Classification score approach for detecting adversarial example in deep neural network

Hyun Kwon¹ · Yongchul Kim¹ · Hyunsoo Yoon² · Daeseon Choi³

Received: 17 April 2019 / Revised: 8 January 2020 / Accepted: 4 June 2020 /

Published online: 21 November 2020

© The Author(s) 2020

Abstract

Deep neural networks (DNNs) provide superior performance on machine learning tasks such as image recognition, speech recognition, pattern analysis, and intrusion detection. However, an adversarial example, created by adding a little noise to an original sample, can cause misclassification by a DNN. This is a serious threat to the DNN because the added noise is not detected by the human eye. For example, if an attacker modifies a right-turn sign so that it misleads to the left, autonomous vehicles with the DNN will incorrectly classify the modified sign as pointing to the left, but a person will correctly classify the modified sign as pointing to the right. Studies are under way to defend against such adversarial examples. The existing method of defense against adversarial examples requires an additional process such as changing the classifier or modifying input data. In this paper, we propose a new method for detecting adversarial examples that does not invoke any additional process. The proposed scheme can detect adversarial examples by using a pattern feature of the classification scores of adversarial examples. We used MNIST and CIFAR10 as experimental datasets and Tensorflow as a machine learning library. The experimental results show that the proposed method can detect adversarial examples with success rates: 99.05% and 99.9% for the untargeted and targeted cases in MNIST, respectively, and 94.7% and 95.8% for the untargeted and targeted cases in CIFAR10, respectively.

Keywords Deep neural network · Evasion attack · Adversarial example · Machine learning · Detection method · Classification score.

✉ Daeseon Choi
sunchoi@ssu.ac.kr

Hyun Kwon
hkwon.cs@gmail.com

¹ Department of Electrical Engineering, Korea Military Academy, 574 Hwarang-ro, Nowon-gu, Seoul 01819, Republic of Korea

² School of Computing, Korea Advanced Institute of Science and Technology (KAIST), 291 Daehak-ro, Yuseong-gu, Daejeon 34141, Republic of Korea

³ Department of Software, Soongsil University, 369 Sangdo-ro, Dongjak-gu, Seoul, Republic of Korea

1 Introduction

Deep neural networks (DNNs) [26] provide excellent service on machine learning tasks such as image recognition [28], speech recognition [10, 11], pattern analysis [4], and intrusion detection [24]. However, DNNs are vulnerable to adversarial examples [29, 33], in which a little noise has been added to an original sample. An adversarial example can cause misclassification by the DNN, but humans cannot detect a difference between the adversarial example and the original sample. For instance, if an attacker generates a modified left-turn sign so that it will be incorrectly categorized by a DNN, the autonomous vehicle with the DNN will incorrectly classify the modified sign as pointing to the right, but a person will correctly classify the modified sign as pointing to the left. Because such adversarial examples are serious threats to a DNN, several defenses against adversarial examples are being studied.

There are two main conventional methods of defending against adversarial examples: the classifier modification method [8, 22, 29] and the input data modification method [6, 27]. In these defense methods, a separate adversarial training process or a process for generating a separate module is required to change a classifier or modify input data. In addition, in changing the classifier or the input data, these defense methods may degrade the accuracy on the original samples. However, adversarial examples can be detected using patterns characteristic of the classification scores without the need to change a classifier or modify input data. To generate an adversarial example, the attacker adds a minimum amount of noise to the original sample until it will cause misclassification by the DNN. In the generated adversarial example, the classification score for the recognized wrong class will be slightly higher than the classification score for the original class. This is a specific pattern that is characteristic of classification scores of an adversarial example. In this paper, we propose a defense system for detecting adversarial examples that does not require modifications to the classifier or the input data, making use of this pattern feature of classification scores. The contribution of this paper is as follows.

- We propose a detection method using the classification score patterns characteristic of adversarial examples. The proposed detection method uses the existing classifier; there is no need for a module that changes the classifier or manipulates the input data. We systematically organize the framework and principle of the proposed scheme.
- We analyze the error rates of the proposed method on the original sample and the adversarial example. We also show the trade-off relationship between the adversarial example and the original sample at the threshold by providing an appropriate threshold criterion.
- Through experiments using the MNIST [14] dataset and CIFAR10 [12] dataset, we show the effectiveness of the proposed scheme. In addition, we provide the results of tests on a targeted adversarial example and an untargeted adversarial example to demonstrate the performance of the proposed method. The proposed method can also be used in combination with other detection methods as an ensemble method.

The remainder of this paper is structured as follows. In the next section, background concepts and related work are introduced. In Section 3, the problem definition is given. The proposed scheme is presented in Section 4. The experiment is described and evaluated in Sections 5 and 6. A discussion of the proposed scheme is provided in Section 7. Finally, we present our conclusions in Section 8.

2 Background and related work

The study of adversarial examples was introduced by Szegedy et al. [29] in 2014. The main goal in using an adversarial example is to induce a DNN to make a mistake. This is accomplished by adding a small amount of noise to an original image such that a human cannot tell the difference between the original image and the distorted version.

The basic method for generating adversarial examples is described in Section 2.1 Adversarial examples can be categorized in four ways: by target model information, type of target recognition, distance measure, and method of generation, as described in Sections 2.2, 2.3, 2.4, and 2.5, respectively. Section 2.6 describes methods of defense against adversarial examples.

2.1 Adversarial example generation

The basic architecture used for generating an adversarial example consists of two elements: a target model and a transformer. The transformer takes an original sample, x , and original class, y , as input data. The transformer then creates as output a transformed example $x^* = x + \delta$, with noise value δ added to the original sample x . The transformed example x^* is given as input to the target model. The target model then provides the transformer with the class probability results for the transformed example. The transformer updates the noise value δ in the transformed example $x^* = x + \delta$ so that the other class probabilities are higher than the original class probabilities while minimizing the distortion distance between x^* and x .

