

Article

Adversarial Detection Based on Inner-Class Adjusted Cosine Similarity [†]

Dejian Guan and Wentao Zhao *

College of Computer, National University of Defense Technology, Changsha 410000, China

* Correspondence: wtzhao@nudt.edu.cn

[†] This paper is an extended version of our paper published in IEEE, Dejian Guan, Dan Liu, Wentao Zhao.

Adversarial Detection based on Local Cosine Similarity. 2022 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA), Dalian, China, 24–26 June 2022.

Abstract: Deep neural networks (DNNs) have attracted extensive attention because of their excellent performance in many areas; however, DNNs are vulnerable to adversarial examples. In this paper, we propose a similarity metric called inner-class adjusted cosine similarity (IACS) and apply it to detect adversarial examples. Motivated by the fast gradient sign method (FGSM), we propose to utilize an adjusted cosine similarity which takes both the feature angle and scale information into consideration and therefore is able to effectively discriminate subtle differences. Given the predicted label, the proposed IACS is measured between the features of the test sample and those of the normal samples with the same label. Unlike other detection methods, we can extend our method to extract disentangled features with different deep network models but are not limited to the target model (the adversarial attack model). Furthermore, the proposed method is able to detect adversarial examples crossing attacks, that is, a detector learned with one type of attack can effectively detect other types. Extensive experimental results show that the proposed IACS features can well distinguish adversarial examples and normal examples and achieve state-of-the-art performance.

**Citation:** Guan, D.; Zhao, W.

Adversarial Detection Based on

Inner-Class Adjusted Cosine

Similarity. *Appl. Sci.* **2022**, *12*, 9406.<https://doi.org/10.3390/app12199406>**Keywords:** adversarial detection; inner-class adjusted cosine similarity; adversarial examples; deep learning

Academic Editors: Giuseppe D'Aniello, Alessandro Micarelli and Giuseppe Sansonetti

Received: 17 August 2022

Accepted: 16 September 2022

Published: 20 September 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In recent years, deep neural networks (DNNs) have attracted extensive attention and provided excellent performance in many fields. However, researchers discovered that DNNs were vulnerable to adversarial examples [1,2]. Szegedy et al. [1] first demonstrated that by adding human imperceptible perturbations on normal examples, adversaries could confuse the judgment of DNNs. This property of DNNs significantly hinders their application in security-critical areas.

There are works trying to explain the reason why there are adversarial examples in DNNs. Szegedy et al. [1] offered a simple explanation that the set of adversarial examples was of extremely low probability, and never or barely appeared in the training and test set. Later, Goodfellow et al. [3] pointed out that the linearity of DNN models is enough to form adversarial examples and they argued that adversarial examples can be explained as a property of high-dimensional dot products; they also highlighted that the direction of perturbation, rather than the specific point in space, mattered most. Tanay et al. [4] argued that the existence of adversarial examples was closely related to model classification boundary and introduced the “boundary tilting” perspective that adversarial examples existed when the classification boundary lay close to the submanifold of normal examples.

The discovery of the fragility of DNNs to adversarial examples triggered a range of research interests in adversarial attacks and defenses. A growing number of methods have been proposed to generate adversarial examples including L-BFGS [1], FGSM [3], and so

on. In order to defend against these attacks, researchers also introduced a range of defense methods to counter attacks by enhancing the robustness model [3,5–9], preprocessing input data [10–12], or attempting to differentiate adversarial examples from normal examples [13–17].

As an intuitive defense means, adversarial detecting has attracted a lot of attention. These methods can be divided into two categories: collecting disentangled features in the input space [18–20] or the activation space of target models [13,17,21]. Furthermore, most detection methods rely too much on target models to extract disentangled features. If we cannot get the target model, the methods may not work.

In this work, we propose a novel adversarial example detection method that is independent of whether we can get the target model or not. Our method utilizes the natural adaptation characteristics of the cosine distance to high-dimensional data and introduces predicted label information to measure the similarity between test data and normal data. In Figure 1, we outline our detection method. The extracted feature map from DNNs and the predicted label information are used to estimate the IACS values and the IACS estimates serve as features to train a linear regression classifier to classify the test data. The contribution of this paper is mainly threefold:

- We propose a similarity metric called the inner-class adjusted cosine similarity (IACS) and apply it to detect adversarial examples.
- Our detection method is independent of whether we can get the target model or not, and the extracted IACS values are stable enough to detect adversarial examples crossing attacks.
- Extensive experiments have been conducted and confirm that our method has excellent advantages in detecting adversarial examples compared with other detection methods. Moreover, our method further confirms that the direction of the adversarial perturbation matters most.

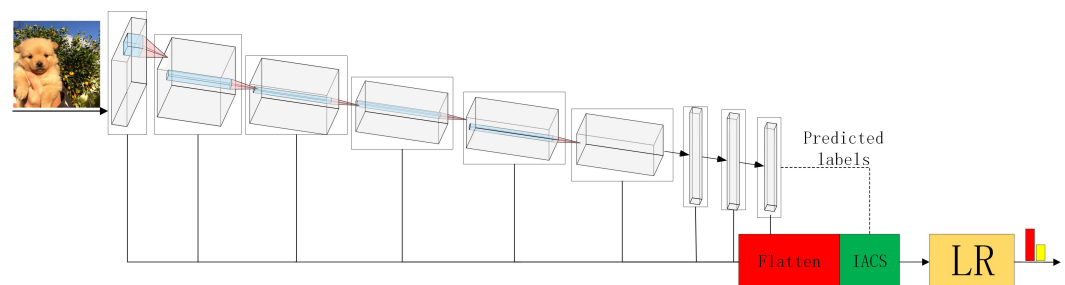


Figure 1. An overview of our detection method based on inner-class adjusted cosine similarity (IACS): We first extract the features of each layer and flatten them into one dimension. Then, the extracted features and predicted label information are used to calculate the IACS and further train the linear regression classifier to discriminate the IACS values of adversarial examples from those of normal examples.

2. Related Works

In this section, we discuss related works which include two parts: adversarial attack and adversarial defense.

2.1. Adversarial Attack

Adversarial attacks try to force deep neural networks (DNNs) to make mistakes by crafting adversarial examples with human imperceptible perturbations. We denote x as the input of DNN, C_x as the label of input x , and $f(\cdot)$ as the well-trained DNNs. Given x and network $f(\cdot)$, we can obtain the label of input x through forward propagation; in general, we can call x an adversarial example if $f(x) \neq C_x$. Here, we introduce five mainstream attack methods including FGSM, PGD, DeepFool, JSMA, and CW. They are all typical attack methods ranging from L_0 , L_2 , to L_∞ norms.

