Robust Deep Learning via Reverse Cross-Entropy Training and Thresholding Test

Tianyu Pang, Chao Du, Jun Zhu[*]
Dept. of Comp. Sci. & Tech., TNList Lab; Center for Bio-Inspired Computing Research
State Key Lab for Intell. Tech. & Systems, Tsinghua University, Beijing, China
{pangty13@mails, du-c14@mails, dcszj@mail}.tsinghua.edu.cn

Abstract

Though the recent progress is substantial, deep learning methods can be vulnerable to the elaborately crafted adversarial samples. In this paper, we attempt to improve the robustness by presenting a new training procedure and a thresholding test strategy. In training, we propose to minimize the reverse cross-entropy, which encourages a deep network to learn latent representations that better distinguish adversarial samples from normal ones. In testing, we propose to use a thresholding strategy based on a new metric to filter out adversarial samples 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 various state-of-the-art networks (e.g., residual networks) and we achieve significant improvements on robust predictions in the adversarial setting.

1 Introduction

Deep learning (DL) has obtained substantial progress in many tasks, including image categorization, speech recognition, and natural language processing [3]. However, a high-accuracy DL model can be vulnerable in the adversarial setting [4, 26], where adversarial samples are carefully crafted to mislead the model to wrong predictions. Several algorithms have been developed to craft such adversarial samples [4, 12, 15, 21, 22]. As DL is becoming ever more prevalent, it is vital to improve the robustness, especially in critical domains (e.g., self-driving cars, healthcare and finance).

Various defense strategies have been proposed to improve the robustness of DL. The adversarial training strategy [26] dynamically augments the training set with crafted adversarial samples. Though it promises an efficient defense on some specific and known attacking methods, it requires extra computational cost and is not a general strategy for unknown attacking methods. [5] considers adding a regularization term on the norm of gradients in the objective function, but it does not promise anything to gradient sign attacking methods that only require the sign of gradients. The defensive distillation method [23] trains a model with a high temperature and tests it with a low temperature. It can work well, but it is heuristic and needs to tune the training temperature. Furthermore, this defensive distillation method can be easily attacked by new attacking methods [1]. Overall, as adversarial samples always exist for a fixed parametric model (thus a fixed decision boundary), it is unlikely for such methods to solve the problem by preventing attackers from crafting adversarial samples, no matter how hard to find them.

In this paper, we attempt to improve the robustness of DL from a new angle. Instead of preventing adversarial samples, we try to distinguish adversarial samples from normal ones, so that we can filter out the adversarial ones for robust predictions. We make contributions by presenting a new method that consists of a novel training procedure and a thresholding test strategy. For training, we propose to minimize a new objective function, named as reverse cross-entropy (RCE), instead of minimizing

[*]Corresponding author.

the common cross-entropy (CE) loss [3]. By minimizing RCE, our training procedure encourages the DL classifiers to map the normal samples to the neighborhoods of low dimensional manifolds in the hidden space, and it also encourages the classifiers to return a high confidence on the true class while a uniform distribution on false classes for each data point. The minimization of RCE is simple to implement using stochastic gradient descent methods, and introduces little extra training cost, as compared to CE. Therefore, it can be easily applied to any deep neural networks.

For testing, we propose a thresholding strategy based on a new compositive metric of J-score, which simultaneously measures the degree of certainty about the given input being a normal sample and the probability of belonging to a selected class. By setting a proper threshold on the J-score, we can filter out adversarial samples for robust predictions, namely, we make the classifiers return meaningful predictions only when the J-score is higher than a given threshold and refuse to predict otherwise.

We apply our method to improve the state-of-the-art residual neural networks (Resnets) [6] on the widely used MNIST [13] and CIFAR-10 [11] datasets. The results show that we improve the robustness on adversarial samples while maintaining state-of-the-art accuracy on normal samples. Finally, we note that recent work [14] shows that a Bayesian neural network (BNN) can improve the robustness of DNN against adversarial samples. Our method is much simpler and computationally cheaper than BNNs which need to do expensive posterior inference. Moreover, our quantitative results demonstrate a better performance than BNNs on filtering adversarial samples.

2 Background

2.1 Neural networks

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. Here, we focus on the DNNs with softmax output layers. For notation clarity, we define the softmax function $\mathrm{softmax}(z) : \mathbb{R}^L \to \mathbb{R}^L$ as $\mathrm{softmax}(z)_i = \exp(z_i)/\sum_{i=1}^{L} \exp(z_i), i \in [L]$, where $[L] := \{1, \cdots, L\}$. Let Z_f be the output vector of the final hidden layer. This defines a mapping function: $X \to Z_f$ to extract good representations. Then, the classifier can be expressed as $F(X, \theta) = \mathrm{softmax}(W_f Z_f + b_f)$, where W_f and b_f are the weight matrix and bias vector of the softmax layer respectively. We denote $Z(X, \theta) := W_f Z_f + b_f$ as the logits. We can easily see that the softmax classifier has a piecewise linear decision boundary in the final hidden space Z_f. Given an input x (i.e., an instance of X), the output of the classifier is a L-dimensional probability vector, with each element $F(x, \theta)_i$ being the probability that x belongs to class i. The predicted label for x is denoted as $\hat{t}(x, \theta) = \arg\max_i F(x, \theta)_i$. The probability value $F(x, \theta)_{\hat{t}(x,\theta)}$ is often used as the corresponding confidence score on this prediction [3], although it is insufficient as we shall see.

Let $\mathcal{D} := \{(x_i, y_i)\}_{i \in [N]}$ be a training set with N input-label pairs, where $y_i \in \mathbb{R}^L$ is an one-hot vector indicating the true class of x_i, i.e., $y_{ij} = 1$ if x_i belongs to class j; 0 otherwise. One common training objective is to minimize the cross-entropy (CE) loss, which is defined as:

$$\mathcal{L}_{CE}(x, y) = -y \cdot \log F(x, \theta) = -\log F(x, \theta)_t,$$

for a single pair (x, y), where t is the class of x. Then the CE training procedure intends to minimize the average CE loss 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 [10, 24].

2.2 Adversarial sample crafting and robustness

Though DNN has obtained substantial progress, adversarial samples can be easily identified to fool the network, even when its accuracy is high [19]. We consider the white-box attack, where attackers know everything about the target classifier $F(X, \theta)$ including the architecture, parameters, objective function and training tools. This setting is interesting because if we can effectively defend adversarial samples under the white-box attack, we can do at least as well in other adversarial settings where attackers have less knowledge on the target classifier. In the white-box setting, attackers can design different algorithms to craft adversarial samples based on their knowledge. In our experiments, we consider the fast gradient sign method (FGSM) [4], which is efficient and has been widely studied [1, 4, 15, 20, 23]. Specifically, let x be a normal input. FGSM crafts an adversarial sample x^* by forcing the target model to classify x^* into a class different from that of x. This is achieved by running the update: $x^* \leftarrow x + \varepsilon \cdot \mathrm{sgn}(\nabla_x J(F, \theta, x))$, where $J(F, \theta, x)$ is the objective function to train the target classifier and ε controls the magnitude of perturbation on normal inputs. Moreover, to evaluate the ability of a classifier to resist adversarial perturbation, we define the robustness $\Delta(x;\hat{t})$ according to [17] as $\Delta(x;\hat{t}) := \min_{r} \|r\|_2$ subject to $\hat{t}(x+r) \neq \hat{t}(x)$, where r is the perturbation.

3 Algorithms

In this section, we present a new algorithm to improve the robustness against adversarial samples. We first provide some new insights on why adversarial samples exist and how can we distinguish them. These insights guide us to a new algorithm.

3.1 Why adversarial samples exist

Previous work [4, 22] hypothesizes that the existence of adversarial samples is because of certain defects in the training phase. One main support is from the universal approximator theorem [8] that DNNs are able to represent functions that resist adversarial samples. However, in practice any given DNN has an architecture of limited scale, which results in a limited representation capability, and the existence of adversarial samples is intrinsic, as explained below.