2.2 Categorization by target model information

Attacks that generate adversarial examples can be divided into two types according to the amount of information about the target that is required for the attack: white box attacks and black box attacks. It is a white box attack [2, 29] when the attacker has detailed information about the target model (i.e., model architecture, parameters, and class probabilities of the output); hence, attack success rates for white box attacks reach nearly 100%. A black box attack [16, 19] is when the attacker can query the target model but does not have the target model information. In this type, the attacker can obtain only the output for the input value. The method proposed in this paper assumes that the attacker does not have information about the threshold of the detector. It is a limited-knowledge attack in that the attacker knows about the target classifier but does not know the detection threshold in the last layer of the network.

2.3 Categorization by type of target recognition

Depending on the class the target model recognizes from the adversarial example, we can place an adversarial example [2, 20, 31] into one of two categories: a targeted adversarial example or an untargeted adversarial example. The first, the targeted adversarial example, is one that causes the target model to recognize the adversarial image as a particular intended class; it can be expressed mathematically as follows:

Given a target model and an original sample $x \in X$, the problem can be reduced to an optimization problem that generates a targeted adversarial example x^* :

$$x^* : \operatorname{argmin}_{x^*} L(x, x^*) \text{ such that } f(x^*) = y^*, \quad (1)$$

where $L(\cdot)$ is a measure of the distance between original sample x and transformed example x^* , y^* is a particular intended class, $\operatorname{argmin}_x F(x)$ is the x value at which the function $F(x)$ becomes minimal, and $f(\cdot)$ is an operation function that provides class results for the input values of the target model.

The untargeted adversarial example, on the other hand, is one that causes the target model to recognize the adversarial image as any class other than the original class; it can be expressed mathematically as follows:

Given a target model and an original sample $x \in X$, the problem can be reduced to an optimization problem that generates an untargeted adversarial example x^* :

$$x^* : \operatorname{argmin}_{x^*} L(x, x^*) \text{ such that } f(x^*) \neq y, \quad (2)$$

where $y \in Y$ is the original class.

The untargeted adversarial example has the advantages of less distortion from the original image and a shorter learning time compared to the targeted adversarial example. However, the targeted adversarial example is a more elaborate and powerful attack for controlling the perception of the attacker's chosen class.

2.4 Categorization by distance measure

There are three ways to measure the distortion between the original sample and the adversarial example [2]. The first distance measure, L_0 , represents the sum of the numbers of all changed pixels:

$$\sum_{i=0}^n |x_i - x_i^*|, \quad (3)$$

where x_i is the original i^{th} pixel and x_i^* is the i^{th} pixel in the adversarial example. The second distance measure, L_2 , represents the standard Euclidean norm, expressed as follows:

$$\sqrt{\sum_{i=0}^n (x_i - x_i^*)^2}. \quad (4)$$

The third distance measure, L_∞ , is the maximum distance value between x_i and x_i^* . The smaller the values of these three distance measures, the more similar the example image is to the original sample from a human perspective.

2.5 Categorization by method of generation

There are four typical methods for generating adversarial examples. The first method is the fast gradient sign method (FGSM) [8], which finds x^* through L_∞ :

$$x^* = x + \epsilon \cdot \operatorname{sign}(\nabla \operatorname{loss}_{F,t}(x)), \quad (5)$$

where F is an object function and t is a target class. In each iteration of the FGSM, the gradient is updated by ϵ from the original x , and x^* is found through optimization. This method is simple and demonstrates good performance.

The second method is iterative FGSM (I-FGSM) [13], which is an updated version of FGSM. Instead of the amount ϵ being changed in every step, a smaller amount, α , is updated and eventually clipped by the same ϵ value:

$$x_i^* = x_{i-1}^* - \text{clip}_\epsilon(\alpha \cdot \text{sign}(\nabla \text{loss}_{F,t}(x_{i-1}^*))). \quad (6)$$

I-FGSM provides better performance than FGSM.

The third is the Deepfool method [18], which is an untargeted attack and uses the L_2 distance measure. This method generates an adversarial example more efficiently than FGSM and one that is as close as possible to the original image. To generate an adversarial example, the method constructs a neural network and searches for x^* using the linearization approximation method. However, because the neural network is not completely linear, multiple iterations are needed to find the adversarial example; i.e., the process is more complicated than FGSM's.

The fourth method is the Carlini attack [2], which is the latest attack method and delivers better performance than FGSM and I-FGSM. This method can achieve a 100% success rate even against the distillation structure [22], which was recently introduced in the literature. The key principle of this method involves the use of a different objective function:

$$D(x, x^*) + c \cdot f(x^*). \quad (7)$$

Instead of using the conventional objective function $D(x, x^*)$, this method proposes a way to find an appropriate binary c value. In addition, it suggests a method for controlling the attack success rate even at the cost of some increase in distortion by incorporating a confidence value as follows:

$$f(x^*) = \max(Z(x^*)_t - \max\{Z(x^*)_i : i \neq t\}, -k), \quad (8)$$

where $Z(\cdot)$ represents the pre-softmax classification result vector and t is a target class. In this paper, we construct the model by applying the Carlini attack, which is the most powerful of the four methods, and use L_2 as the distance measure.

2.6 Defense of adversarial example

There are two well-known conventional methods for detecting adversarial examples: the classifier modification method [8, 22, 29] and the input data modification method [6, 27]. The first, the classifier modification method, can be further divided into two techniques: an adversarial training technique [8, 29] and a defensive distillation technique [22]. The adversarial training technique was introduced by Szegedy et al. [29] and Goodfellow et al. [8]; the main strategy of this technique is to impose an additional learning process for adversarial examples. A recent study by Tramèr et al. [30] proposed an ensemble adversarial training technique that uses a number of local models to increase the resistance against adversarial examples. However, the adversarial training technique has a high probability of reducing the classification accuracy of the original sample. The defensive distillation technique, proposed by Papernot et al. [22], resists adversarial example attacks by preventing the gradient descent calculation. To block the attack gradient, this technique uses two neural networks; the output class probability of the classifier is used as the input for the second stage of classifier training. However, this technique also requires a separate structural improvement and is vulnerable to white box attack.