- **FGSM:** The fast gradient sign method (FGSM) was proposed by Goodfellow et al. [3] and is a single-step attack method. The elements of the imperceptibly small perturbation are equal to the sign of the elements of the gradient of the loss function with respect to the input; therefore, it is a typical l_∞ -norm attack method. The discovery of the FGSM also proved that the direction of the perturbation, rather than the specific point in space, mattered most.
- **PGD:** The projected gradient descent (PGD) was proposed by Madry et al. [7] and is a multistep attack method. As in the FGSM [3], it also utilizes the gradient of the loss function with regard to the input to guide the generation of adversarial examples. However, the method introduces random perturbations and replaces one big step with several small steps; therefore, it can generate more accurate adversarial examples but it also requires a higher computation complexity.
- **JSMA:** The Jacobian based saliency map attack (JSMA) [22] was proposed by Papernot et al. and is a typical l_0 -norm method. It aims to change as few pixels as possible by perturbing the most significant pixels to mislead the model. In this process, the approach updates a saliency map to guide the choice of the most significant pixel at each iteration. The saliency map can be calculated by:

$$S(X, t)[i] = \begin{cases} 0, & \text{if } \frac{\partial F_t(X)}{\partial X_i} > 0 \text{ or } \sum_{j \neq t} \frac{\partial F_j(X)}{\partial X_i} < 0, \\ \left(\frac{\partial F_t(X)}{\partial X_i} \right) / \sum_{j \neq t} \left| \frac{\partial F_j(X)}{\partial X_i} \right|, & \text{otherwise} \end{cases} \quad (1)$$

where i is a pixel index of the input.

- **DeepFool:** This algorithm was proposed by Dezfouli et al. [23] and is a nontarget attack method. It aims to find minimal perturbations. The method views the model as a linear function around the original sample and adopts an iterative procedure to estimate the minimal perturbation from the sample to its nearest decision boundary. By moving vertically to the nearest decision boundary at each iteration, it reaches the other side of the classification boundary. Since the DeepFool algorithm can calculate the minimal perturbations, therefore, it can reliably quantify the robustness of DNNs.
- **CW:** This refers to a series of attack methods for the L_0 , L_2 , and L_∞ distance metrics proposed by Carlini and Wagner [24]. In order to generate strong attacks, they introduced confidence to strengthen the attack performance, and to ensure the modification yielded a valid image, they introduced a *change of variables* to deal with the “box constraint” problem. As a typical optimization-based method, the overall optimization function can be defined as follows:

$$\text{minimize } D(x, x + \delta) + c * f(x + \delta), \quad (2)$$

where c is the confidence, D is the distance function, and $f(\cdot)$ is the cost function. We adopted the l_2 -norm attack in the following experiments.

Furthermore, there are black-box adversarial attack methods. Compared with white-box adversarial attacks, they are harder to work or need more perturbations, therefore are easier to be detected. In this paper, we focus on white-box attacks to test detectors.

2.2. Adversarial Defense

In general, adversarial defense can be roughly categorized into three classes: (i) improving the robustness of the network, (ii) input modification, and (iii) detecting-only and then rejecting adversarial examples.

The methods aimed to build robust models try to classify the adversarial example as the right label. As an intuitive method, adversarial training has been extended to many versions from its original version [3] to fitting on large-scale datasets [25] and to ensemble adversarial training [6]. Currently it is still a strong defense method. Although adversarial

training is useful, it is computationally expensive. Papernot et al. [8] proposed a defensive distillation to conceal the information of the gradient to defend against adversarial examples. Later, Ross et al. [26] refuted that the defensive distillation could make the models more vulnerable to attacks than an undefended model under certain conditions, and proposed to enhance the model with an input gradient regularization.

The second line of research is input modification, which modifies the input data to filter or counteract the adversarial perturbations. Data compression as a defense method has attracted a lot of attention. Dziugaite et al. [11] studied the effects of JPG compression and observed that JPG compression could actually reverse the drop in classification accuracy of adversarial images to a large extent. Das et al. [12] proposed an ensemble JPEG compression method to counteract the perturbations. Although data compression methods achieve a resistance effect to a certain extent, compression also results in a loss of the original information. In the article [10], the authors proposed a thermometer encoding to defend against adversarial attacks which could ensure no loss of the original information.

Detection-only defense is the other way to defend against adversarial attacks. We divided these methods into two categories: (i) detecting adversarial examples in the input space with raw data and (ii) using latent features of the models to extract disentangled features. For the first category of methods, Kheerchouche et al. [18] proposed to collect natural scene statistics (NSS) from input space to detect adversarial examples. Grosse et al. [19] proposed to train a new $N + 1$ class for adversarial examples classification. Gong et al. [20] constructed a similar method to train a new binary classifier with normal examples and adversarial examples.

The second category of adversarial detection methods uses the target model to extract disentangled features to discriminate adversarial examples. Yang et al. [17] observed that the feature attribution map of an adversarial example near the decision boundary was always different from the corresponding original example. They proposed to calculate the feature attributions from the target model and use the leave-one-out method to measure the differences in feature attributions between adversarial examples and normal examples and further detect adversarial examples. feinman et al. [21] proposed to detect the adversarial examples by kernel density estimates in the hidden layer of a DNN. They trained kernel density estimates (KD) on normal examples according to different classes, and the probability density values of adversarial examples should be less than that of those normal examples, by which they formed an adversarial detector. Schwinn et al. [27] analyzed the geometry of the loss landscape of neural networks based on the saliency maps of the input and proposed a geometric gradient analysis (GGA) to identify the out-of-distribution (OOD) and adversarial examples.

Most related to our work, Ma et al. [13] proposed to use the local intrinsic dimensionality (LID) to detect adversarial examples; the estimator of the LID of x was defined as follows:

$$LID\hat{(x)} = -\left(\frac{1}{k} \sum_{i=1}^k \log \frac{r_i(x)}{r_k(x)}\right)^{-1}, \quad (3)$$

where $r_i(x)$ denotes the distance between x and its i th nearest neighbor in the activation space and the $r_k(x)$ is the largest distance among the k -nearest neighbors. They calculated the LID value of samples in each layer and trained a linear regression classifier to discriminate the adversarial examples from normal examples. Our method used the same intuition, that is, we compared the test data with normal data, but we introduced the concept of inner class to limit the comparison scope within the same class label and unlike the LID calculating a Euclidean distance, we used a different basic similarity metric, the cosine similarity.

