bounds*

$$\|\mathbb{S}(-Z_{pre}(x, \theta_R^*)) - 1_y\|_\infty \leq \alpha(L-1)^2,$$

and $\forall j, k \neq y$,

$$|\mathbb{S}(-Z_{pre}(x, \theta_R^*))_j - \mathbb{S}(-Z_{pre}(x, \theta_R^*))_k| \leq 2\alpha^2(L-1)^2.$$

Proof. For simplicity we omit the dependence of the logits Z_{pre} on the input x and the parameters θ_R^*. Let $G = (g_1, g_2, ..., g_L)$ be the exponential logits, where $g_i = \exp(Z_{pre,i})$. Then from the condition $\|\mathbb{S}(Z_{pre}) - R_y\|_\infty \leq \alpha$ we have

$$\begin{cases} \frac{g_y}{\sum_i g_i} \leq \alpha \\ \left|\frac{g_j}{\sum_i g_i} - \frac{1}{L-1}\right| \leq \alpha \quad j \neq y. \end{cases}$$

Let $C = \sum_i g_i$, we can further write the condition as

$$\begin{cases} g_y \leq \alpha C \\ (\frac{1}{L-1} - \alpha)C \leq g_j \leq (\frac{1}{L-1} + \alpha)C \quad j \neq y. \end{cases}$$

Then we can have bounds ($L \geq 2$)

$$\begin{aligned} \mathbb{S}(-Z_{pre})_y &= \frac{\frac{1}{g_y}}{\frac{1}{g_y} + \sum_{i \neq y} \frac{1}{g_i}} \\ &= \frac{1}{1 + \sum_{i \neq y} \frac{g_y}{g_i}} \\ &\geq \frac{1}{1 + \sum_{i \neq y} \frac{\alpha C}{(\frac{1}{L-1} - \alpha)C}} \\ &= \frac{1}{1 + \frac{\alpha(L-1)^2}{1 - \alpha(L-1)}} \\ &= 1 - \frac{\alpha(L-1)^2}{1 - \alpha(L-1) + \alpha(L-1)^2} \\ &\geq 1 - \alpha(L-1)^2 \end{aligned}$$

and $\forall j \neq y$,

$$\begin{aligned} \mathbb{S}(-Z_{pre})_j &= \frac{\frac{1}{g_j}}{\frac{1}{g_y} + \sum_{i \neq y} \frac{1}{g_i}} \\ &= \frac{\frac{g_y}{g_j}}{1 + \frac{g_y}{g_j} + \sum_{i \neq y, j} \frac{g_y}{g_i}} \\ &\leq \frac{\frac{g_y}{g_j}}{1 + \frac{g_y}{g_j}} \\ &= \frac{1}{1 + \frac{g_j}{g_y}} \\ &\leq \frac{1}{1 + \frac{(\frac{1}{L-1} - \alpha)C}{\alpha C}} \\ &= \alpha(L-1) \\ &\leq \alpha(L-1)^2. \end{aligned}$$

Then we have proven that $\|\mathbb{S}(-Z_{pre}) - 1_y\|_\infty \leq \alpha(L-1)^2$. Furthermore, we have $\forall j, k \neq y$,

$$\begin{aligned} |\mathbb{S}(-Z_{pre})_j - \mathbb{S}(-Z_{pre})_k| &= \frac{\left|\frac{1}{g_j} - \frac{1}{g_k}\right|}{\frac{1}{g_y} + \sum_{i \neq y} \frac{1}{g_i}} \\ &\leq \frac{\frac{1}{(\frac{1}{L-1} - \alpha)C} - \frac{1}{(\frac{1}{L-1} + \alpha)C}}{\frac{1}{\alpha C} + \sum_{i \neq y} \frac{1}{(\frac{1}{L-1} + \alpha)C}} \\ &= \frac{\frac{L-1}{1 - \alpha(L-1)} - \frac{L-1}{1 + \alpha(L-1)}}{\frac{1}{\alpha} + \frac{(L-1)^2}{1 + \alpha(L-1)}} \\ &= \frac{2\alpha^2(L-1)^2}{1 + \alpha(L-1)^2(1 - \alpha L)} \\ &\leq 2\alpha^2(L-1)^2. \end{aligned}$$

□

B. Additional Experiments

B.1. Training Settings

We apply the same hyperparameters when training Resnet networks via the CE and RCE. The optimizer is SGD with momentum, and the mini-batch size is 128. The weight decay is 0.0002, the leakiness of Relu is 0.1.