The second conventional method, the input data modification method, can be further divided into three techniques: the filtering module [6, 27], feature squeezing [32], and Magnet techniques [17]. Shen et al. [27] proposed the filtering module technique, which

eliminates adversarial perturbation using generative adversarial nets [7]. This technique preserves classification accuracy for the original sample but requires an additional module process to create a filtering module. Xu et al. [32] proposed the feature squeezing technique, which modifies input samples. In this technique, the depth of each pixel in the image is reduced, and the difference between each pair of corresponding pixels is reduced by spatial smoothing. However, a separate processing procedure is required to alter the input data. Recently, an ensemble technique combining several defense techniques such as Magnet [17] has been introduced. Magnet, proposed by Meng and Chen [17], consists of a detector and a reformer to resist adversarial example attacks. In this technique, the detector detects the adversarial example by measuring distances far away, and the reformer changes the input data to the nearest original sample. However, the Magnet technique is vulnerable to white box attack and requires a separate process to create the detector and reformer modules.

Because they must change a classifier or modify input data, the above defense methods require an additional separate learning process for an adversarial example. Our proposed scheme, by contrast, is a new detection method that does not require an additional learning process but rather uses a pattern characteristic of the classification scores of adversarial examples.

3 Problem definition

The adversarial example is an optimization problem, that of finding a sample that satisfies two conditions that cause misclassification and has minimal distortion. Table 1 shows the classification scores for an adversarial example (“8” → “7”) and for the corresponding original sample (“8”). As seen in the table, the class “8” score for the original sample (14.1) is higher than any other classification score. However, the class “7” score for the adversarial example (8.417) is slightly higher than that for the original class, “8” (8.411). This is because the adversarial example generation process adds minimal noise to the original sample and stops adding noise when the wrong classification score becomes slightly higher than that for the original class.

Figure 1 shows the decision boundary between the adversarial example and the original sample. The decision boundary represents the line within which the model can correctly recognize the classification for the input. For example, in the figure, the original sample x is within the decision boundary, so the original sample x is correctly recognized as the original class. Because adversarial examples x^* are outside the decision boundary, however, they are mistakenly recognized as wrong classes. As the generation of an adversarial example is an optimization problem, the adversarial example should be placed in the area where the model will misrecognize it while minimizing the distance from the original sample. Therefore, the adversarial example should be located slightly outside the decision boundary line.

Table 1 Classification scores of an adversarial example: “8” → “7”

Description	Original sample (“8”)	Adversarial example (“7”)

Classification scores	[1.11 -0.57 1.92 -0.45 -2.78 -0.63 -2.23 -4.56 14.1 -2.38]	[0.79 -16.08 -2.53 -1.70 3.16 -17.42 -3.19 8.417 8.411 2.71

Fig. 1 Decision boundary of model D for original sample and adversarial examples

Thus, there is a pattern displayed by adversarial examples with regard to their classification scores and the decision boundary, as shown in Table 1 and Fig. 1. The original sample displays a pattern in which the classification score for the original class is higher than that of any other class. However, the adversarial example displays a pattern in which the classification score for the targeted wrong class is slightly higher than the classification score for the original class. Therefore, by comparing the differences between the largest number and the second-largest number in the classification scores, we can tell whether it is the original sample or the adversarial example. In this paper, we propose a detection method that works by analyzing the pattern of classification scores for an adversarial example.

4 Proposed methods

As Fig. 2 shows, the proposed method consists of two parts: calculating the classification score of each class for the input data and comparing a threshold with the score differences calculated from the calculated classification scores.

The first step is to calculate the classification scores for the input if the proposed method receives the original sample or the adversarial example as the input value. The classification

Fig. 2 Proposed architecture

score is related to the softmax layer, which is the last layer in the neural network. Most neural networks use the softmax function for the last layer:

$$\text{softmax}(l)_i = \frac{\exp(l_i)}{\sum_{k=0}^n \exp(l_k)} \quad (9)$$

The softmax result is the probability value for the mass function on the classes. The input to the softmax function is the vector l , called logit. Let $\text{rank}(l, i)$ be the index of the element with the highest i^{th} rank among all elements in l . Given a logit l of original sample x , the attacker creates an adversarial example x^* to satisfy the new logit l' , such as $\text{rank}(l, 1) \neq \text{rank}(l', 1)$. Let $g(\cdot)$ be the output of the last layer of the neural network for the input. As the characteristics of $g(x)$ for the original sample x and $g(x^*)$ for the adversarial example x^* are different, the classification score for each class associated with the softmax layer is calculated according to the input value. If the input value is the original sample x , $g(x)$ for the original class will be larger than $g(x)$ for the other classes. However, if the input value is adversarial example x^* , $g(x^*)$ for the targeted wrong class will be similar to the $g(x^*)$ value for the original class.

The second step is to calculate the score difference using the classification score for each extracted class and to detect adversarial examples using the threshold. We extract the highest value $\text{top}_{\max}$ and the second highest value $\text{sec}_{\max}$ among the classification scores. Then, we compute the classification score difference s_d from the extracted $\text{top}_{\max}$ and $\text{sec}_{\max}$ values:

$$s_d = \text{top}_{\max} - \text{sec}_{\max}. \quad (10)$$

If the calculated s_d value is smaller than a threshold T defined by the defender, it is classified as an adversarial example. If the value is larger than threshold T , it is classified as an original sample. The selection of an appropriate threshold value is described in Section 6 through an experimental method. The procedure for detecting adversarial examples is given in more detail in Algorithm 1.