For a given DNN classifier, its decision boundary is fixed. Then, any pair of similar instances that are located on different sides of a decision boundary will be classified into different classes. Very often, such a pair of input points are not distinguishable by human observers; thus, it sounds counterintuitive and irrational to have a jump on the predicted labels. The previous defense attempts on adjusting the training phase, e.g., by adding regularizers in the objective function [5] or augmenting the training dataset [26], will only result in the change of the distribution of decision boundaries but the jump on the predicted labels nearby the decision boundary still exists. Therefore, a smart enough attacker can always find adversarial samples nearby the decision boundaries to successfully attack the target classifier no matter how the decision boundaries change, as has been demonstrated in [26].

In consideration of the intrinsic existence of adversarial samples, instead of designing defense strategies for the classifiers to prevent attackers from crafting adversarial samples, we design a defense strategy to help the classifiers to distinguish adversarial samples from normal ones so as to filter adversarial samples out for robust predictions.

3.2 The insufficiency of confidence and a new metric of non-ME

We need some metrics to distinguish whether an input x is adversarial or not for a given classifier $F(X,\theta)$. A potential candidate is the confidence $F(x,\theta)_{\hat{t}(x,\theta)}$ on the predicted label $\hat{t}(x,\theta)$, which has in fact been widely used [3]. As F predicts incorrectly on adversarial samples, intuitively we should have a higher confidence on a normal sample than that on an adversarial sample, which consequently allows us to distinguish them by the confidence values. However, it has been shown that a well-trained DNN classifier usually not only misclassifies adversarial samples but also gives high confidence on its predictions [4, 19], which renders the confidence unreliable in the adversarial setting. This is because in the normal setting without attackers, the distribution of 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, attackers can explore the points far from the data distribution [14], and the predictions in these domains mostly result from interpolation with high uncertainty. Therefore, a high confidence returned in these domains is unreliable and probably a false positive.

To avoid the unreliability of using confidence as a single 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 $F(x,\theta)$, as:

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

where $\hat{F}(x,\theta)_i = \frac{F(x,\theta)_i}{\sum_{j\neq\hat{t}(x,\theta)} F(x,\theta)_j}$ are the normalized non-maximal elements in $F(x,\theta)$.

It is easy to show that $\text{non-ME}(x,\theta) \leq \log(L-1)$ and the equality holds if and only if all the non-maximal elements in $F(x,\theta)$ are equal, independent of the confidence score. Therefore, the non-ME metric conveys certain information in $F(x,\theta)$ that is ignored by the confidence. Note that for piecewise linear classifiers (e.g., DNNs with a softmax output layer), any input point that locates on the oppositely elongated decision boundaries has at least two equal non-maximal elements in $F(x,\theta)$. Therefore, any input point that has a high value of $\text{non-ME}(x,\theta)$ and $(L-1)$ approximately equal non-maximal elements in $F(x, \theta)$ must locate nearby the junction manifolds of at least $(L-2)$ oppositely elongated decision boundaries.

For a DNN classifier $F(x, \theta) = \mathrm{softmax}(W_f z_f + b_f)$, if the learned representation transformation: $x \to z_f$ can map the normal inputs to the neighborhoods of oppositely elongated decision boundaries in the final hidden space, then using the two metrics of confidence and non-ME will improve the robustness in the final-layer hidden space and further result in an improvement in the input space. To illustrate this idea and show the effect of non-ME, Fig. 1 presents an example in the final hidden space, where $z_f \in \mathbb{R}^2$ and $L = 3$. In this case, all the points that locate on any one oppositely elongated decision boundary have two equal non-maximal elements in $F(x, \theta)$, thus have highest values of $\mathrm{non\text{-}ME}(x, \theta) = \log 2$. A good way to use non-ME to filter out adversarial samples is to enforce all the normal points have non-ME values higher than a threshold l, which indicates that the normal instances are located in the neighborhood of oppositely elongated decision boundaries in the final hidden space. Therefore, given any unidentified input (i.e., adversarial sample), if its non-ME value is less than l the classifier will immediately know that this input is not normal and filters it out.

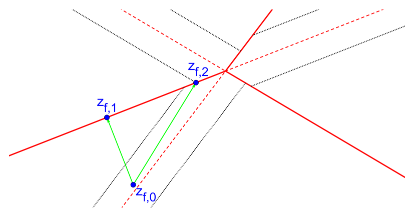


Figure 1: The red solid lines are the decision boundaries, the red dashed lines are the oppositely elongated decision boundaries and the black dot lines are the isolines of $\mathrm{non\text{-}ME} = l$ in the corresponding decision domains.

As shown in Fig. 1, $z_{f,0}$ is a normal instance located in the neighborhood of the oppositely elongated decision boundaries, where the neighborhood boundaries are the isolines of $\mathrm{non\text{-}ME} = l$ in the correspond decision domains. Let $z_{f,0}$ be the original input, from which an attacker tries to craft adversarial samples. When confidence is the only metric, the nearest successful adversarial sample $z_{f,1}$ locates on the nearest decision boundary w.r.t $z_{f,0}$. In contrast, when both confidence and non-ME are used, $z_{f,1}$ will be easily filtered out by the classifier because $\mathrm{non\text{-}ME}(z_{f,1}, \theta)$ is less than l, and the nearest successful adversarial sample $z_{f,2}$ in this case locates on the nearest junction manifolds of neighborhood boundaries and decision boundaries w.r.t $z_{f,0}$. It is easy to find that the minimal perturbation $\|z_{f,0} - z_{f,2}\|_2$ is larger than $\|z_{f,0} - z_{f,1}\|_2$ (equal on a zero measure set in the input space), which means that using two metrics can improve robustness $\Delta(z_{f,0}; \hat{t})$, a.e. Besides, higher values of the non-ME threshold l can lead to larger minimal perturbation $\|z_{f,0} - z_{f,2}\|_2$ and better robustness $\Delta(z_{f,0}; \hat{t})$. It is easy to prove that these conclusions are tenable in more general cases of different values of L and different dimensions of the final hidden space.

3.3 Reverse cross-entropy

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 neighborhoods of oppositely elongated decision boundaries in the final hidden space. This can be achieved by letting the non-maximal elements of $F(x, \theta)$ be as equal as possible and have high non-ME values for all the normal inputs. Specifically, for a training data (x, y), where t is the index of the true label, we let y_R denote its reverse label vector whose tth element is zero and other elements equal to $\frac{1}{L-1}$. One obvious way to achieve the above goal is to add a cross-entropy term between y_R and $F(x, \theta)$ in the traditional cross-entropy objective:

$$\mathcal{L}_{CE}^{\lambda}(x, y) = \mathcal{L}_{CE}(x, y) - \lambda y_R \cdot \log F(x, \theta) = -\log F(x, \theta)_t - \frac{\lambda}{L-1} \sum_{i \neq t} \log F(x, \theta)_i, \quad (2)$$

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 $F(x, \theta)$ and the N-dimensional probability vector P^{λ}:

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

Note that there are $y = P^0$ and $y_R = P^{\infty}$. Let $\theta_{\lambda}^* = \arg\min_{\theta} \mathcal{L}_{CE}^{\lambda}$, then the vector $F(x, \theta_{\lambda}^*)$ will tend to equal to P^{λ}, rather than the ground-truth y. This makes the estimate of θ by minimizing $\mathcal{L}_{CE}^{\lambda}$ be biased. In order to have an unbiased estimator of θ meeting the requirements that simultaneously make the output vector $F(x, \theta)$ tend to y and encourage uniformity among probabilities on untrue classes, we define another objective function based on what we call reverse cross-entropy (RCE) as

$$\mathcal{L}_{CE}^{R}(x, y) = -y_R \cdot \log F(x, \theta) = -\frac{1}{L-1} \sum_{i \neq t} \log F(x, \theta)_i. \quad (4)$$

Minimizing the 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 'liar' classifier $F(X, \theta_R^*)$, which means that given an input x, the 'liar' classifier $F(X, \theta_R^*)$ will not only tend to conceal the index of true class by assigning a low probability to it but also tend to output a uniform distribution on other classes.

3.4 The reverse cross-entropy training procedure

It is easy to know the index that $F(X, \theta_R^*)$ wants to conceal the most (i.e., assign the lowest probability) is actually the true label. This simple insight leads to our 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 'liar' 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 to calculate the output predictions as $F_R(X, \theta_R^*) = \mathrm{softmax}(-Z(X, \theta_R^*))$.

