Detection of Adversarial Examples Based on Sensitivities to Noise Removal Filter

Akinori Higashi*[1], Minoru Kuribayashi*[2], Nobuo Funabiki*, Huy H Nguyen†, Isao Echizen†
* Okayama University, Okayama, Japan
E-mail: [1] p8m25ybd@s.okayama-u.ac.jp, [2] kminoru@okayama-u.ac.jp
† National Institute of Informatics, Japan

***Abstract*—An injection of malicious noise causes a serious problem in machine learning system. Due to the uncertainty of the system, the noise may misleads the system to the wrong output determined by a malicious party. The created images, videos, speeches are called adversarial examples. The study of fooling an image classifier have been reported as a potential threat for the CNN-based systems. The noise is well-designed so that the existence in an image is kept hidden from human eyes as well as computer-based classifiers. In this paper, we propose a novel method for detecting adversarial images by using the sensitivities of image classifiers. As adversarial images are created by adding noise, we focus on the behavior of outputs of image classifier for differently filtered images. Our idea is to observe the outputs by changing the strength of a noise removal filtering operation, which is called operation-oriented characteristics. With the increase of the strength, the output from a softmax function in an image classifier is drastically changed in case of adversarial images, while it is rather stable in case of normal images. We investigate the operation-oriented characteristics for some noise removal operations and the propose a simple detector of adversarial images. The performance is quantitatively evaluated by experiments for some typical attacks.**

I. Introduction

Deep neural networks have achieved dramatic success in many machine learning tasks such as image recognition and classification. However, DNN is known to be vulnerable to adversarial attacks [1], where adversarial noise is added to images or speech files so that a target DNN-based system outputs wrong results. An attacker can calculate intentional noise to mislead the output. Adversarial examples are perturbed images, videos and speeches that are created by optimizing the input to maximize the prediction error with minimum changes [1]. The perturbation is so tiny that it is difficult to detect with the human perceptual capability. Although some DNN-based systems can achieve near human accuracy, they do not perceive the input in the same way as humans do. Adversarial examples are a vulnerability of DNN-based system that can be abused from a malicious user to influence the behavior. Examples of systems that can be attacked using adversarial examples include face recognition and automatic driving system. Moreover, the DNN training data set could also be poisoned in the situation where the adversary has access to the training database. Since the existence of adversarial example, there is a threat in the spread of DNN-based system with such a vulnerability.

The attack methods of adversarial examples include non-targeted attacks that look for classes likely to be misidentified, and targeted attacks that misidentify any class of attackers. The basic approach of non-targeted attack is to first observe the output of a target DNN-based classifier from a modified input image which is perturbed from an original by adding a randomly generated noise. Similar to the training a neural network of an image classifier, the FGSM attack [2] uses the slope of the loss function for the update of the weights to modify the input so that a target classifier does not work correctly. Focusing on the linearity of the neural network, this method enables fast computation of the adversarial examples. It is high speed because it can be generated by computing the maximum noise vector once.

There are three types of approaches to defend against adversarial examples. One is to training the classifier to be robust against adversarial attacks [2]. It tries to include adversarial samples in the training set so that the classifier can recognize them. This technique is effective when the possible attacks are known beforehand. The second is to conceal the gradient information of a DNN-based classifier [3], [4]. It aims to mask gradients so that attackers can hardly leverage them to construct adversarial samples. The third is to detect adversarial examples. A statistical-based approach is used for the detection in [5], [6], raw images are used in [7], and features from intermediate layers of the targeted DNN are used in [8]. Feature Squeezing [9] is a detection method to analyze the fluctuation of the result of image classifier by applying image processing filter for the purpose of noise removal. This method detects adversarial images using the distance of outputs of softmax function for multiple filtered images, such as reduction of color bit pixels and smoothing filters. However, the detection method can be defeated using standard attacks, as long as the budget for the adversary is increased [10].