Algorithm 1 Adversarial example detection algorithm.

Input: test image t_{image} , threshold T

Adversarial example detection:

```

logitarray  $\leftarrow g(t_{\text{image}})$ 
topmax  $\leftarrow \max(\text{logit}_{\text{array}})$ 
secmax  $\leftarrow \text{sec}_{\max}(\text{logit}_{\text{array}})$ 
sd  $\leftarrow \text{top}_{\max} - \text{sec}_{\max}$ 
if  $s_d \leq T$  then
    flag  $\leftarrow 1$ 
else
    flag  $\leftarrow 0$ 
end if
return flag

```

5 Experiment setup

Through experiments, we show that the proposed method can effectively detect adversarial examples. We used the Tensorflow library [1], widely used for machine learning, and a Xeon E5-2609 1.7-GHz server.

5.1 Datasets

MNIST [14] and CIFAR10 [12] were used as experimental datasets. MNIST is a standard dataset with handwritten images from 0 to 9. It consists of 60,000 training data and 10,000 test data. CIFAR10 is composed of color images of 10 classes of objects: plane, car, bird, cat, deer, dog, frog, horse, ship, truck. It consists of 50,000 training data and 10,000 test data.

5.2 Pretraining of classifier

The classifiers pretrained on MNIST and CIFAR10 are common CNNs [15] and VGG19 networks [9]. Their configuration and training parameters are shown in Tables 9, 10, and 11 in the Appendix. In the MNIST test, the pretrained classifier D correctly classified the original MNIST samples with over 99% accuracy. In the CIFAR10 test, the pretrained classifier D correctly classified the original CIFAR10 samples with over 91% accuracy.

5.3 Generation of adversarial example

The demonstration of the performance of the proposed method must take into account the fact that the proposed method assumes that the attacker does not have information about the threshold of the detector. Rather, it is a limited-knowledge attack, in which the attacker knows about the target classifier but does not know the detection threshold in the last layer of the network. Therefore, the adversarial example was created given an attacker that has all the information about the target classifier. The attacker generated the adversarial example using the Carlini method, which is a state-of-the-art method with a 100% attack success rate, using the L_2 distortion measure. For MNIST, the number of iterations was 500, the learning rate was 0.1, and the initial value was 0.01. For CIFAR10, the number of iterations was 10,000, the learning rate was 0.01, and the initial value was 0.01. For each dataset, we created 1000 randomly targeted adversarial examples and untargeted adversarial examples, respectively.

6 Experimental results

Experimental input consisted of images from the MNIST and CIFAR10 datasets. For each dataset, we show examples of dataset images for the original sample, the targeted adversarial example, and the untargeted adversarial example. With regard to detection, we analyzed the error rate of the original sample, the detection rate of the targeted adversarial example, and the detection rate of the untargeted adversarial example according to the threshold value. In addition, we analyzed the distortion and classification scores of the generated adversarial examples. By adjusting the threshold, we found the “sweet spot” detection rates and corresponding thresholds.

The experimental results show the error rate for each type of adversarial example as the metric of the detection performance of the proposed scheme. In the experiment, the threshold is the value applied as a judgment criterion: If the classification score difference (s_d) is less than threshold, the input is identified as an adversarial example, and if it is greater than the threshold, the input is determined to be an original sample. The error rate is defined as the rate at which the original sample or adversarial example is not detected correctly by the proposed method. For example, if 99 out of 100 adversarial examples are correctly detected,

the error rate is 1%. The distortion is defined as the square root of the sum of each pixel's difference from the corresponding pixel in the original sample (the L_2 distortion). We created an adversarial example using the Carlini method for the pretrained classifier as a white box attack. With regard to thresholds, we found sweet spots through experiments to find the appropriate thresholds for detecting adversarial examples by providing the generated adversarial examples and original samples to the classifier.

6.1 MNIST

This section compares the adversarial examples and original samples in MNIST and shows the distortion of the adversarial examples. The proposed method detects a given adversarial example, which is hard to discern by eye, at a detection rate that varies with the threshold. By adjusting the threshold, we find the detection rate corresponding to the sweet spot for MNIST. In terms of classification scores, we performed a comparative analysis between the original samples and the adversarial examples.

Table 4 shows the distortion of the targeted adversarial examples and untargeted adversarial examples for MNIST and CIFAR10 when the attack success rate was 100%. In a comparison of the datasets, MNIST shows less overall distortion than CIFAR10. In a comparison of the types of adversarial example, the untargeted adversarial example had less distortion than the targeted adversarial example.

Table 2 shows samples of original samples, untargeted adversarial examples, and targeted adversarial examples when the attack success rate was 100% for MNIST. For the table, the target class was selected by adding 1 to the original class, and the targeted adversarial example was generated to be misclassified as the target class chosen by the attacker. On the other hand, untargeted adversarial examples were generated to be misrecognized as any arbitrary class other than the original class. To human perception, however, both the targeted adversarial examples and the untargeted adversarial examples are similar to the original sample.

Figure 3a shows the error rates by threshold value for 1000 samples each of original samples, untargeted adversarial examples, and targeted adversarial examples on MNIST. As seen in the figure, as the threshold increases, the error rate for the original samples increases and the error rates for the targeted adversarial examples and untargeted adversarial examples decrease. In addition, the targeted adversarial examples had lower error rates than the untargeted adversarial examples. The reason is that the targeted adversarial example is a more sophisticated attack than the untargeted adversarial example, and a targeted adversarial example must have more distortion to be misclassified as a targeted class. As the trend is for the s_d value to decrease as the distortion increases, the s_d value for the targeted adversarial examples is less than the s_d value for the untargeted adversarial examples.