Then we will obtain the reverse 'liar' network $F_R(X, \theta_R^*)$ that returns normal distribution on classes. Theorem 1 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 y . Second, the upper bounds of the difference between any two non-maximal elements in outputs decrease much faster in the RCE training than in the CE training w.r.t the training error α, where the upper bounds decrease as $\mathcal{O}(\alpha^2(L-1)^2)$ for RCE while $\mathcal{O}(\alpha)$ for CE. These two properties make the RCE training procedure meet our requirements as described above.

Theorem 1. *(Proof in Appendix) Let x be a given input in the training dataset, y be the one-hot label vector, y_R be the reverse label vector corresponding to y, L be the number of different classes and θ_R^* is obtained from the reverse training. Under the L_∞-norm, if there is a training error $\alpha \ll \frac{1}{L}$ that*

$$\left\|\mathrm{softmax}(Z(x, \theta_R^*)) - y_R\right\|_\infty \leq \alpha.$$

Then we have bounds $\left\|\mathrm{softmax}(-Z(x, \theta_R^*)) - y\right\|_\infty \leq \alpha(L-1)^2$ *and* $\forall j, k \neq t$

$$|\mathrm{softmax}(-Z(x, \theta_R^*))_j - \mathrm{softmax}(-Z(x, \theta_R^*))_k| \leq 2\alpha^2(L-1)^2,$$

where t is the index of true label.

3.5 The thresholding test procedure

Given a trained reverse 'liar' classifier $F_R(X, \theta_R^*)$, we present a thresholding test procedure for robust prediction. The test procedure consists of three key steps, as explained below.

Calculate the predicted label and confidence: Given a testing input x, we first calculate the output vector $F_R(x, \theta_R^*)$ and then assign the predicted label of x to be $\hat{t} = \arg\max_i F_R(x, \theta_R^*)_i$. Thus, the corresponding confidence is $\mathrm{Confidence} = F_R(x, \theta_R^*)_{\hat{t}}$.

Calculate the kernel non-ME: We let non-ME_R denote the non-ME value as defined in Eq. (1), where $F_R(x, \theta_R^*)$ substitutes $F(x, \theta)$. Since in practice non-ME changes much slower than confidence, instead of directly using non-ME_R as a metric, we use the kernel non-ME defined as

$$\text{non-ME}^{Ker} = \exp(-(\text{non-ME}_R - \mu)^2/\eta\sigma^2). \tag{5}$$

Here μ and σ are the kernel parameters, and η is the relaxation factor. As we will see in experiments, a good choice of μ and σ are $\log(L-1)$ and the standard deviation of the non-ME values to $\log(L-1)$ on the training set, and we simply set the relaxation factor η to be 1.

Threshold-output strategy on the J-score: In the prediction release phase, we multiply the values of $\mathrm{Confidence}$ and non-ME^{Ker} to obtain a joint score (J-score) to measure a compositive certainty on a prediction. J-score has the same value range as the confidence (i.e., $[0, 1]$). A high J-score implies that the classifier is quite sure about both the label and the normality of the input x, while a low J-score implies uncertainty on either the label or the normality of the input x, or even uncertainty on both. Therefore, given a threshold T on the J-score, we decide to return a meaningful pair of label index and J-score if J-score $\geq T$ or return NOT SURE otherwise.

4 Experiments

We now present both quantitative and qualitative results to demonstrate the effectiveness of our method on improving the robustness of state-of-the-art DNN classifiers.

4.1 Setup

We use the two widely studied datasets—MNIST and CIFAR-10. MNIST is a collection of 70,000 images of handwritten digits in classes 0 to 9. It has a training set of 60,000 samples and a test set of 10,000 samples. 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. Every image in both sets is standardized before feeding to classifiers to remove interval dependence among pixels, following the setup in [6].

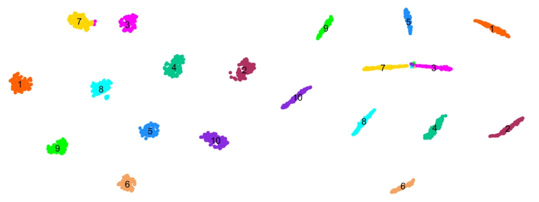
We implement Resnet-32, Resnet-56 and Resnet-110 [6] on both datasets. For each network, we use both the CE and RCE as the training objective, optimized by the same training tools as in [7]. The training steps for both methods are set to be 20k on MNIST and 80k on CIFAR-10. Table 1 shows the test accuracy, where the Threshold-output strategy is disabled and all points receive their predicted labels. The results show that the classification performance of the networks trained via the RCE procedure is almost as good as those trained by the traditional CE procedure. Due to limited space, we only show the results on Resnet-32 in the sequel, while deferring the results on Resnet-56 and Resnet-110 and other DNN architectures to Appendix, which essentially lead to the same conclusions.

Table 1: Overview of Test Accuracy

Model & Training procedure	Accuracy (MNIST)	Accuracy (CIFAR-10)
Resnet-32 & CE	99.28%	92.23%
Resnet-32 & RCE	99.21%	92.10%
Resnet-56 & CE	99.30%	93.30%
Resnet-56 & RCE	99.30%	92.59%
Resnet-110 & CE	99.36%	93.57%
Resnet-110 & RCE	99.23%	92.94%

4.2 Junction manifolds in the final hidden space

In order to verify that the RCE training procedure tends to map all the normal inputs to the neighborhoods of oppositely elongated decision boundaries in the final hidden space, we apply the technique t-SNE [16] to visualize the distribution of the final hidden vector z_f on the training set. Fig. 2 shows the 2-D visualization results. For clarity, we show the results on 1,000 training samples of MNIST. We defer the results on CIFAR-10 to Appendix, which are similar. From the results we note that the networks trained via the RCE procedure can successfully map the training samples to the neighborhoods of low-dimensional manifolds in the final hidden space. These results also partly verify the effectiveness of the non-ME being a metric.



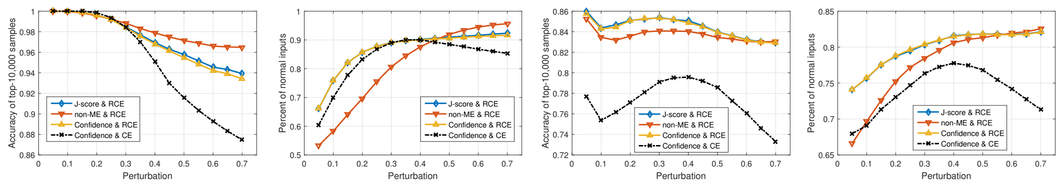
(a) CE (b) RCE

Figure 2: t-SNE visualization of final hidden vectors on 1,000 samples in the training set of MNIST, and the model is Resnet-32.

4.3 The ability of recognizing adversarial samples

To analyze the ability of a classifier to distinguish between adversarial samples and normal ones, we construct two mixed datasets—one for MNIST and one for CIFAR-10. Each mixed set takes the corresponding test set as the 10,000 normal samples, and then crafts 10,000 adversarial samples based on the normal ones by FGSM.

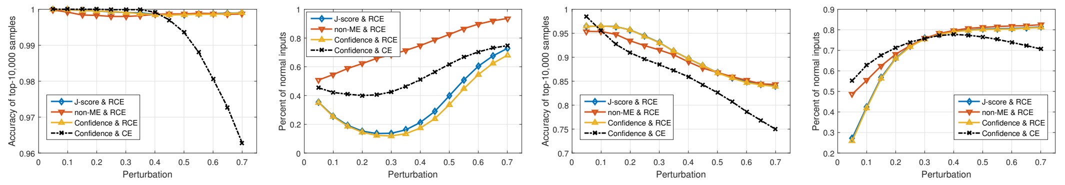
We feed all the 20,000 samples of a mixed set into the Resnet classifier that is learned on the corresponding training set, with Threshold-output strategy still disabled. Then we sort the samples in a descending order according to their values of a particular metric (i.e., confidence, non-ME or J-score). Fig. 3 shows the classification accuracy on the top-10,000 samples as well as the percentage of normal samples in the top-10,000 samples w.r.t the perturbation ε in FGSM. Here, we treat the Resnets trained by the CE procedure as the baselines, where the metric is the confidence.