3. Method

In this section, we introduce our method in detail. Our method stems from the core idea of the fast gradient sign method (FGSM) [3] where the authors pointed out that the

direction of the perturbation mattered most. In other words, the adversarial perturbation was sensitive to angles or direction. As a result, we intuitively attempted to use the cosine similarity as the basic metric to discriminate the adversarial examples from normal examples. We studied the cosine similarity and its variant the adjusted cosine similarity [28], which introduces the normalization on the basis of cosine similarity. Furthermore, in order to fit the anomaly detection task, we introduced the predicted label information to extract the disentangled feature between normal examples and adversarial examples. The code is available at <https://github.com/lingKok/adversarial-detection-based-on-IACS>.

3.1. Basic Metric and Inner-Class Metric

On the basis of a basic metric, we introduce the idea of inner class and propose the inner-class cosine similarity (ICS) and inner-class adjusted cosine similarity (IACS). In this section, we introduce the basic metrics, the cosine similarity (CS) and adjusted cosine similarity (ACS), and the inner-class metrics, the ICS and IACS.

3.1.1. Cosine Similarity

The cosine similarity (CS) is a classical similarity measurement method that measures the similarity between two vectors. With the increase of dimensionality, similarities based on the Euclidean distance face the curse of dimensionality and their characterization ability cannot be guaranteed. Unlike the Euclidean distance, the cosine similarity can effectively measure the relationship between high-dimensional data. The cosine similarity (CS) can be formulated as follows:

$$CS(x, y) = \frac{x \cdot y}{\|x\| \|y\|}, \quad (4)$$

where $(\cdot)$ denotes the dot-product of two vectors.

3.1.2. Adjusted Cosine Similarity

The adjusted cosine similarity (ACS) is a variant of the cosine similarity. Although the cosine similarity can deal with the curse of dimensionality, it is more concerned with the relationship between the angles of vectors and is not sensitive to the absolute value of specific data such as size and length. Therefore, Sarvar et al. [28] proposed the concept of an adjusted cosine similarity. The adjusted cosine similarity offsets the shortcoming by subtracting the corresponding feature mean value. The adjusted cosine similarity of a sample x_i and sample x_j is given by :

$$ACS(x_i, x_j) = \frac{(x_i - \bar{x}) \cdot (x_j - \bar{x})}{\|x_i - \bar{x}\| \|x_j - \bar{x}\|}, \quad (5)$$

where $\bar{x}$ denotes the mean value of samples.

3.1.3. Inner-class Cosine Similarity

The inner-class cosine similarity introduces the concept of inner class on the basis of cosine similarity, which computes the cosine similarity limited to the same predicted class. Given the category of x , the ICS of x is calculated by:

$$ICS(x) = \frac{1}{|C(x)|} \sum_{x_j \in C(x)} CS(x, x_j), \quad (6)$$

where $C(x)$ denotes the set of samples with the same class as x , and $|C(x)|$ denotes the number of elements in set $C(x)$.

3.1.4. Inner-class Adjusted Cosine Similarity

Similar to ICS, the inner-class adjusted cosine similarity (IACS) computes the adjusted cosine similarity limited to the same predicted class. Given the category of x , the IACS of x is calculated by:

$$IACS(x) = \frac{1}{|C(x)|} \sum_{x_j \in C(x)} ACS(x, x_j). \quad (7)$$

3.2. Adversarial Detection Based on Inner-Class Cosine Similarity

In this section, we describe the implementation of the detection method in detail.

3.2.1. Notation and Terminology

Given a well-trained deep neural network classifier $f(\cdot)$, we denote the mixture data as $x_i \in D_{mix}$ (including normal and adversarial examples), the baseline data as $x_j \in D_{bsd}$ (only including normal data), $f_k(\cdot)$ as the output of the k_{th} layer of the classifier ($0 \leq k \leq n$), and $L_k(\cdot)$ as the flattened feature of the $f_k(\cdot)$.

3.2.2. Detector Training

In the training phase, we first collect the flattened features of each classifier layer. The flattening operation of sample x can be formulated as follows:

$$L_k(x) = \text{flatten}(f_k(x)), \quad (8)$$

where the $\text{flatten}(\cdot)$ denotes the flattening operation, which flattens the multidimensional data into one dimension.

Then, we calculate the adjusted cosine similarity of the mixture data $x_i \in D_{mix}$ with $x_j \in D_{bsd}$, which can be formulated as follows:

$$ACS(L_k(x_i), L_k(x_j)) = \frac{(L_k(x_i) - L_k(\bar{x}_i)) \cdot (L_k(x_j) - L_k(\bar{x}_j))}{\|L_k(x_i) - L_k(\bar{x}_i)\| \|L_k(x_j) - L_k(\bar{x}_j)\|}, \quad (9)$$

where $L_k(\bar{x}_i)$ denotes the average of the k_{th} layer output features of the mixture examples, and $L_k(\bar{x}_j)$ denotes the average of the k_{th} layer output features of the baseline examples (normal examples). This means we calculate the ACS values between the mixture data and normal data.

In order to better fit the anomaly detection task, we propose the inner-class adjusted cosine similarity (IACS) metric to detect adversarial examples. Given some label information predicted by classifier $f(\cdot)$, the adjusted cosine similarity (ACS) with the same predicted label as the x 's label is selected to calculate the mean value, which is used as the IACS value of the sample x at the k_{th} layer, as shown in Equation (10).

$$IACS_k(x) = \frac{1}{|C_k(x)|} \sum_{L_k(x_j) \in C_k(x)} ACS(L_k(x), L_k(x_j)), \quad (10)$$

where $C_k(x)$ denotes the set of the k_{th} layer output features of normal samples with the same label as sample x .

We next describe how the inner-class adjusted cosine similarity (IACS) estimates can serve as features to train a detector to discriminate adversarial examples from normal examples. Just as Algorithm 1 shows, the IACS values associated with each mixture sample are estimated with the baseline samples by Equations (9) and (10). Then, we use the IACS values (one value for one layer) to train a linear regression classifier, in which the IACS values from adversarial examples are labeled as 1 and the IACS values from normal examples are labeled as 0 in the experiment.

Algorithm 1 Adversarial detection algorithm based on IACS.**Require:** $f(\cdot)$: A target classifier trained well by normal examples. D_{mix} : Mixture dataset D_{mix} , $x_i \in D_{mix}$. D_{bsd} : Baseline dataset D_{bsd} , $x_j \in D_{bsd}$.**Ensure:** Linear regression classifier LR.