Table 5 shows the classification score difference for the original samples, untargeted adversarial examples, and targeted adversarial examples, using 1000 samples each from MNIST and CIFAR10. There is a numerical difference in the classification score differences between the adversarial example and the original sample. The average classification score difference value on MNIST for the original samples is more than 90 times greater than that for the untargeted adversarial examples and for the targeted adversarial examples. In particular, in the case of targeted adversarial examples on MNIST, it can be seen that the classification score difference is nearly 200 times greater than that for the original samples.

Table 6 shows the sweetspot for the untargeted adversarial examples and targeted adversarial examples in Fig. 3. This table shows that the proposed method can detect adversarial examples with detection rates: 99.05% and 99.9% for the untargeted and targeted cases

Table 2 For MNIST, samples of targeted and untargeted adversarial examples when the attack success rate was 100%

Original class	"0"	"1"	"2"	"3"	"4"	"5"	"6"	"7"	"8"	"9"
Orig.
Wrong class
Unt.
Target class
Tar.

These samples are images having the average distortion as given in Table 4: The target class was selected by adding 1 to the original class ; Orig. is the original sample ; Unt. is the untargeted adversarial example ; Tar. is the targeted adversarial example.

Fig. 3 Error rate of the proposed method on original samples, untargeted adversarial examples, and targeted adversarial examples for each threshold value

in MNIST, respectively, and 94.7% and 95.8% for the untargeted and targeted cases in CIFAR10, respectively. In particular, the proposed method has better performance in case of targeted adversarial example.

6.2 CIFAR10

This section compares the adversarial examples and original samples in CIFAR10 and shows the distortion of the adversarial examples. The proposed method detects a given adversarial example, which is hard to discern by eye, at a detection rate that varies with the threshold. By adjusting the threshold, we find the detection rate corresponding to the sweet spot for CIFAR10. In terms of classification scores, we performed a comparative analysis between the original samples and the adversarial examples.

For CIFAR10 as for MNIST, the targeted adversarial examples were more distorted than the untargeted adversarial examples (Table 4). However, as CIFAR10 is of three-dimensional color images, there is still no difference from the original samples in human recognition, as shown in Table 3.

Table 3 shows samples of original samples, untargeted adversarial examples, and targeted adversarial examples when the attack success rate was 100% for CIFAR10. As with MNIST, to human perception, targeted adversarial examples and untargeted adversarial examples are almost the same as the original samples (Table 4).

Fig. 3 (b) shows the error rates by threshold value for 1000 samples each of original samples, untargeted adversarial examples, and targeted adversarial examples on CIFAR10. As with MNIST, as the threshold increases, the error rate for the original samples increases and the error rates for the targeted adversarial examples and untargeted adversarial examples decrease. As with MNIST again, the error rates of the targeted adversarial examples are lower than the error rates of the untargeted adversarial examples. As seen in the figure, the overall error rates with CIFAR10 are higher than those with MNIST.

Like MNIST, CIFAR10 displays numerical differences in the classification score differences between the adversarial examples and the original samples. As seen in Table 5 for CIFAR10, the mean classification score difference for the original samples is more than five times greater than that for the untargeted adversarial examples and the targeted adversarial examples. In particular, the classification score difference value for targeted adversarial examples is almost 10 times greater than that for original samples.

Table 3 For CIFAR10, samples of targeted and untargeted adversarial examples when the attack success rate was 100%

Original class	“plane”	“car”	“bird”	“cat”	“deer”	“dog”	“frog”	“horse”
Orig.
Wrong class	“horse”	“ship”	“plane”	“dog”	“bird”	“frog”	“cat”	“deer”
Unt.
Target class	“car”	“bird”	“cat”	“deer”	“dog”	“frog”	“horse”	“ship”
Tar.

These samples are images having the average distortion as given in Table 4: The target class was selected by adding 1 to the original class ; Orig. is the original sample ; Unt. is the untargeted adversarial example ; Tar. is the targeted adversarial example.

Table 4 Distortion of targeted adversarial example and untargeted adversarial example when the attack success rate is 100%

Description	MNIST		CIFAR10	
	Untargeted	Targeted	Untargeted	Targeted
Mean distortion	1.03	1.94	27.41	53.42
SD distortion	0.6	0.9	26.44	32.39
Maximum distortion	3.35	6.127	135.62	187.85
Minimum distortion	0.000006	0.000006	0.004	0.004

“SD” is standard deviation

7 Discussion

Assumption The proposed method assumes that the attacker does not have information about the threshold of the detector. Rather, it is a limited-knowledge attack: The attacker knows about the target classifier but does not know the detection threshold in the last layer of the network. Under this assumption, the attacker creates an optimized adversarial example with minimal distortion through multiple iterations, which causes misclassification by the target classifier. Therefore, the adversarial example has a high attack success rate for the target classifier, with minimal distortion from the original sample (Table 6).

The proposed scheme requires a classifier that learns on original training data. This scheme has an advantage in that it can easily detect adversarial examples by using the classification score pattern from the last softmax layer of the general classifier. Therefore, the proposed method does not need to learn adversarial examples, use another processing module, or be equipped to perform input data filtering.

Applicability in ensemble methods The proposed method can be used in combination with other detection methods as an ensemble method. For example, an ensemble method is available that detects adversarial examples first by using class result differences caused by modules that can add noise to the input data; after that, secondary detection is performed using the classification score. Because the original sample is less influenced by small amounts of external noise, the class result of the original sample is not affected. In the case of the adversarial example, however, the class result is changed by the addition of a small amount of noise, because it is near the decision boundary of the classifier. As shown in Fig. 4, the first detection method detects an adversarial example by comparing a class result that passes the modulation with a class result that does not pass the modulation. After

Table 5 Classification score difference (s_d) for original samples, untargeted adversarial examples, and targeted adversarial examples using 1000 of each

Description	MNIST			CIFAR10		
	Original	Untargeted	Targeted	Original	Untargeted	Targeted
Average	10.641	0.0119	0.0068	1.623	0.131	0.082
SD	5.937	0.0823	0.032	0.811	0.748	0.54
Maximum	33.529	0.99	0.99	14.23	5.621	4.941
Minimum	0.013	0.0000004	0.0000003	0.0082	0.0000007	0.0000005

“SD” is standard deviation.