(a) The accuracy of the top-10,000 samples (MNIST) (b) The percent of normal samples in the top-10,000 samples (MNIST) (c) The accuracy of the top-10,000 samples (CIFAR-10) (d) The percent of normal samples in the top-10,000 samples (CIFAR-10)

Figure 3: Recognizing adversarial samples in the mixed datasets. The model is Resnet-32, J in FGSM is the CE cost function.

It is worth noting that when we apply FGSM to craft adversarial samples in Fig. 3, the J in FGSM we used is the CE cost function. However, in FGSM the cost function J should be the objective function used to train the target classifier, and cross-entropy is not the objective function used in the RCE training procedure. Therefore we set J be the RCE cost function to evaluate the performance of our method under more specific attacks. Here we do not directly apply FGSM on the reverse 'liar' classifier $F_R(X, \theta_R^*)$, instead we apply FGSM on the 'liar' classifier $F(X, \theta_R^*)$ to craft adversarial samples, because a successful attack on reverse classifiers can also successfully attack original classifiers, and vice versa. For comparison, we also apply FGSM with the same cost function on classifiers trained via the CE procedure, and similarly FGSM acts on the reverse classifier $F_R(X, \theta^*)$. Results on are shown in Fig. 4.



(a) The accuracy of the top-10,000 samples (MNIST)
(b) The percent of normal samples in the top-10,000 samples (MNIST)
(c) The accuracy of the top-10,000 samples (CIFAR-10)
(d) The percent of normal samples in the top-10,000 samples (CIFAR-10)

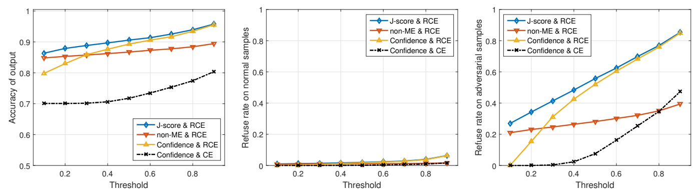
Figure 4: Recognizing adversarial samples in the mixed datasets. The model is Resnet-32, J in FGSM is the RCE cost function.

We can find that all three metrics combined with the RCE training procedure perform pretty well on both datasets. Here we simply set the relaxation factor η be 1, which leads to similar performance when using J-score and using confidence as the metric. The effect of different values of η is discussed later. Besides, we find that FGSM with CE cost function can better attack target classifiers than with RCE cost function, no matter which training procedure and metric the target classifier uses. On the MNIST dataset, non-ME works even better than confidence as the metric, which demonstrates that non-ME is a powerful and reasonable metric combined with the RCE training procedure. Note that in Fig. 4 the baselines have higher values on the percent of normal samples than using confidence or J-score with the RCE training procedure, the accuracy of the baselines is however much lower. The reason is that when the classifiers can still correctly classify most of the adversarial samples, a relatively low percent of normal samples is then rational and does not necessarily lead to a low accuracy, and an unsuccessful adversarial sample can actually be regarded as a normal sample.

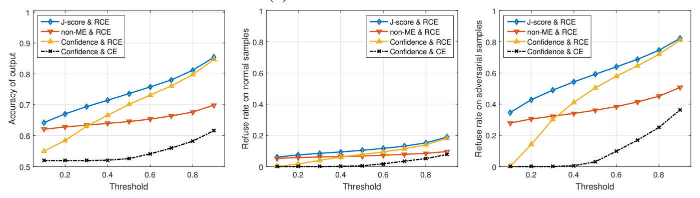
4.4 The performance of the entire algorithm

In order to evaluate the performance of our entire method, we activate the Threshold-output strategy in the prediction release phase and again feed the mixed datasets of 20,000 samples into the classifiers. For the purpose of comparison, we also impose the Threshold-output strategy in the prediction release phase on the baselines and other classifiers using different metrics and the RCE training procedure.

The results on MNIST and CIFAR-10 are shown in Fig. 5, where J in FGSM is set to be the CE cost function because of its better attacking ability. The 'Accuracy of output' refers to the accuracy of returned predictions on the total 20,000 samples. The 'Refuse rate' on normal or adversarial samples refers to the rate that a classifier refuses to predict and returns NOT SURE on the corresponding 10,000 samples.



(a) Results on MNIST



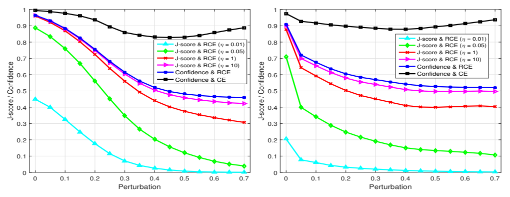
(b) Results on CIFAR-10

Figure 5: (*Left*) The accuracy of output labels. (*Center*) The refuse rate on the normal samples. (*Right*) The refuse rate on the adversarial samples. The model is Resnet-32, the perturbation $\varepsilon = 0.4$, the cost function in FGSM is the CE cost function.

The results show that our method is much more sensitive to adversarial samples by having high 'Refuse rate's on them while still has low 'Refuse rate's on normal samples. Especially on the MNIST dataset, FGSM can not effectively attack the reverse 'liar' networks trained by the RCE procedure. Compared to the MNIST dataset, the classification task on CIFAR-10 is much more difficult. However our method can still distinguish well between adversarial samples and normal samples by returning quite different 'Refuse rate's. It is worth noting that using non-ME has high 'Refuse rate's on adversarial samples even when the threshold T is small, but their values increase slowly as T increases. In contrast, using confidence has low 'Refuse rate's on adversarial samples when threshold T is small, however, their values increase obviously as T increases. So as a combination of non-ME and confidence, using J-score benefits from non-ME when T is small and from confidence when T is large, which makes J-score perform best on different values of T. In addition to FGSM, we believe our method will also perform well on defending other attacking algorithms, because as demonstrated in our theoretical analysis we improve the robustness generally, not specially for certain attacking algorithms.

4.5 Different values of relaxation factor η

In the above experiments, we simply set the relaxation factor η at 1. We now investigate its influence on J-score. In Fig. 6 we separately plot the mean values of J-score and confidence on the 10,000 adversarial samples crafted by FGSM based on the test sets of MNIST and CIFAR-10. Note that the value ranges of both J-score and confidence are $[0, 1]$, and when η is large the non-ME tends to be 1, which leads to identical values of J-score and confidence on all the inputs.



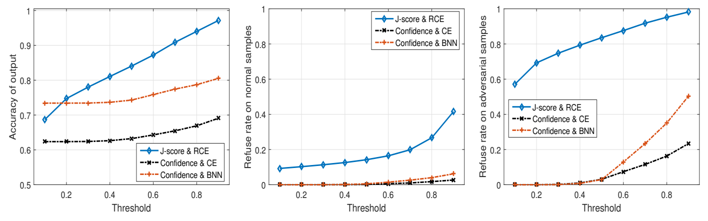
(a) Results on MNIST (b) Results on CIFAR-10

Figure 6: Mean values of J-score or confidence w.r.t the perturbation ε, the model is Resnet-32 and J in FGSM is the CE cost function.

From Fig. 6, we can verify two important points. First, the RCE training returns more reliable and reasonable confidence than the CE training, and J-score is indeed a more distinguishable metric than confidence in the adversarial setting. Second, there are certain medium values of η (e.g., $\eta = 0.05$ in both cases) work the best on distinguishability, in the sense of making the values of J-score decrease fastest as the perturbation increases. However, when $\eta = 0.05$ the values of J-score are undesirably low on normal samples in CIFAR-10, while our default choice of $\eta = 1$ has more balanced performance in both cases.

4.6 Comparison with Bayesian neural networks

Bayesian neural networks (BNNs) [18] are a family of models that may defend adversarial samples. Recent work [14] shows the ability of BNNs on defending adversarial samples. Here we quantitatively compare our method with BNNs. We access a fully connected network architecture with 2 hidden layers and 100 units in each layer, and apply the SGMCMC method [2] to train the BNN on MNIST. For comparison, we also apply the CE and the RCE procedure to train the network. We feed the 20,000 mixed samples based on MNIST to these trained networks, and impose the Threshold-output strategy on them. The results are shown in Fig. 7. We find that the trained BNN is not as vulnerable to FGSM as discriminative DNNs, partly because FGSM is designed for discriminative models. Our method performs even better than BNN on the overall accuracy and the refuse rate of adversarial samples. Moreover, our method is much simpler and computationally cheaper without the need to do expensive posterior inference as in BNN.