- 1: Extract the output of $f(\cdot)$'s layer: $\{f_k(x)\}_1^n$.
- 2: Flatten the output and get: $\{L_k(x)\}_1^n$.
- 3: **for** $k=1:n$ (number of layer) **do**
- 4: Calculate the mean value of $L_k(x_i)$ and $L_k(x_j)$ in a minibatch, $x_i \in M$, $x_j \in N$.
- 5: Calculate the adjusted cosine similarity by Equation (9) and get $ACS(L_k(x_i), L_k(x_j))$.
- 6: Calculate the IACS by Equation (10) and get $IACS_k(x_i)$.
- 7: **end for**
- 8: Set the feature $IACS(x_i)$ as 1 if x_i is from adversarial example else 0;
 $IACS(x_i) = [IACS_1, IACS_2, \dots, IACS_n]^T$
- 9: Train a linear regression classifier LR on $(IACS_{pos}, IACS_{neg})$.

In addition, note that there is no need to choose a very big baseline dataset (normal examples) to calculate the IACS values, provided that the baseline data is chosen relatively randomly and there are enough samples in the same category to fully maintain its inner-class characteristics. This can significantly reduce the computation load. In the experiment, we found that the detecting performance could be efficiently ensured even for a size of baseline data as small as 100, that is, 10 normal samples per class.

3.2.3. Detector Assessment

In the detecting phase, the test data can be classified by its IACS values. In fact, the trained linear regression classifier (LR) is a binary classifier, therefore, we used the AUC score to measure the performance of the LR. The AUC score denotes the area under the receive operating characteristic which can efficiently avoid the difference caused by manual selection thresholds. The closer the AUC score is to 1, the better the performance is and the closer it is to 0.5, the worse the performance of the LR is.

In experiments, we divided the mixture dataset into a training set and the test set with the ratio of 7:3. That is, we used the IACS values of the training set to train the LR and calculated the AUC score to measure the performance of the LR.

4. Experiments and Results

In this section, we evaluated the discrimination ability of IACS values between adversarial examples and normal examples and tested these features on the MNIST, SVHN, and CIFAR10 datasets. We conducted a comparison with the state-of-art methods including kernel density estimates (KD)-based method [21], local intrinsic dimensionality (LID)-based method [13] and natural scene statistics (NSS)-based method [18].

4.1. Experiment Settings

Hardware setup: All our experiments were conducted on a computer that was equipped with an Intel(R) Core(TM) i9-10920X CPU and an RTX 3080 GPU.

Model: The pretrained DNN model structure used for MNIST and SVHN was the same, that is, a Convnet with $3 \times 3 \times 16$, $3 \times 3 \times 32$, and $3 \times 3 \times 64$ convolutional layers followed by 2×2 max pooling layers and two 200-unit fully connected layers. They achieved an accuracy of 99.34% and 87.39% on MNIST and SVHN, respectively. For CIFAR10, we trained a fine-tuned Resnet20 with an additional linear layer. This model reported an accuracy of 87.09%. Refer to Table 1 for the detailed training parameters.

Table 1. Parameters set for training the classifier.

Parameter	MNIST	SVHN	CIFAR
Optimization Method	SGD	SGD	Adam
Learning Rate	0.05	0.05	0.001
Momentum	0.9	0.9	-
Batch Size	200	100	100
Epoch	20	40	200

Adversarial examples: We implemented five attacks based on an open uniform platform for security analysis [29], including: fast gradient sign method (FGSM) [3], projected gradient descent (PGD) [7], Jacobian based saliency map attack (JSMA) [22], DeepFool [23], and CW₂ [24].

- **FGSM:** We set the perturbation amplitude ϵ . For MNIST, we set the amplitude ϵ as 0.3. For SVHN and CIFAR10, we set it as 0.1.
- **PGD:** There were two parameters to set: the number of iterations it and the perturbation amplitude ϵ . In the experiment, we set it as 1000 for the three datasets and we set ϵ as 0.3 for MNIST and 0.1 for both SVHN and CIFAR10.
- **JSMA:** The perturbation coefficient θ was set to 1 and the modified pixel number was limited by the parameter γ , which was set to 0.2 for the three datasets.
- **DeepFool:** We set the number of iterations it as 50 and the overshoot coefficient as 0.02 for all datasets.
- **CW₂:** There were four parameters that could affect the adversarial examples: the number of iterations it , the confidence coefficient c , the number of search step n_s , and the learning rate lr . We set $c = 0$, $it = 1000$, $n_s = 10$, and $lr = 0.002$ for all datasets.

For each attack, we chose 1000 candidate samples from the test dataset (which were classified correctly by the target classifier) and generated the adversarial examples. We also chose an equal number of test samples as baseline data.

4.2. Evaluation of the Discrimination Ability of IACS

In this section, we evaluated the differences between adversarial examples and normal examples based on IACS values. Figure 2 shows the IACS values (from the penultimate layer of Resnet for CIFAR10) of 100 randomly selected adversarial examples (green) generated by CW₂ [24] and those of 100 random normal examples (red) from the CIFAR10 test dataset. We found that the IACS values for the normal examples were significantly larger than the IACS values for the adversarial examples. This met our expectation that the similarity between a normal example and a normal example was greater than that between an adversarial example and a normal example.

We also studied the cosine similarity as a basic metric. We evaluated the AUC score with just a single layer detector with IACS values and ICS values.

In Figure 3, we show the AUC score of each layer from the start layer to the end layer. We found that the overall performance of the IACS was better than that of the ICS. Notice that we only output one IACS or ICS value for each Resnet block for convenience.