Table 6 Sweetspot of targeted adversarial example and untargeted adversarial example in Fig. 3

Description	MNIST		CIFAR10	
	Untargeted	Targeted	Untargeted	Targeted
Threshold	0.25	0.10	0.20	0.15
Detection rate (%)	99.05	99.9	94.7	95.8
Error rate of orig. (%)	0.9	0.5	6.0	2.5

Orig. is the original sample.

the first detection, the secondary detection method can improve performance by detecting the adversarial example using the classification score.

In the experiment for the ensemble method, the modification was from a module that provides noise by random normalization with a noise coefficient of 0.05. We applied the ensemble method on 1000 adversarial examples generated by white box attacks against the pretrained classifier.

To obtain the experimental results, we analyzed the classification score, class result, detection rate, and error rate for the ensemble method on the original sample in MNIST. Table 7 shows the class results and classification scores of an adversarial example (“7” → “4”) and original sample (“7”) before and after modification. The adversarial example shown in Table 7 was misclassified as 4, but the adversarial example with some noise added was correctly recognized as 7 because of the large change in the score of the original class. On the other hand, the original sample with some noise added was correctly recognized as 7, because the effect of noise on the class score is small. Thus, the first detection detects an adversarial example by using a difference due to random noise that affects a class result.

Table 8 shows the sweet spot for targeted adversarial examples and untargeted adversarial examples when the noise coefficient is 0.05, the threshold for untargeted attacks is 0.25, and the threshold for targeted attacks is 0.10. Experimentally, when the noise coefficient was 0.05, the rate of detection of the adversarial example was high without the error rate of the original sample. As shown in Table 8, the first detection method detects an adversarial example at a detection rate of about 65%. After the first detection method, the secondary detection method detects the adversarial example through the classification scores for the remaining input values. In the case of MNIST, the second detection method showed a 99.5% detection rate for untargeted attacks and a 99.9% detection rate for targeted attacks. Thus, it

Fig. 4 Ensemble method using proposed method

Table 7 Comparison on class result and classification scores of an adversarial example: “7” → “4” before and after the modification

Description	Original sample (“7”)	Adversarial example (“4”)

Classification scores	[-14.8 -6.24 -0.78 -7.15 10.1 -5.06 -12.6 19.6 -1.61 3.33]	[-15.4 -6.1 -2.15 -8.12 13.939 -3.4 -10.6 13.936 -1.28 2.54]
	Original sample with some noise (“7”)	Adversarial example with some noise (“7”)

Classification scores	[-14.9 -6.33 -0.42 -7.23 9.94 -5.82 -12.3 19.2 -0.69 3.23]	[-15.6 -5.67 -2.82 -8.27 13.95 -3.11 -10.9 14.21 -1.61 2.79]

is demonstrated that the proposed method can be used in combination with other detection methods as an ensemble method.

Detection considerations Regarding the establishment of the threshold value, we analyze the error rates for original samples, targeted adversarial examples, and untargeted adversarial examples. If the defender needs to keep the error rate for the original samples small, it is necessary to set the threshold as low as 0.05. However, as the error rates for the original samples and for the adversarial examples are in a trade-off relationship with the threshold, it is necessary to set a threshold value appropriate to the situation.

Regarding the type of adversarial example recognition, we analyze the error rates of the proposed method for targeted adversarial examples and untargeted adversarial examples. This analysis provides the information that the detection performance will change according to the type of adversarial example (i.e., targeted or untargeted). Untargeted adversarial examples have a higher error rate than targeted adversarial examples because they have less distortion from the original samples. However, on MNIST and CIFAR10, the detection rate for untargeted adversarial examples could be kept over 98.63% and 89.5%, respectively.

Regarding datasets, the performance with MNIST is better than that with CIFAR10. This is because the accuracy of the classifier for the original sample is high (99%) for MNIST but is relatively low (91%) for CIFAR10. If the accuracy of the classifier is low, the level of detection will be reduced proportionately. Therefore, it is important for the defender to select a classifier that has high accuracy to properly detect adversarial examples.

Regarding ambiguity between two classes (top_{max} and sec_{max}), if the difference between the highest value top_{max} and the second highest value sec_{max} is ambiguous, the original

Table 8 Sweet spot of targeted adversarial example and untargeted adversarial example when noise coefficient is 0.05, threshold for untargeted attack is 0.25, and threshold for targeted attack is 0.10

Description	1st detection		2nd detection	
	Untargeted	Targeted	Untargeted	Targeted
Detection rate	63 %	64.5 %	99.5%	99.9%

sample is likely to be recognized as an adversarial example. However, it is rare for there to be an original sample that has ambiguity between these two classes (top_{max} and sec_{max}). As shown in Table 5, for MNIST, the average classification score for the original sample was 10.641, and the average classification scores for the untargeted and targeted adversarial examples were 0.0119 and 0.0068, respectively. For CIFAR10, the average classification score for the original sample was 1.623, and the average classification scores for the untargeted and targeted adversarial examples were 0.131 and 0.082, respectively. Therefore, according to this table, an original sample is less likely to be misdetected as an adversarial example, because it is much less likely for an original sample to have ambiguity between the two classes (top_{max} and sec_{max}) than for an adversarial example to have such an ambiguity.