(a) The accuracy of output labels (b) The refuse rate on the normal samples (c) The refuse rate on the adversarial samples

Figure 7: Comparison with the BNN. The perturbation $\varepsilon = 0.4$, J in FGSM is the CE cost function.

5 Conclusions and Discussions

We present a promising method to improve the robustness by recognizing and filtering out adversarial samples, which performs well on different state-of-the-art DNN architectures, e.g., Resnets, full-connected networks and other DNN architectures (shown in Appendix). Besides, our method outperforms the BNN on improving the robustness in our experiments. Our method can be implemented using standard algorithms with little extra training cost. Because of the simplicity, our method is also easy to extend with other defense strategies such as adversarial training [26].

References

[1] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. *arXiv preprint arXiv:1608.04644*, 2016.

[2] Nan Ding, Youhan Fang, Ryan Babbush, Changyou Chen, Robert D Skeel, and Hartmut Neven. Bayesian sampling using stochastic gradient thermostats. In *Advances in neural information processing systems*, pages 3203–3211, 2014.

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

[4] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[5] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016.

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

[9] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[10] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

[12] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

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

[14] Yingzhen Li and Yarin Gal. Dropout inference in bayesian neural networks with alpha-divergences. *arXiv preprint arXiv:1703.02914*, 2017.

[15] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016.

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

[17] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.

[18] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[19] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015.

[20] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

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

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

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

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

[25] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[26] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

A Proof and supplementary experiments

A.1 Proof

Theorem 1. *Let x be a given input in the training dataset, y be the one-hot label vector, y_R be the reverse label vector corresponding to y, L be the number of different classes and θ_R^* is obtained from the reverse training. Under the L_∞-norm, if there is a training error $\alpha \ll \frac{1}{L}$ that*

$$\|\text{softmax}(Z(x, \theta_R^*)) - y_R\|_\infty \leq \alpha.$$

Then we have bounds

$$\|\text{softmax}(-Z(x, \theta_R^*)) - y\|_\infty \leq \alpha(L-1)^2$$

and $\forall j, k \neq t$

$$|\text{softmax}(-Z(x, \theta_R^*))_j - \text{softmax}(-Z(x, \theta_R^*))_k| \leq 2\alpha^2(L-1)^2,$$

where t is the index of true label.

Proof. For simplicity we omit the dependence of the logits Z on the input x and the parameters θ_R^*, and explicitly express the logits as $Z = (z_1, z_2, ..., z_L)$. Let $G = (g_1, g_2, ..., g_L) = (e^{z_1}, e^{z_2}, ..., e^{z_L})$ be the exponential logits, t be the label of x, then from the condition $\|\text{softmax}(Z) - y_R\|_\infty \leq \alpha$ we have

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

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

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

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

$$\begin{aligned} \text{softmax}(-Z)_t &= \frac{\frac{1}{g_t}}{\frac{1}{g_t} + \sum_{i\neq t}\frac{1}{g_i}} \\ &= \frac{1}{1 + \sum_{i\neq t}\frac{g_t}{g_i}} \\ &\geq \frac{1}{1 + \sum_{i\neq t}\frac{\alpha C}{(\frac{1}{L-1} - \alpha)C}} \\ &= \frac{1}{1 + \frac{\alpha(L-1)^2}{1 - \alpha(L-1)}} \\ &= 1 - \frac{\alpha(L-1)^2}{1 - \alpha(L-1) + \alpha(L-1)^2} \\ &\geq 1 - \alpha(L-1)^2 \end{aligned}$$

and $\forall j \neq t$,

$$\begin{aligned}\mathrm{softmax}(-Z)_j &= \frac{\frac{1}{g_j}}{\frac{1}{g_t}+\sum_{i\neq t}\frac{1}{g_i}}\\ &= \frac{\frac{g_t}{g_j}}{1+\frac{g_t}{g_j}+\sum_{i\neq t,j}\frac{g_t}{g_i}}\\ &\leq \frac{\frac{g_t}{g_j}}{1+\frac{g_t}{g_j}}\\ &= \frac{1}{1+\frac{g_j}{g_t}}\\ &\leq \frac{1}{1+\frac{(\frac{1}{L-1}-\alpha)C}{\alpha C}}\\ &= \alpha(L-1).\end{aligned}$$

Then we have proven that $\|\mathrm{softmax}(-Z)-y\|_\infty \leq \alpha(L-1)^2$. Furthermore, we have $\forall j,k\neq t$,

$$\begin{aligned}|\mathrm{softmax}(-Z)_j-\mathrm{softmax}(-Z)_k| &= \frac{\left|\frac{1}{g_j}-\frac{1}{g_k}\right|}{\frac{1}{g_t}+\sum_{i\neq t}\frac{1}{g_i}}\\ &\leq \frac{\frac{1}{(\frac{1}{L-1}-\alpha)C}-\frac{1}{(\frac{1}{L-1}+\alpha)C}}{\frac{1}{\alpha C}+\sum_{i\neq t}\frac{1}{(\frac{1}{L-1}+\alpha)C}}\\ &= \frac{\frac{L-1}{1-\alpha(L-1)}-\frac{L-1}{1+\alpha(L-1)}}{\frac{1}{\alpha}+\frac{(L-1)^2}{1+\alpha(L-1)}}\\ &= \frac{2\alpha^2(L-1)^2}{1+\alpha(L-1)^2(1-\alpha L)}\\ &\leq 2\alpha^2(L-1)^2.\end{aligned}$$

□

A.2 Experiments on simple CNN

In order to test our method on other convolutional neural networks (CNNs) [3], we implement two neural networks as described in Table 2 and the training details are included in Table 3. The architectures are based on the most basic CNN operations, and we use training tools like Adam [10], dropout [25] and batch normalization [9] in our training procedures.

Table 2: Overview of Architecture

Layer Type	MNIST	CIFAR-10
Relu Convolutional	32filters(3x3)	64filters(3x3)
Relu Convolutional	-	64filters(3x3)
Max Pooling	2x2	2x2
Relu Convolutional	64filters(3x3)	128filters(3x3)
Relu Convolutional	-	128filters(3x3)
Max Pooling	2x2	2x2
Relu Fully Connect.	200 units (sigmoid)	256 units
Relu Fully Connect.	200 units (sigmoid)	256 units
Softmax	10 units	10 units

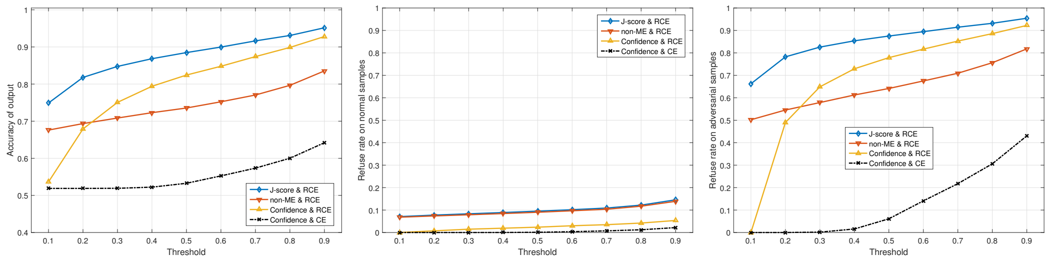
Table 3: Overview of Training Parameters

Parameter	MNIST	CIFAR-10
Learning Rate	0.001(Adam)	0.1(Momentum)
Dropout rate	0.5	0.5
Batch normalization	-	Yes
Batch size	100	100
Epochs	5	60

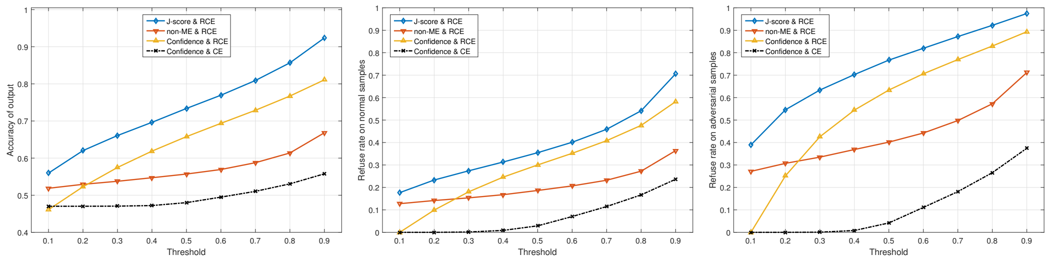
We separately apply the CE training procedure and the RCE training procedure on the MNIST and the CIFAR-10 datasets using the same architectures and training methods described above. The test accuracy is shown in Table 4, where the Threshold-output strategy is disabled and all points in the test set can receive their predicted labels. We use a weight decay of 0.001 and momentum of 0.9 to train the CIFAR-10 network. Besides, we standardize the data points in CIFAR-10 before feeding to classifiers.