On MNIST the training steps are 20,000, with piecewise learning rate as

$$\text{steps:}[10,000, 15,000, 20,000],$$

$$\text{lr:}[0.1, 0.01, 0.001, 0.0001].$$

Each training image pixel values are scaled to be in the interval $[-0.5, 0.5]$.

On CIFAR-10 the training steps are 90,000, with piecewise learning rate as

$$\text{steps:}[40,000, 60,000, 80,000],$$

$$\text{lr:}[0.1, 0.01, 0.001, 0.0001].$$

The training set is augmented by two ways as

- Resizing images to $40 \times 40 \times 3$ and then randomly cropping them back to $32 \times 32 \times 3$.
- Randomly flipping images along their second dimension, which is width.

After augmentation each training image pixel values are also scaled to be in the interval $[-0.5, 0.5]$.

B.2. Time Costs on Crafting Adversarial Examples

Our experiments are done on NVIDIA Tesla P100 GPUs. We set the binary search steps to be 9 and the maximal iteration steps to be 10,000 in C&W-family attacks (i.e., C&W, C&W-hc and C&W-wb), which promises large enough searching capacity for these attacks. We set the maximal iteration steps to be 100 for JSMA, which means that JSMA perturbs at most 100 pixels on each image. Table 5 demonstrates the average time costs on crafting each adversarial example via different attacks. We can find that C&W-family attacks are extremely time consuming compared to other iterative methods. Furthermore, C&W-family attacks usually take longer time to attack the networks trained by the RCE than those trained by the CE.

Table 5. The average time costs (s) on crafting each adversarial example via different attacks. The values are also the average values between MNIST and CIFAR-10. The models is Resnet-32.

Attack	Objective	Time
FGSM	CE	$\sim 1.9 \times 10^{-3}$
	RCE	$\sim 2.4 \times 10^{-3}$
BIM	CE	$\sim 3.3 \times 10^{-3}$
	RCE	$\sim 3.6 \times 10^{-3}$
ILCM	CE	$\sim 4.1 \times 10^{-3}$
	RCE	$\sim 4.3 \times 10^{-3}$
JSMA	CE	$\sim 2.9 \times 10^{1}$
	RCE	$\sim 2.0 \times 10^{1}$
C&W	CE	$\sim 4.5 \times 10^{1}$
	RCE	$\sim 5.5 \times 10^{1}$
C&W-hc	CE	$\sim 6.5 \times 10^{1}$
	RCE	$\sim 1.1 \times 10^{2}$
C&W-wb	CE	$\sim 7.0 \times 10^{2}$
	RCE	$\sim 1.3 \times 10^{3}$

B.3. Robustness to Noisy Examples

For more complete analysis, we investigate whether our method can distinguish between noisy examples and adversarial examples. The noisy examples (RAND) here are defined as

$$x^* = x + U(-\epsilon, \epsilon)$$

where $U(-\epsilon, \epsilon)$ denotes an element-wise distribution on the interval $[-\epsilon, \epsilon]$. Fig. 5 gives the classification error rates on the test set of CIFAR-10, where $\epsilon_{RAND} = 0.04$. We find that the networks trained by both the CE and RCE are robust to noisy examples in the sense of having low error rates.

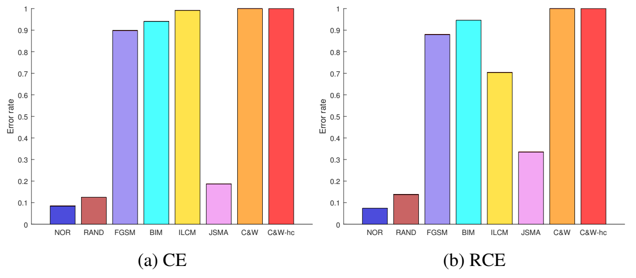


Figure 5. Classification error rates on CIFAR-10. Two panels separately show the results when the networks are trained via the CE and RCE. The models is Resnet-32.

Furthermore, in Fig. 6 and Fig. 7, we show the number of images w.r.t. the values of K-density under various attacks, also on normal and noisy examples. We work on 1,000 test images of CIFAR-10, and our baseline is the kernel density estimate method (CE as the objective and K-density as the metric). We can see that the baseline returns quite different distributions on K-density between normal and noisy examples, and it cannot distinguish noisy examples from the adversarial ones crafted by, e.g., JSMA and C&W-hc, as shown in Fig. 6. In comparison, our method (RCE as the objective and K-density as the metric) returns similar distributions on K-density between normal and noisy examples, and noisy examples can be easily distinguished from other adversarial ones, as shown in Fig. 7.