Dataset The error rate of the proposed method differs with the type of dataset. In terms of error rate performance, MNIST is better than CIFAR10. The distortion of the adversarial example also differs with the dataset. This is because of differences in the pixel count and the dimensionality of the datasets. For example, MNIST contains one-dimensional monochrome images, and CIFAR10 contains three-dimensional color images. In terms of the number of pixels, MNIST images each have 784 (1, 28, 28), whereas CIFAR10 images have 3072 (3, 32, 32). Therefore, CIFAR10 images will have more distortion than MNIST images. However, in terms of human perception, because CIFAR10 images are color images, adversarial examples for CIFAR10 are more similar to their original samples than are adversarial examples for MNIST.

Applications The proposed scheme can be applied in the field of autonomous vehicles and in military environments. If an attacker maliciously creates an adversarial example on a road sign, the proposed method can detect the adversarial example in advance and take action to warn the driver. The proposed method has the advantage that adversarial examples can be detected with high probability using the existing classifier without requiring a separate device. In addition, the proposed method can be applied as a detection method when an object is identified by a UAV or a drone in a military situation.

Limitation If the proposed method assumes a perfect-knowledge attack in which the detector threshold is known, an adversarial example can be generated beyond that threshold. Because the Carlini method [2] increases the confidence value when the attacker has all the information about the detection method, the attack success rate can be increased by adding more distortion in the range above the threshold. As mentioned by Carlini and Wagner [3], the Carlini method, which outperforms FGSM [8] and the JSMA [21] method, has a nearly 100% success rate in white box attacks, in which all the detector information is known. However, if the confidence value in the Carlini method increases to exceed the threshold, greater distortion occurs, and an adversarial example with substantial distortion can be detected by human perception, reconstruction error, and probability divergence methods.

In addition, in the case of an untargeted attack, the attacker can generate an untargeted adversarial example having a classification from the wrong class with high confidence. However, if the confidence value for a wrong class in the untargeted adversarial example increases, greater distortion occurs, and an adversarial example with substantial distortion can be detected by human perception, reconstruction error, and probability divergence methods.

8 Conclusion

The scheme proposed in this paper is a defense system that detects adversarial examples using an existing classifier. The proposed method is designed to detect adversarial examples using patterns characteristic of adversarial examples without needing to learn adversarial examples, use other processing modules, or be equipped to perform input data filtering. The

method detects adversarial examples while maintaining the accuracy of the original samples and without modifying the classifier. The proposed method was tested with MNIST and CIFAR10 for both untargeted adversarial examples and targeted adversarial examples and was analyzed with respect to error rates and threshold values. The experimental results show that the proposed method can detect adversarial examples with detection rates: 99.05% and 99.9% for the untargeted and targeted cases in MNIST, respectively, and 94.7% and 95.8% for the untargeted and targeted cases in CIFAR10, respectively.

Future work will extend the method to other image datasets such as ImageNet [5] and to the face domain [25]. In terms of datasets, the proposed method can be extended to face domains such as VGG-Face [23] or to ImageNet, which has 1000 image classes. Another challenge will be to design a detection method based on correlations among various classifiers. For this, we can investigate detection methods that improve performance by combining multiple softmax thresholds on multiple classifiers instead of using a single classifier.

Acknowledgments This work was supported by the Hwarang-Dae Research Institute of Korea Military Academy and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No.2020R1A2C1014813).

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Appendix

Table 9 Target classifier architecture for MNIST

Layer type	MNIST shape
Convolution+ReLU	[3, 3, 32]
Convolution+ReLU	[3, 3, 32]
Max pooling	[2, 2]
Convolution+ReLU	[3, 3, 64]
Convolution+ReLU	[3, 3, 64]
Max pooling	[2, 2]
Fully connected+ReLU	[200]
Fully connected+ReLU	[200]
Softmax	[10]

Table 10 Target classifier architecture [9] for CIFAR10

Layer type	CIFAR10 shape
Convolution+ReLU	[3, 3, 64]
Convolution+ReLU	[3, 3, 64]
Max pooling	[2, 2]
Convolution+ReLU	[3, 3, 128]
Convolution+ReLU	[3, 3, 128]
Max pooling	[2, 2]
Convolution+ReLU	[3, 3, 256]
Convolution+ReLU	[3, 3, 256]
Convolution+ReLU	[3, 3, 256]
Convolution+ReLU	[3, 3, 256]

Table 10 (continued)

Layer type	CIFAR10 shape
Max pooling	[2, 2]
Convolution+ReLU	[3, 3, 512]
Convolution+ReLU	[3, 3, 512]
Convolution+ReLU	[3, 3, 512]
Convolution+ReLU	[3, 3, 512]
Max pooling	[2, 2]
Convolution+ReLU	[3, 3, 512]
Convolution+ReLU	[3, 3, 512]
Convolution+ReLU	[3, 3, 512]
Convolution+ReLU	[3, 3, 512]
Max pooling	[2, 2]
Fully connected+ReLU	[4096]
Fully connected+ReLU	[4096]
Softmax	[10]

Table 11 Target classifier parameters

Parameter	MNIST classifier	CIFAR10 classifier
Learning rate	0.1	0.1
Momentum	0.9	0.9
Delay rate	-	10 (decay 0.0001)
Dropout	0.5	0.5
Batch size	128	128
Epochs	50	200

References

1. Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M et al (2016) Tensorflow: A system for large-scale machine learning in OSDI, vol 16
2. Carlini N, Wagner D (2017) Towards evaluating the robustness of neural networks. In: Security and Privacy SP, 2017 IEEE Symposium on, pp. 39–57 IEEE
3. Carlini N, Wagner D (2017). arXiv:1705.07263
4. Collobert R, Weston J (2008) A unified architecture for natural language processing: Deep neural networks with multitask learning. In: Proceedings of the 25th international conference on Machine learning pp. 160–167 ACM
5. Deng J, Dong W, Socher R, Li L.-J., Li K, Fei-Fei L (2009) Imagenet: A large-scale hierarchical image database. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pp. 248–255 IEEE
6. Fawzi A, Fawzi O, Frossard P (2015) Analysis of classifiers' robustness to adversarial perturbations. Mach Learn 107:1–28
7. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Advances in neural information processing systems, pp 2672–2680
8. Goodfellow I, Shlens J, Szegedy C (2015) Explaining and harnessing adversarial examples. In: International Conference on Learning Representations
9. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Inproceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
10. Hinton G, Deng L, Yu D, Dahl GE, Mohamed A-R, Jaitly N, Senior A, Vanhoucke V, Nguyen P, Sainath TN et al (2012) Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Proc Mag 29(6):82–97