Table 4: Overview of Test Accuracy

Training procedure	MNIST	CIFAR-10
CE	98.81%	85.62%
RCE	98.79%	85.68%



(a) Results on MNIST



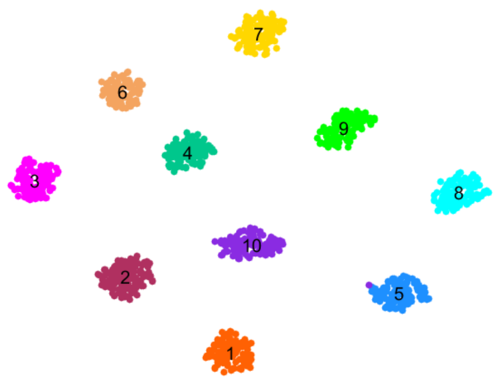
(b) Results on CIFAR-10

Figure 8: (*Left*) The accuracy of output labels. (*Center*) The refuse rate on the normal samples. (*Right*) The refuse rate on the adversarial samples. The perturbation $\varepsilon = 0.4$, J in FGSM is the CE cost function.

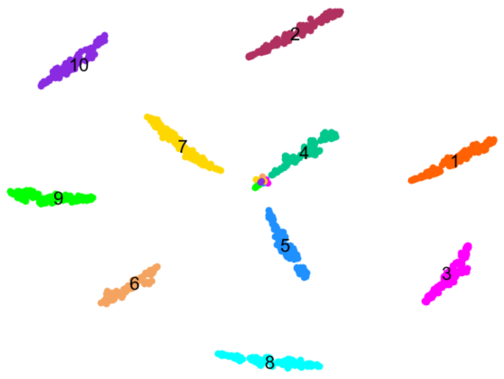
We separately feed the mixed datasets of MNIST and CIFAR-10 into the corresponding trained networks. Results are shown in Fig. 8. We can find that our method can also work well on simple CNN architectures. Note that in the experiments on CIFAR-10, the gradual increase of the refuse rate on normal samples is reasonable because the test accuracy of the classifiers on CIFAR-10 are only around 85%, not higher than 98% as on MNIST.

B t-SNE on the CIFAR-10

In Fig. 9 we show the results of t-SNE visualization on CIFAR-10.



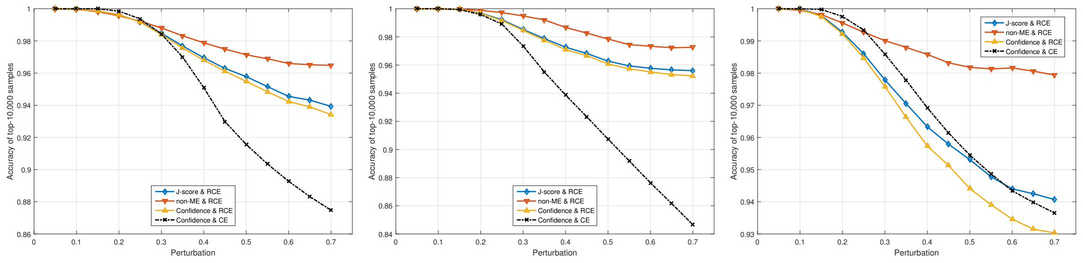
(a) CE



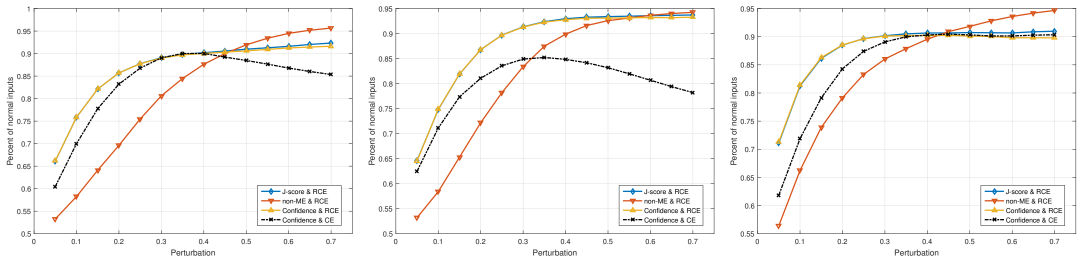
(b) RCE

Figure 9: t-SNE visualization of final hidden vectors on 1,000 samples in the training set of CIFAR-10, and the model is Resnet-32.

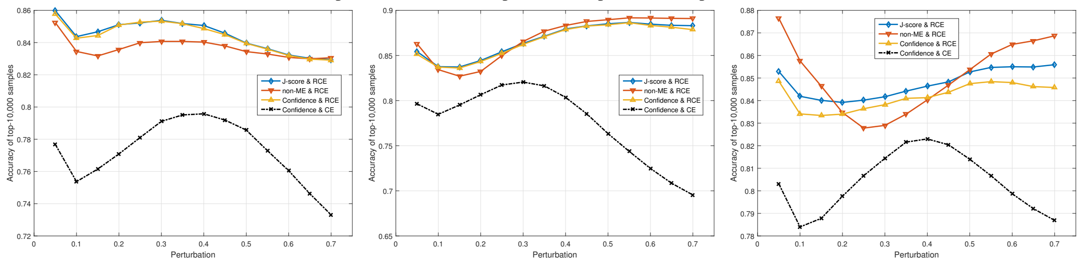
C Full results of Resnets



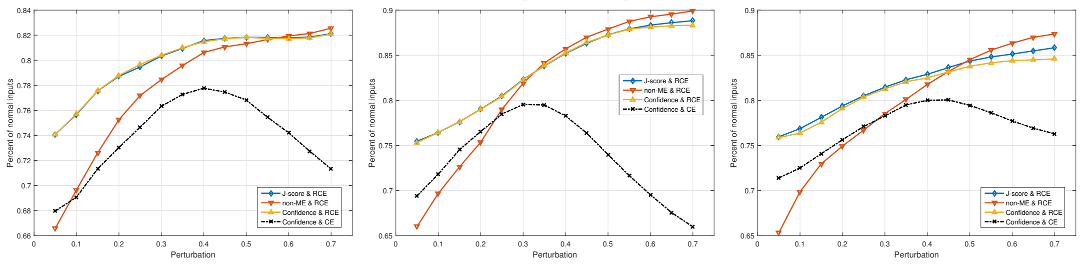
(a) The accuracy of the top-10,000 samples (MNIST)



(b) The percent of normal samples in the top-10,000 samples (MNIST)

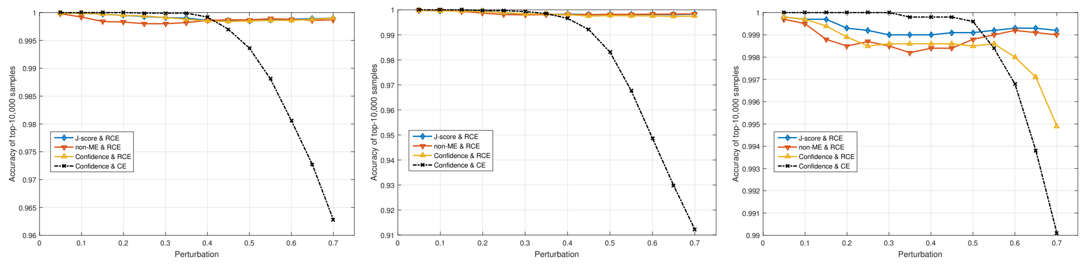


(c) The accuracy of the top-10,000 samples (CIFAR-10)