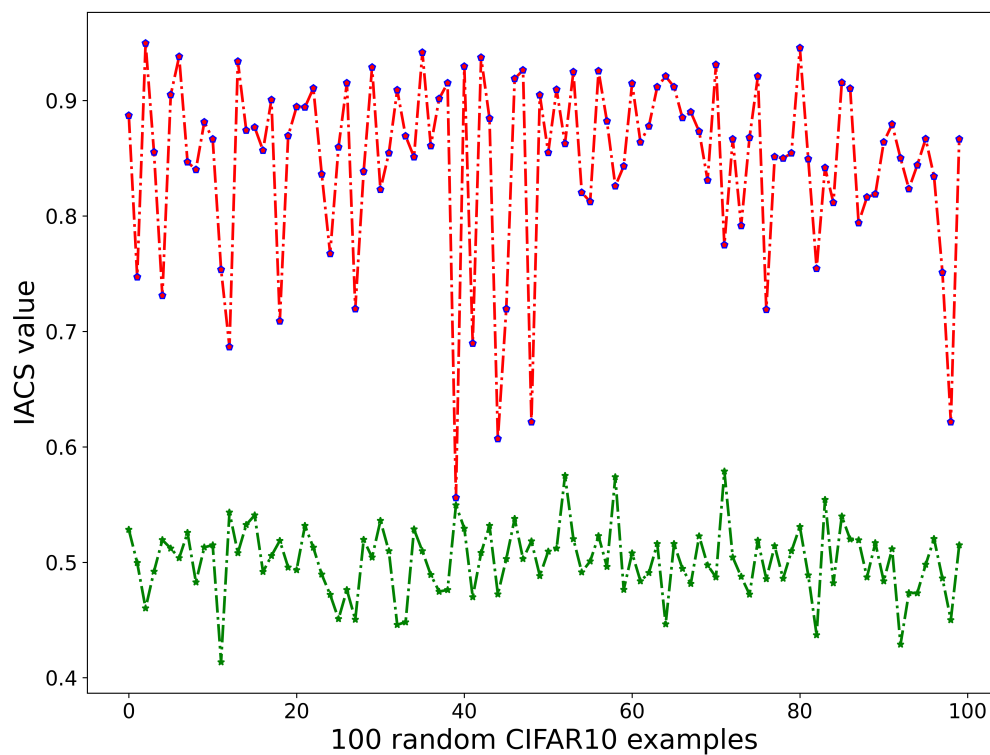


Figure 2. IACS comparison between normal and adversarial examples. The red points denote the normal examples' IACS values, and the green points denote the adversarial examples' IACS values.

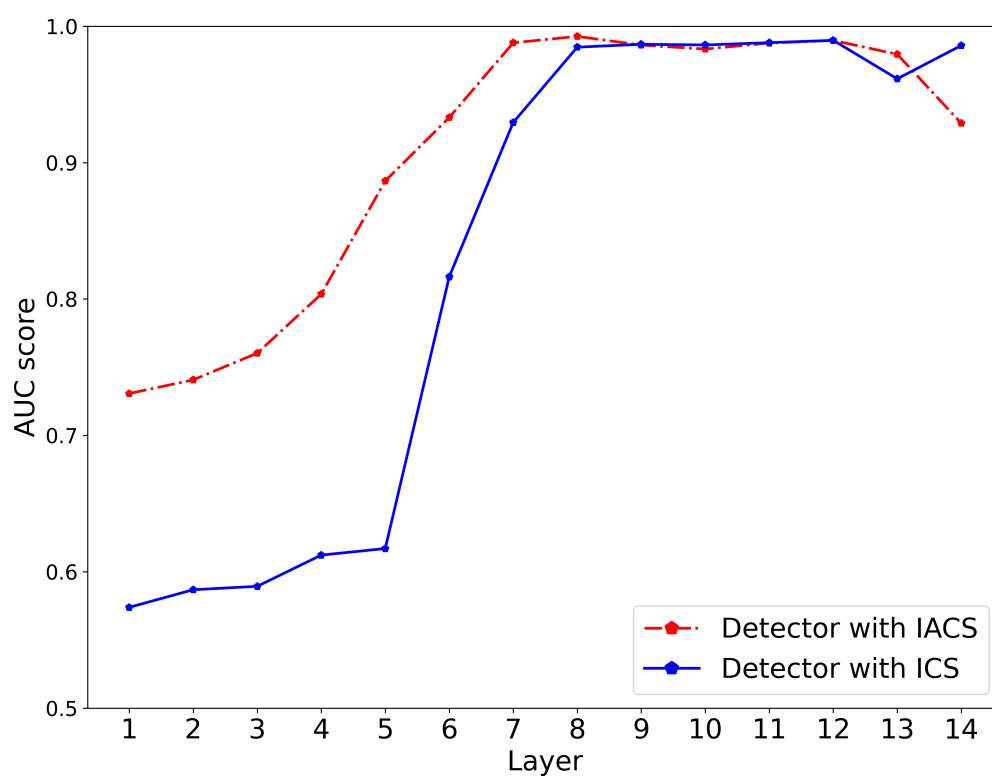


Figure 3. Single layer detector's AUC score with IACS and ICS.

4.3. Comparison with Other Methods

We conducted comparative experiments with other three state-of-the-art methods: kernel density estimates (KD)-based method [21], local intrinsic dimensionality (LID)-based method [13], and natural scene statistic (NSS)-based method [18], which are all supervised methods, as is our method. As Tables 2 and 3 show, we report the AUC score of different detection methods on different datasets with different attacks. We found our method achieved good results in almost all datasets and attacks. Especially on CW₂ [24], JSMA [22], DeepFool [23] attacks, our method had obvious advantages.

Table 2. The AUC score of different detection methods including the KD-based method, the LID-based method, the NSS-based method, and the IACS-based (our method) method on MNIST and SVHN datasets. The best results are highlighted in bold.

	MNIST				SVHN			
	KD	LID	NSS	IACS	KD	LID	NSS	IACS
FGSM	0.9284	0.9907	1.0000	1.0000	0.6787	0.996	1.0000	0.9987
PGD	0.8938	0.8929	1.0000	1.0000	0.7926	0.9735	1.0000	0.9982
DeepFool	0.9597	0.9844	1.0000	1.0000	0.5494	0.8048	0.5102	0.9996
JSMA	0.9711	0.983	1.0000	1.0000	0.6801	0.9225	0.9961	1.0000
CW₂	0.9847	0.9872	1.0000	1.0000	0.5163	0.7709	0.6250	1.0000

Table 3. The AUC score of different detection methods including the KD-based method, the LID-based method, the NSS-based method, and the IACS-based (our method) method on the CIFAR10 dataset. The best results are highlighted in bold.

	CIFAR			
	KD	LID	NSS	IACS
FGSM	0.7355	0.9950	0.9999	0.9832
PGD	0.9774	0.9950	0.9995	0.9898
DeepFool	0.6434	0.9109	0.5214	0.9837
JSMA	0.5847	0.7575	0.5248	0.9869
CW₂	0.716	0.9292	0.5239	0.9842

Crossing Attack Study: As an intuitive thought, we hoped the detector trained with one type of attack could be used to detect other types. Therefore, we studied the property of the detector's crossing attacks. We conducted the experiments on the CIFAR10 dataset and compared our method with the LID-based method [13], KD-based method [21], and NSS-based method [18] on different attacks. From Figure 4, we can observe that our method obtained better performance against crossing attacks than the other methods, which meant our method had the ability to detect unknown attacks. We speculated that it was because the IACS value was relatively stable on different attacks. To confirm our conjecture, we presented the IACS values at the penultimate layer on different attacks. As Figure 5 shows, the IACS values of normal examples distributed around about 0.85, and the adversarial examples around about 0.5. The results supported our conjecture.