B.4. The Limitation of C&W-wb

When we apply the C&W-wb attack, the parameter κ is set to be 0. This makes C&W-wb succeed to fool the K-density detector but fail to fool the confidence metric. Thus we construct a high-confidence version of C&W-wb, where we set κ be 5. However, none of the crafted adversarial examples can have $f_2(x^*) \leq 0$, as shown in Table 6. This means that it is difficult for C&W-wb to simultaneously fool both the confidence and the K-density metrics.

Table 6. The ratios (%) of $f_2(x^*) > 0$ of the adversarial examples crafted by the high-confidence version of C&W-wb on MNIST and CIFAR-10. The model is Resnet-32 and the metric is K-density.

Objective	**MNIST**	**CIFAR-10**
CE	100	100
RCE	100	100

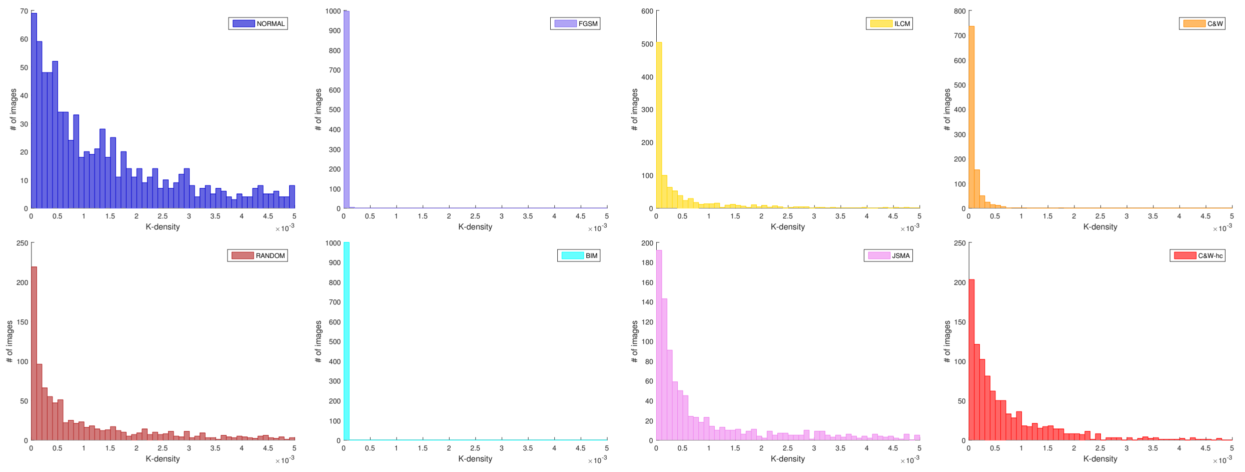


Figure 6. Number of images w.r.t. K-density. The target networks are trained by the CE.

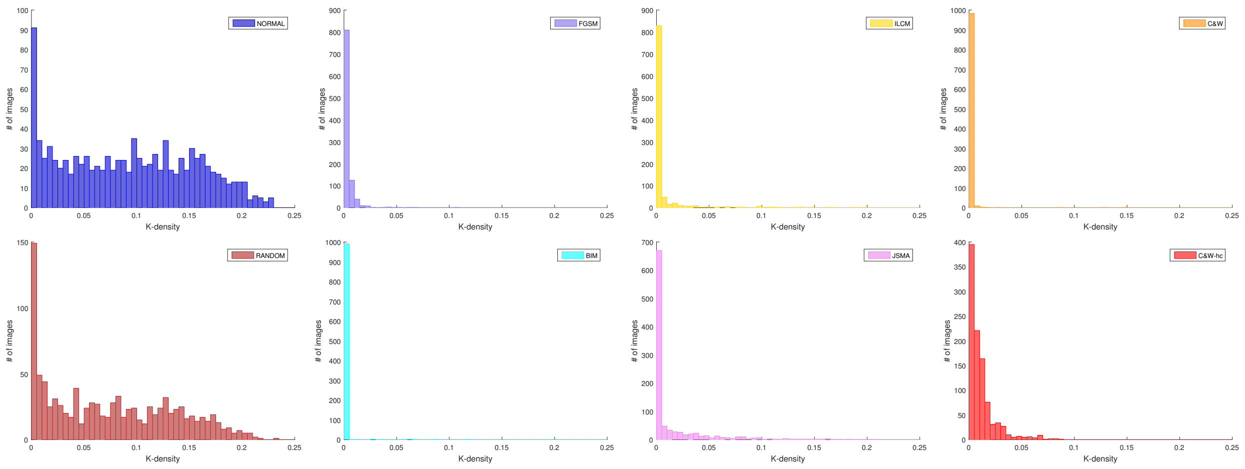


Figure 7. Number of images w.r.t. K-density. The target networks are trained by the RCE.