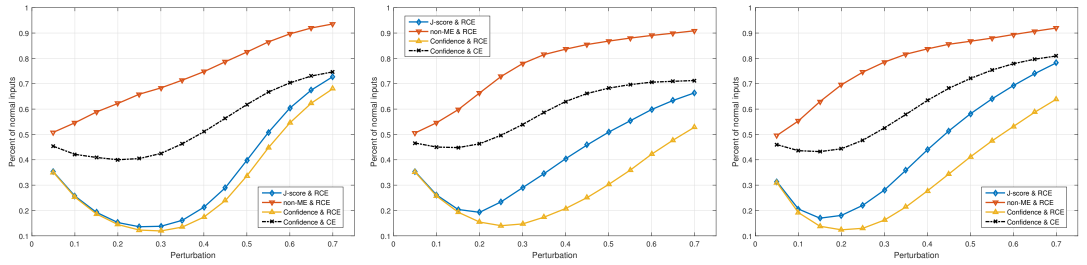


(d) The percent of normal samples in the top-10,000 samples (CIFAR-10)

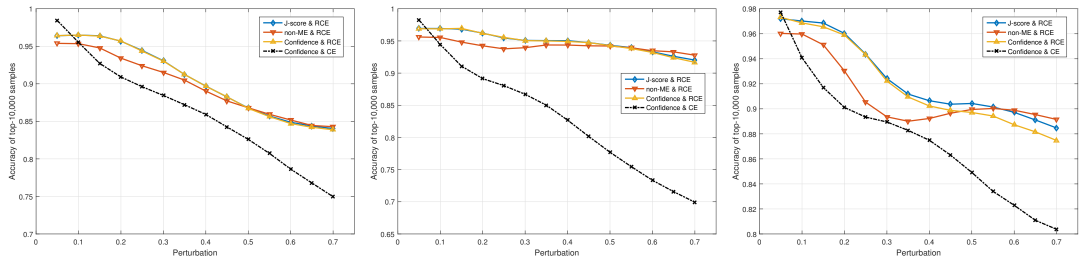
Figure 10: Recognizing adversarial samples in the mixed datasets. J in FGSM is the CE cost function. (*Left*) Resnet-32. (*Center*) Resnet-56. (*Right*) Resnet-110.



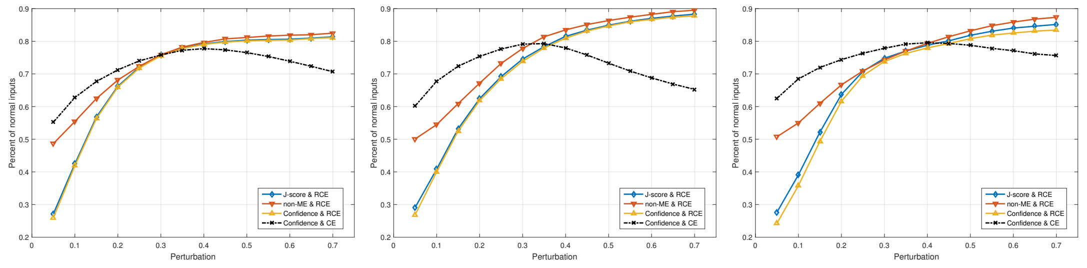
(a) The accuracy of the top-10,000 samples (MNIST)



(b) The percent of normal samples in the top-10,000 samples (MNIST)

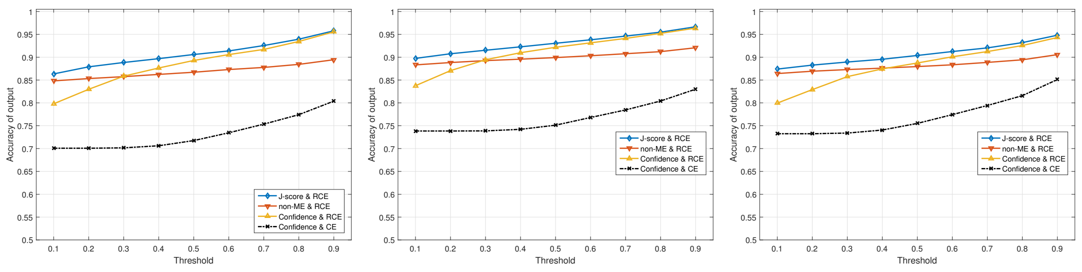


(c) The accuracy of the top-10,000 samples (CIFAR-10)

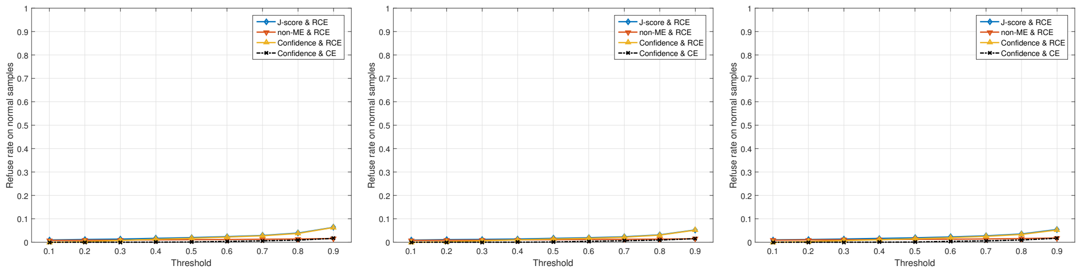


(d) The percent of normal samples in the top-10,000 samples (CIFAR-10)

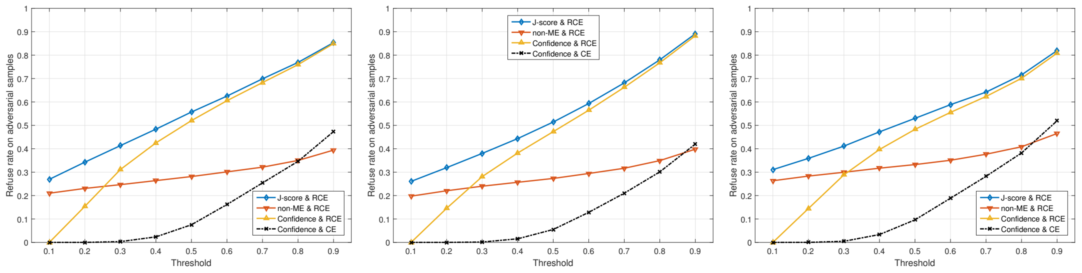
Figure 11: Recognizing adversarial samples in the mixed datasets. J in FGSM is the RCE cost function. (*Left*) Resnet-32. (*Center*) Resnet-56. (*Right*) Resnet-110.