Crossing Model Study: To further evaluate our method, we used a different model (ConvNet) with $3 \times 3 \times 32$ and $3 \times 3 \times 64$ convolutional layers on the CIFAR10 dataset to extract the features (it reported an accuracy of 84% on the test dataset). In other words, we detected adversarial examples generated by the different models. Table 4 reports the accuracy of the adversarial examples on different models and shows that the adversarial examples generated by DeepFool, JSMA, and CW₂ basically have no attack ability on Convnet. Then, we compared our method with the LID-based method [13] and the KD-based method [21], which rely on the target model to extract features. In Figure 6, we see that the performance did not change much and was even better on our method, while the performance of the other methods decreased significantly, especially the KD-based

method [21]. We conjectured that the adjusted cosine similarity could better seize the intrinsic differences between adversarial examples and normal examples even though the adversarial examples had no attack effect.

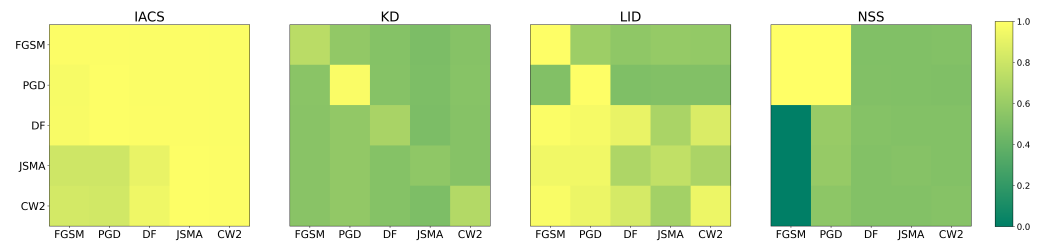


Figure 4. Crossing attacks performance: The horizontal axis represents the training set, and the vertical axis represents the test set. The closer the color is to yellow, the better the detector's performance is (DP refers to DeepFool).

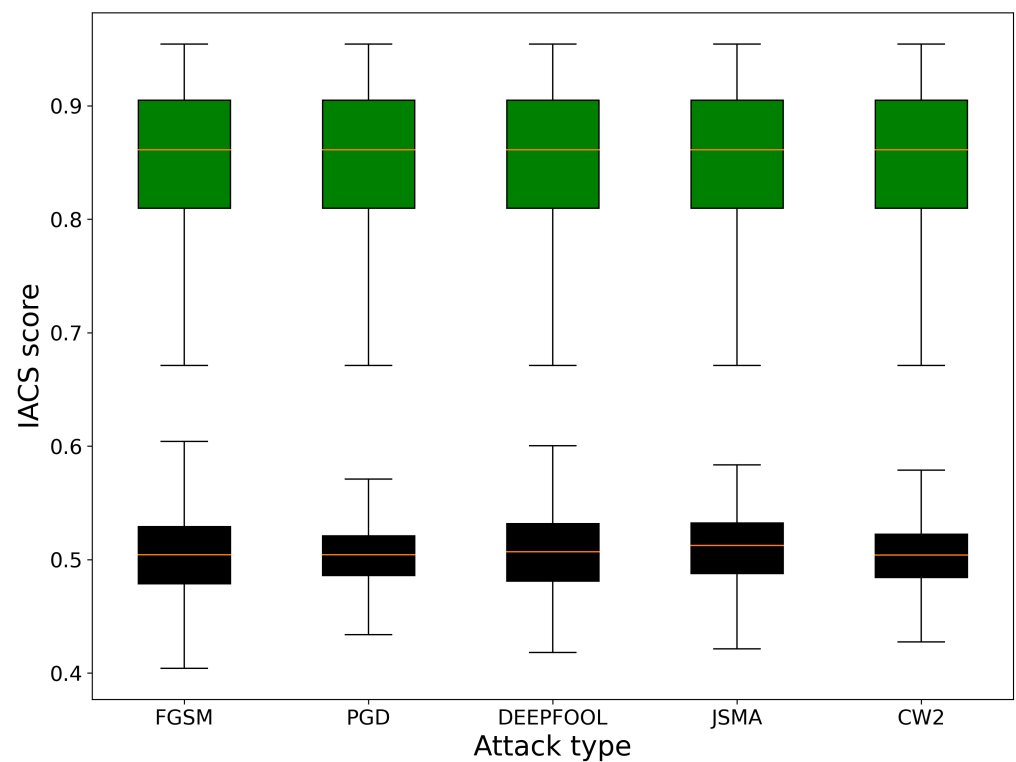


Figure 5. The IACS value with different attacks. Green denotes normal examples' box plots, and black denotes adversarial examples' box plots.

Table 4. The accuracy of different adversarial examples in Resnet (target model) and Convnet.

Model	Resnet	Convnet
FGSM	0.10	0.28
PGD	0.00	0.16
DeepFool	0.00	0.85
JSMA	0.09	0.81
CW ₂	0.01	0.83