In this paper, we focused on the sensitivities of adversarial images to noise filtering operations, and proposed a simple architecture for detecting them. Based on the idea of feature squeezing, we use multiple filtering operations. Among several candidates of filtering operations, it is difficult to find the best combinations for extracting desirable features which can be used to classifying normal images and adversarial images. Instead, we change the strengths of one filtering operation and observe the transition of the outputs obtained from a CNN-based image classifier. Due to the sensitivities, the outputs

are changed according to the strength of filtering operations, which is operation-oriented characteristics. It is expected that the outputs are drastically changed in case of adversarial images while the outputs are almost stable in case of normal images. Under such an assumption, we extract a feature vector from the outputs of softmax function in a given CNN-based image classifier by changing the strengths of filtering operation. Using the feature vectors collected both from normal images and adversarial images, we train a simple classifier which is designed by some fully connected layers of neural network. The classification accuracy of the proposed method is evaluated both for targeted and non-targeted attacks in the experiments.

II. Related Works

In this section, we briefly review the techniques of adversarial attacks and defense techniques.

A. Adversarial Attack

There are two types of attacks to fool a DNN-based classifier: targeted attacks and non-targeted attacks. In the targeted attack, an attacker makes a classifier to misclassify a particular class while the attacker generates adversarial examples which are misclassified by the classifier into any class as long as it is different from the true class in non-targeted attack.

Consider that for an input image vector $\boldsymbol{x}$ classified into a class by an image classifier, the classifier classifies $\boldsymbol{x'} = \boldsymbol{x}+\boldsymbol{\eta}$ into different classes by adding a noise vector. In training a neural network of image classifiers, there is an attack that uses the gradient of the loss function used to update the weights to modify the input, so that the classifier does not work properly.

The optimization problem is formulated as follows to search for an adversary $\boldsymbol{x'}$ that minimizes distortion with normal images. For the purpose of optimizing this problem, the first term imposes a similarity between $\boldsymbol{x'}$ and $\boldsymbol{x}$. Since the second term facilitates the algorithm to find $\boldsymbol{x'}$ with a small loss value to class label t, a classifier is very likely to predict $\boldsymbol{x'}$ as t. By continuously varying the value of the constant c, we can find $\boldsymbol{x'}$ with a minimum distance up to $\boldsymbol{x}$x and at the same time deceive the classifier.

1) FGSM: Goodfellow et al. [2] proposed the Fast Gradient Sign Method (FGSM) for generating adversarial examples. It uses the derivative of the loss function of the model pertaining to the input feature vector. Given an input image, the gradient direction of each feature is perturbed by the gradient. Then the classification result of the input image will be changed.

Let $\boldsymbol{\theta}$ be a parameter of the classifier model and t be a class label for the correct answer to $\boldsymbol{x}$. Let $J(\boldsymbol{\theta}, \boldsymbol{x}, t)$ be the loss function to be used for training, and treat this loss function as a vector adjusted for the positive and negative signs of the small value ϵ so that the loss is increased by differentiating it by $\boldsymbol{x}$.

$$\boldsymbol{\eta} = \epsilon \cdot \mathrm{sign}\big(\Delta_x J(\boldsymbol{\theta}, \boldsymbol{x}, t)\big) \tag{1}$$

where sign() is a function that returns a positive or negative sign. The FGSM (Fast Gradient Sign Method) attack is a method of finding $\boldsymbol{\eta}$ in this way. In FGSM attacks, the value of the loss function corresponding to the specified class t increases when computing t, making it difficult to infer that t is the value of the loss function, which leads to misrecognition by the trainer.

2) LBFGS: In mathematical optimization, there is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [11], which is one of the iterative solution methods for non-restricted nonlinear optimization problems. There is the L-BFGS method, which has been modified to handle many variables, and the BFGS-B method, which deals with simple rectangular constraints. LBFGS attacks use these techniques to not only minimize the distance between the input image and the adversarial example, but also minimize the cross-entropy between the classification class of the adversarial examples and the class given as the target. This method requires the input images to be easily classified into the target class.

3) BIM: The BIM (Basic Iterative Method) attack [12] causes $\boldsymbol{x}$ to repeatedly apply the FGSM attack by running a finer optimization for multiple iterations. An adversarial image is created by applying the fast gradient sign method several times, and also by clipping the result in each iteration to avoid large changes on each pixel. The L1- and L2- versions of BIM implemented in FoolBox [13].