11. Kereliuk C, Sturm BL, Larsen J (2015) Deep learning and music adversaries. *IEEE Transactions on Multimedia* 17(11):2059–2071
12. Krizhevsky A, Nair V, Hinton G (2014) The cifar-10 dataset. online: <http://www.cs.toronto.edu/kriz/cifar.html>
13. Kurakin A, Goodfellow I, Bengio S (2017) Adversarial examples in the physical world. *ICLR Workshop*
14. LeCun Y, Cortes C, Burges CJ (2010) Mnist handwritten digit database, AT&T Labs [Online]. Available: <http://yannlecun.com/exdb/mnist>, vol 2
15. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. *Proc IEEE* 86(11):2278–2324
16. Liu Y, Chen X, Liu C, Song D (2017) Delving into transferable adversarial examples and black-box attacks
17. Meng D, Chen H (2017) Magnet: a two-pronged defense against adversarial examples. In: *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 135–147 ACM
18. Moosavi-Dezfooli S-M., Fawzi A, Frossard P (2016). In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp 2574–2582
19. Narodytska N, Kasiviswanathan S (2017) Simple black-box adversarial attacks on deep neural networks. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops CVPRW*, pp. 1310–1318 IEEE
20. Oliveira GL, Valada A, Bollen C, Burgard W, Brox T (2016) Deep learning for human part discovery in images. In: *Robotics and Automation ICRA, 2016 IEEE International Conference on*, pp. 1634–1641 IEEE
21. Papernot N, McDaniel P, Jha S, Fredrikson M, Celik ZB, Swami A (2016) The limitations of deep learning in adversarial settings. In: *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pp. 372–387 IEEE
22. Papernot N, McDaniel P, Wu X, Jha S, Swami A (2016) Distillation as a defense to adversarial perturbations against deep neural networks
23. Parkhi OM, Vedaldi A, Zisserman A et al (2015) Deep face recognition in bmvc, vol 1
24. Potluri S, Diedrich C (2016) Accelerated deep neural networks for enhanced intrusion detection system. In: *Emerging Technologies and Factory Automation ETFA, 2016 IEEE 21st International Conference on*, pp. 1–8 IEEE
25. Rozsa A, Günther M., Rudd EM, Boulton TE (2017) Facial attributes: Accuracy and adversarial robustness, *Pattern Recognition Letters*
26. Schmidhuber J (2015) Deep learning in neural networks: an overview. *Neural networks* 61:85–117
27. Shen S, Jin G, Gao K, Zhang Y (2017) Ape-gan: Adversarial perturbation elimination with gan ICLR Submission available on OpenReview
28. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: *International Conference on Learning Representations*
29. Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I, Fergus R (2014) Intriguing properties of neural networks. In: *International Conference on Learning Representations*
30. Tramèr F., Kurakin A, Papernot N, Goodfellow I, Boneh D, McDaniel P (2018) Ensemble adversarial training: Attacks and defenses. In: *International Conference on Learning Representations ICLR*
31. Tramèr F., Papernot N, Goodfellow I, Boneh D, McDaniel P (2017) The space of transferable adversarial examples. *arXiv:1704.03453*
32. Xu W, Evans D, Qi Y (2018) Feature squeezing: Detecting adversarial examples in deep neural networks
33. Zhang F, Chan PP, Biggio B, Yeung DS, Roli F (2016) Adversarial feature selection against evasion attacks. *IEEE transactions on cybernetics* 46(3):766–777

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Hyun Kwon received the B.S degree in mathematics from Korea Military Academy, South Korea, in 2010. He also received the M.S. degree in School of Computing from Korea Advanced Institute of Science and Technology (KAIST) in 2015, and the Ph.D. degree at School of Computing, KAIST in 2020. He is currently an assistant professor in Korea Military Academy. His research interests include information security, machine learning, computer security, and intrusion tolerant system.

Yongchul Kim received the B.E. degree in electrical engineering from the Korea Military Academy, South Korea, in 1998, the M.S. degree in electrical engineering from the University of Surrey, U.K., in 2001, and the Ph.D. degree in electrical and computer engineering with the department of electrical and computer engineering, North Carolina State University, in 2007. He has been a professor in the department of electrical engineering at the Korea Military Academy. His research interests include WiMAX and wireless relay networks.

Hyunsoo Yoon received the B.E. degree in electronics engineering from Seoul National University, South Korea, in 1979, the M.S. degree in computer science from Korea Advanced Institute of Science and Technology (KAIST) in 1981, and the Ph.D. degree in computer and information science from the Ohio State University, Columbus, Ohio, in 1988. [Author Biographies Click here to access/download;Author Biographies;Biography.pdf](#) From 1988 to 1989, he was a member of technical staff at AT&T Bell Labs. Since 1989 he has been a faculty member of School of Computing at KAIST. His main research interest includes wireless sensor networks, 4G networks, and network security.

Daeseon Choi received the B.S. degree in computer science from Dongguk University, South Korea, in 1995, the M.S. degree in computer science from Pohang Institute of Science and Technology (POSTECH), South Korea, in 1997, and the Ph.D. degree in computer science from the Korea Advanced Institute of Science and Technology (KAIST), South Korea, in 2009. He is currently a professor at department of software, Soongsil University, South Korea. His research interests include information security and identity management.