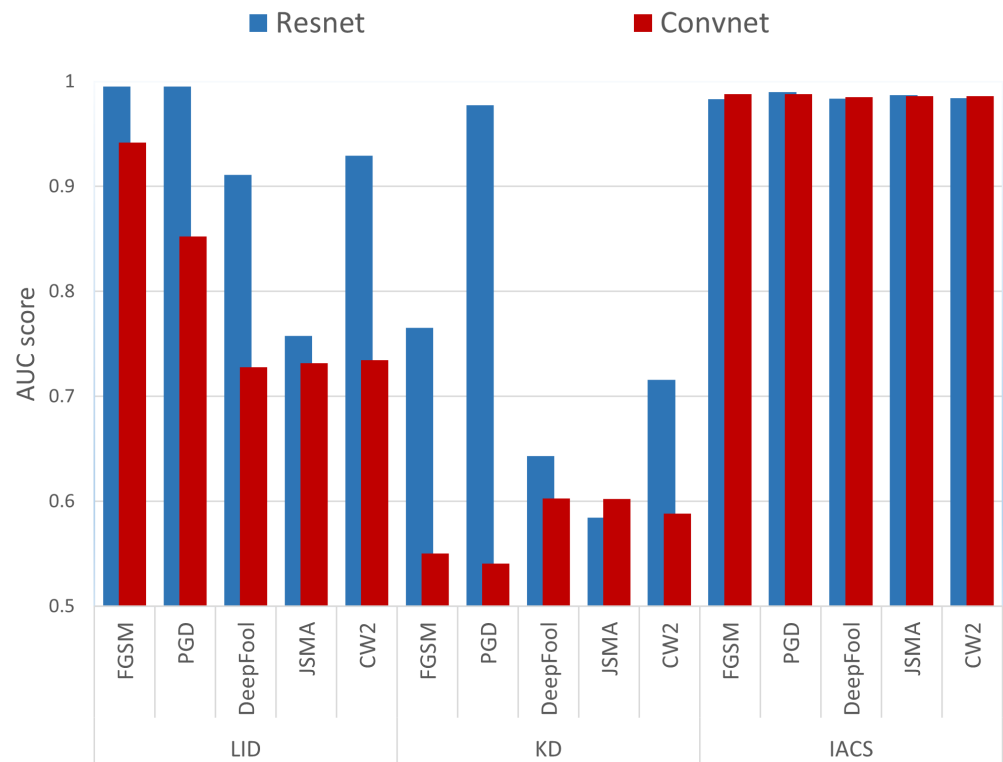


Figure 6. Crossing model performance: the green bars denote the AUC scores of the detector which extracts disentangled features from Resnet (target model), and the red bars denote the AUC scores of the detector based on Convnet.

5. Discussions

In order to figure out the reason why our method worked well, we further discuss the following problems.

Inner class: We performed an ablation study to analyze the contributions of the inner-class property. For comparison, we introduced the property of locality as a comparison with the property of inner class and leveraged K-nearest neighbors to capture the property of locality of the samples. We conducted comparative experiments on three datasets and five attacks with the local adjusted cosine similarity (LACS)-based method and the adjusted cosine similarity (ACS)-based method. For the LACS-based method, which is similar to our preliminary work [30], we averaged the adjusted cosine similarity in the k-nearest neighbors, but not within other normal samples with the same class as the sample. In the ACS-based method, we averaged the adjusted cosine similarity in a minibatch but the k-nearest neighbors or having the same label was not considered. As Table 5 shows, we found that without the inner-class property, the AUC score of the ACS-based method decreased significantly, and replacing the inner class with locality, the LACS-based method was not very efficient, especially for JSMA, DeepFool, and CW₂. These results meant that the label information played an important role. We speculated that this was because the label information predicted by the classifier limited the scope of comparison in the “same” class. As for why the ACS-based method had a low performance, we conjectured it was because the adjusted cosine similarity of adversarial examples was relatively low but the adjusted cosine similarity of normal examples with different classes’ samples was also low. Therefore, the averages of the adjusted cosine similarity were close.

Table 5. A comparison of discrimination power (AUC score of a logistic regression classifier) among IACS, LACS, and ACS methods on the different datasets and with different attacks. The best results are highlighted in bold.

	MNIST			SVHN			CIFAR		
	IACS	LACS	ACS	IACS	LACS	ACS	IACS	LACS	ACS
FGSM	1.0000	0.9968	0.5938	0.9987	0.9920	0.6683	0.9832	0.9485	0.5838
PGD	1.0000	0.8075	0.5532	0.9982	0.9328	0.7188	0.9898	0.9903	0.6985
DeepFool	1.0000	0.9485	0.5864	0.9996	0.7690	0.5730	0.9837	0.8758	0.8652
JSMA	1.0000	0.9539	0.5165	1.0000	0.8854	0.6695	0.9869	0.6764	0.5385
CW ₂	1.0000	0.9787	0.5910	1.0000	0.8689	0.5816	0.9842	0.9161	0.5680

Basic metric choice: In order to evaluate the contribution of the basic metric, the cosine similarity, we introduced the Euclidean distance as the basic metric, and we proposed the inner-class Euclidean distance (IED)-based method in which we averaged the Euclidean distance within the scope of the same predicted label. As Table 6 shows, the IACS had obvious advantages, especially for more complicated datasets. This further confirmed the advantages of the cosine similarity for high-dimensional data and that the direction of the adversarial perturbation mattered most.

Table 6. A comparison of discrimination power between IACS and IED method on the different datasets and with different attacks. The best results are highlighted in bold.

	MNIST		SVHN		CIFAR	
	IACS	IED	IACS	IED	IACS	IED
FGSM	1.0000	1.0000	0.9987	0.9920	0.9832	0.9958
PGD	1.0000	0.9541	0.9982	0.9328	0.9898	0.9546
DeepFool	1.0000	0.9878	0.9996	0.8690	0.9837	0.8759
JSMA	1.0000	0.9539	1.0000	0.7954	0.9869	0.7879
CW ₂	1.0000	0.9614	1.0000	0.8689	0.9842	0.8125

6. Conclusions

In this paper, we proposed an adversarial examples detection method based on the inner-class adjusted cosine similarity. By introducing the predicted label information and leveraging the natural advantages of the cosine distance on high-dimensional data, it greatly improved the detection ability on adversarial examples. Extensive experiments were conducted and showed that our method could achieve a greater performance gain compared with other detection methods. Most importantly, our method could be extended to extract the disentangled features with different models other than the target model (the adversarial attack model) and could also detect adversarial examples from crossing attacks. Therefore, our method had a wider scope of application. Moreover, our method further confirmed that the direction of the adversarial perturbation mattered most. For future research, it would be meaningful to explore more datasets, especially more complicated datasets, such as ImageNet, and other fields such as video outlier detection.