(a) The accuracy of output labels

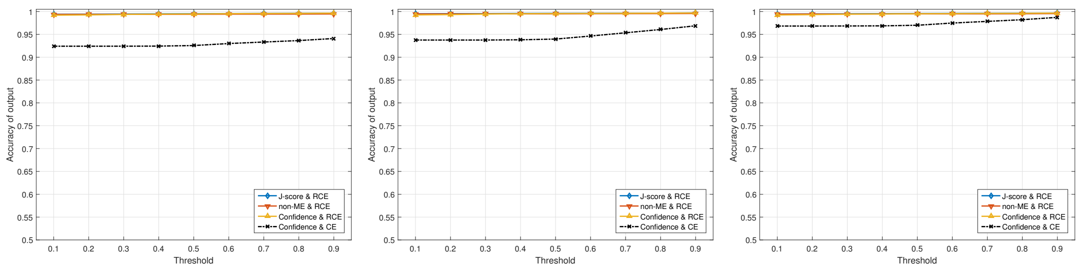


(b) The refuse rate on the normal samples

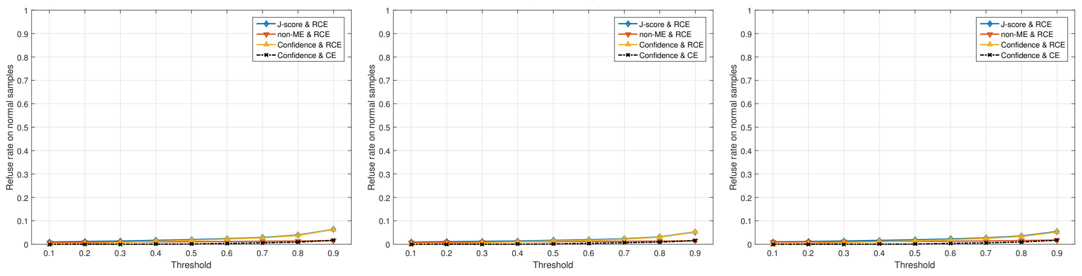


(c) The refuse rate on the adversarial samples

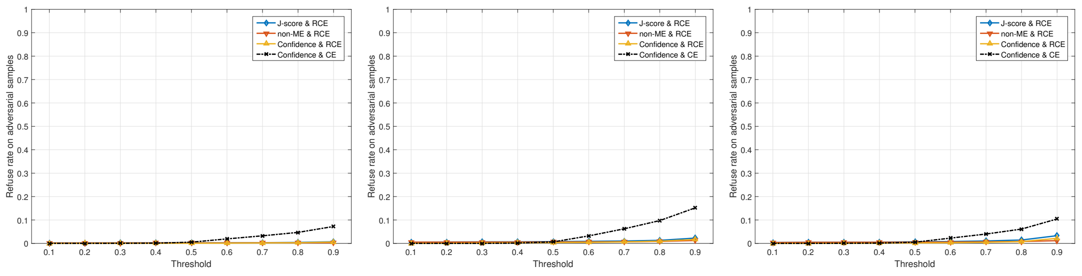
Figure 12: Results on MNIST, J in FGSM is the CE cost function. (*Left*) Resnet-32. (*Center*) Resnet-56. (*Right*) Resnet-110.



(a) The accuracy of output labels

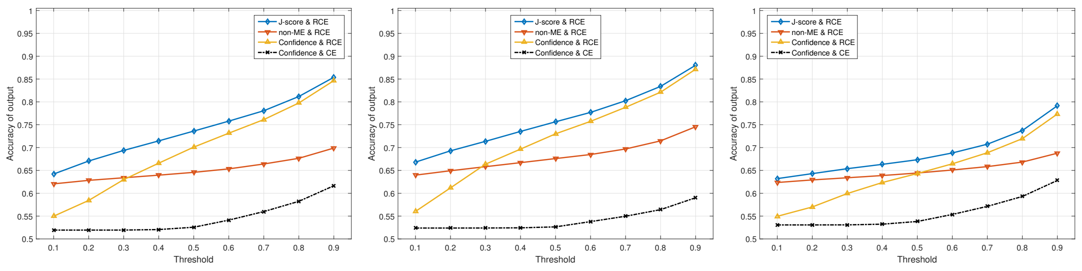


(b) The refuse rate on the normal samples

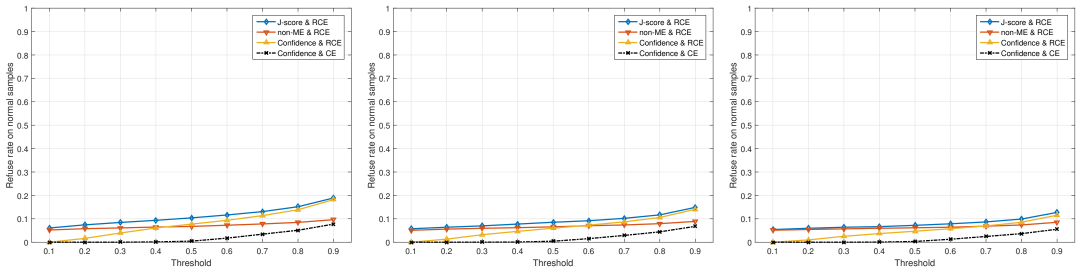


(c) The refuse rate on the adversarial samples

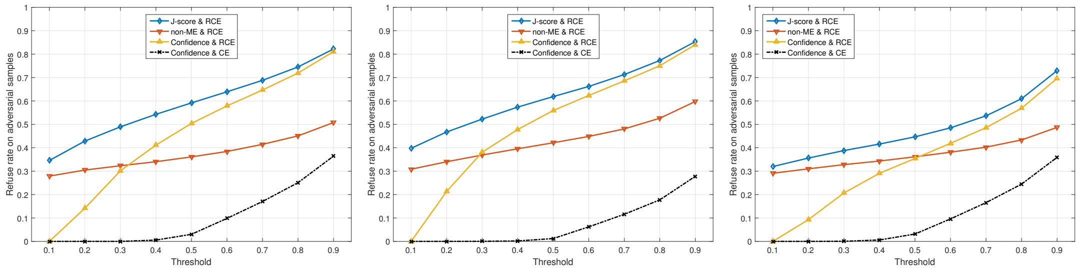
Figure 13: Results on MNIST, J in FGSM is the RCE cost function. (*Left*) Resnet-32. (*Center*) Resnet-56. (*Right*) Resnet-110.



(a) The accuracy of output labels

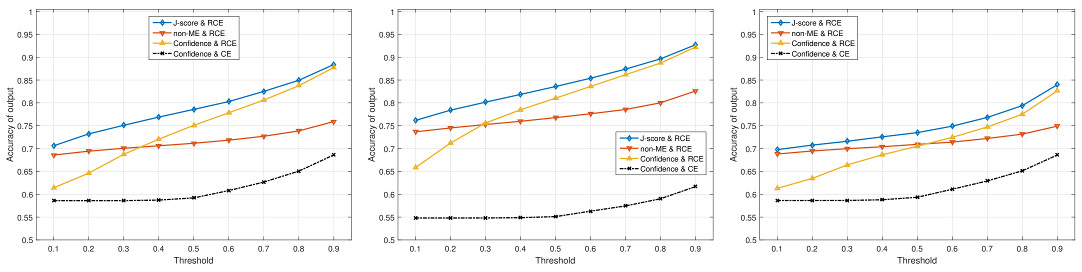


(b) The refuse rate on the normal samples

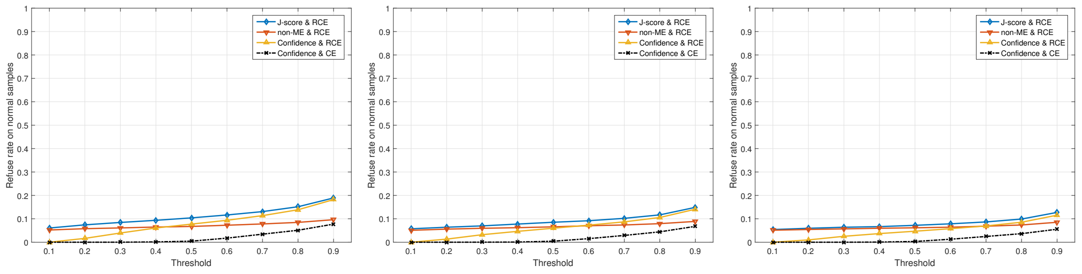


(c) The refuse rate on the adversarial samples

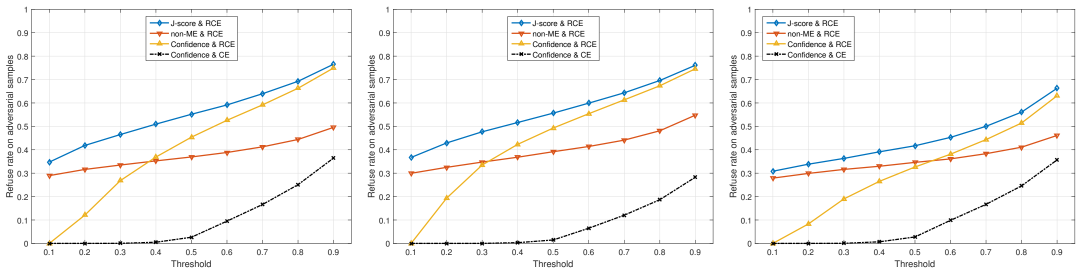
Figure 14: Results on CIFAR-10, J in FGSM is the CE cost function. (*Left*) Resnet-32. (*Center*) Resnet-56. (*Right*) Resnet-110.



(a) The accuracy of output labels



(b) The refuse rate on the normal samples



(c) The refuse rate on the adversarial samples

Figure 15: Results on CIFAR-10, J in FGSM is the RCE cost function. (*Left*) Resnet-32. (*Center*) Resnet-56. (*Right*) Resnet-110.