Author Contributions: Conceptualization, D.G.; methodology, D.G.; software, D.G.; validation, D.G. and W.Z.; formal analysis, D.G.; investigation, D.G.; resources, D.G.; data curation, D.G.; writing—original draft preparation, D.G.; writing—review and editing, D.G.; supervision, W.Z.; project administration, W.Z. funding acquisition, W.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (No. U1811462).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are available on request to the authors.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; Fergus, R. Intriguing properties of neural networks. *arXiv* **2014**, arXiv:1312.6199.
2. Yu, Z.; Zhou, Y.; Zhang, W. How Can We Deal With Adversarial Examples? In Proceedings of the 2020 12th International Conference on Advanced Computational Intelligence (ICACI), Yunnan, China, 14–16 March 2020; pp. 628–634.
3. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and Harnessing Adversarial Examples. *arXiv* **2015**, arXiv:1412.6572.
4. Tanay, T.; Griffin, L. A Boundary Tilting Perspective on the Phenomenon of Adversarial Examples. *arXiv* **2016**, arXiv:1608.07690.
5. Miyato, T.; Maeda, S.I.; Koyama, M.; Nakae, K.; Ishii, S. Distributional Smoothing with Virtual Adversarial Training. *arXiv* **2016**, arXiv:1507.00677.
6. Tramèr, F.; Kurakin, A.; Papernot, N.; Goodfellow, I.; Boneh, D.; McDaniel, P. Ensemble Adversarial Training: Attacks and Defenses. *arXiv* **2020**, arXiv:1705.07204.
7. Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; Vladu, A. Towards Deep Learning Models Resistant to Adversarial Attacks. *arXiv* **2019**, arXiv:1706.06083.
8. Papernot, N.; McDaniel, P.; Wu, X.; Jha, S.; Swami, A. Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. In Proceedings of the 2016 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2016; IEEE: San Jose, CA, USA, 2016; pp. 582–597. <https://doi.org/10.1109/SP.2016.41>.
9. Dong, Y.; Su, H.; Zhu, J.; Bao, F. Towards interpretable deep neural networks by leveraging adversarial examples. *arXiv* **2017**, arXiv:1708.05493.
10. Buckman, J.; Roy, A.; Raffel, C.; Goodfellow, I. Thermometer Encoding: One Hot Way to Resist Adversarial Examples. In Proceedings of International Conference on Learning Representations (ICLR), Vancouver, BC, Canada, April 30–May 3 2018; p. 22.
11. Dziugaite, G.K.; Ghahramani, Z.; Roy, D.M. A study of the effect of JPG compression on adversarial images. *arXiv* **2016**, arXiv:1608.00853.
12. Das, N.; Shanbhogue, M.; Chen, S.T.; Hohman, F.; Chen, L.; Kounavis, M.E.; Chau, D.H. Keeping the Bad Guys Out: Protecting and Vaccinating Deep Learning with JPEG Compression. *arXiv* **2017**, arXiv:1705.02900.
13. Ma, X.; Li, B.; Wang, Y.; Erfani, S.M.; Wijewickrema, S.; Schoenebeck, G.; Song, D.; Houle, M.E.; Bailey, J. Characterizing Adversarial Subspaces Using Local Intrinsic Dimensionality. *arXiv* **2018**, arXiv:1801.02613.
14. Gondara, L. Detecting Adversarial Samples Using Density Ratio Estimates. *arXiv* **2017**, arXiv:1705.02224.
15. Wang, J.; Dong, G.; Sun, J.; Wang, X.; Zhang, P. Adversarial Sample Detection for Deep Neural Network through Model Mutation Testing. In Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), Montreal, QC, Canada, 25–31 May 2019; pp. 1245–1256. <https://doi.org/10.1109/ICSE.2019.00126>.
16. Katzir, Z.; Elovici, Y. Detecting Adversarial Perturbations Through Spatial Behavior in Activation Spaces. In Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 14–19 July 2019; IEEE: Budapest, Hungary, 2019; pp. 1–9. <https://doi.org/10.1109/IJCNN.2019.8852285>.
17. Yang, P.; Chen, J.; Hsieh, C.J.; Wang, J.L.; Jordan, M.I. ML-LOO: Detecting Adversarial Examples with Feature Attribution. *arXiv* **2019**, arXiv:1906.03499.
18. Kherchouche, A.; Fezza, S.A.; Hamidouche, W.; Déforges, O. Detection of adversarial examples in deep neural networks with natural scene statistics. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–7.
19. Grosse, K.; Manoharan, P.; Papernot, N.; Backes, M.; Cispá, P.M.; Campus, S.I.; University, P.S. On the (Statistical) Detection of Adversarial Examples. *arXiv* **2017**, arXiv:1702.06280.
20. Gong, Z.; Wang, W.; Ku, W.S. Adversarial and Clean Data Are Not Twins. *arXiv* **2017**, arXiv:1704.04960.
21. Feinman, R.; Curtin, R.R.; Shintre, S.; Gardner, A.B. Detecting Adversarial Samples from Artifacts. *arXiv* **2017**, arXiv:1703.00410.
22. Papernot, N.; McDaniel, P.; Jha, S.; Fredrikson, M.; Celik, Z.B.; Swami, A. The Limitations of Deep Learning in Adversarial Settings. *arXiv* **2015**, arXiv:1511.07528.
23. Moosavi-Dezfooli, S.M.; Fawzi, A.; Frossard, P. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; IEEE: Las Vegas, NV, USA, 2016; pp. 2574–2582. <https://doi.org/10.1109/CVPR.2016.282>.
24. Carlini, N.; Wagner, D. Towards Evaluating the Robustness of Neural Networks. In Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–24 May 2017; IEEE: San Jose, CA, USA, 2017; pp. 39–57. <https://doi.org/10.1109/SP.2017.49>.
25. Kurakin, A.; Goodfellow, I.J.; Bengio, S. Adversarial Machine Learning at Scale. *arXiv* **2016**, arXiv:arXiv:1611.01236.
26. Ross, A.S.; Doshi-Velez, F. Improving the Adversarial Robustness and Interpretability of Deep Neural Networks by Regularizing Their Input Gradients. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; p. 10.
27. Schwinn, L.; Nguyen, A.; Raab, R.; Bungert, L.; Tenbrinck, D.; Zanca, D.; Burger, M.; Eskofier, B. Identifying untrustworthy predictions in neural networks by geometric gradient analysis. In Proceedings of the Uncertainty in Artificial Intelligence, PMLR, Online, 27–29 July 2021; pp. 854–864.

28. Sarwar, B.; Karypis, G.; Konstan, J.; Reidl, J. Item-based collaborative filtering recommendation algorithms. In Proceedings of the Tenth International Conference on World Wide Web—WWW '01, Hong Kong, China, 1–5 May 2001; ACM Press: Hong Kong, China, 2001; pp. 285–295. <https://doi.org/10.1145/371920.372071>.
29. Ling, X.; Ji, S.; Zou, J.; Wang, J.; Wu, C.; Li, B.; Wang, T. DEEPSEC: A Uniform Platform for Security Analysis of Deep Learning Model. In Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 20–22 May 2019; IEEE: San Francisco, CA, USA, 2019; pp. 673–690. <https://doi.org/10.1109/SP.2019.00023>.
30. Guan, D.; Liu, D.; Zhao, W. Adversarial Detection based on Local Cosine Similarity. In Proceedings of the 2022 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA), Dalian, China, 24–26 June 2022; pp. 521–525.