4) PGD: The PGD (Project Gradient Descent)-[12] attacks generate adversarial cases by repeatedly applying FGSM attacks as well as BIM attacks and projecting multiple perturbed examples as valid examples.

5) Deepfool: The Deepfool [14] is an non-targeted attack method to generate an adversarial example by iteratively perturbing an image. It explores the nearest decision boundary. An input image is modified a little to reach the boundary in each iteration. The algorithm stops once the modified image changes the classification of a classifier.

6) Carlini & Wagner Method: The Carlini & Wagner attack [15] can be targeted or non-targeted, and has three metrics to measure its distortion (L_0 norm, L_1 norm, and L_2 norm). The authors point out that the non-targeted l2 norm version has the best performance. It generates adversarial examples by solving the following optimization problem: This attack searches for the smallest perturbation measured by L_2 norm and makes a classifier classify the modified image incorrectly at the same time. The C&W method is known as a strong attack which is difficult to defend.

B. Defense Techniques

We discuss defense techniques against adversarial examples.

1) Adversarial Training: Adversarial Training [2] is a method to classify adversarial examples correctly by generating adversarial examples in the learning process and using them as supervised data for training. In the training phase, each time we generate an adversarial example for the network at that point in time and use a new loss function $J(\boldsymbol{\theta}, \boldsymbol{x}, t)$ that mixes the calculated loss function with the normal loss function.

$$\tilde{J}(\boldsymbol{\theta}, \boldsymbol{x}, t) = \beta \cdot J(\boldsymbol{\theta}, \tilde{\boldsymbol{x}}, t) + (1-\beta) \cdot J(\boldsymbol{\theta}, \boldsymbol{x}, t) \tag{2}$$

where β is a mixing parameter. Although this method improves the accuracy of classification of adversarial examples, it suffers from a decrease in the accuracy of identifying normal images.

2) Distillation: There is a method of defense called Defensive Distillation [16], which was developed to reduce the size of neural network. When training a new network model, the output of the original network model is used instead of supervised data, and the new model is trained to have the same output.

The softmax output of the original network model contains information on which classes have similar characteristics to each other, since probabilities are assigned in addition to the classes of correct answers. Here, if we edit the softmax layer to increase the probability of being assigned to a class that is not the correct answer, the gradient is reduced, which allows us to train a new network model with robustness. In other words, in order to fool the edited model, it is necessary to change the input to a level that is perceptible to humans.

However, it has been reported in [15] that the model created by Defensive Distillation was also able to cause false recognition. Many defensive methods have been used to defend against certain attacks, and most of them tend to focus on improving robustness against specific attacks. It is challenging task to be robust against unknown Attacks that are different from the attacks tested in a defense method.

3) Detecting Adversarial Examples: Different from the approaches of adversarial training and distillation, the detection of adversarial examples has been investigated. Feature Squeezing [9] takes advantage of the changeable nature of the classification results of CNN-based classifiers by applying image processing to adversarial images. The basic idea is to distinguish whether a given image is an adversarial example or not by the measuring the distance between the usual classification result and some image processing results in the confidence vector of each class of inference value output by the CNN. In [9], median filters and bit-depth reductions in which the number of bits of color in the image are tested to evaluate the sensitivities of classifier. In general, several image processing filters can be considered to be used together.

For the threshold determination, the value with the largest distance of the confidence vector in each image filter is used. However, since the values of the distances differed greatly among the image processing filters, there is a problem in the method of selecting the value to be used for the threshold judgment, i.e., the selection of the image processing filters. However, it is unreasonable to check the effects of many image processing filters.

III. Proposed Detecting Method

Adversarial Training [2] and Defensive Distillation [16] are both defense methods that are applied in the training phase, so they cannot be applied directly to trained models. Therefore, in this study, we focus on the detecting method assuming to use a pre-filtering operation. It can exclude adversarial images from inputting into a CNN-based image classifier.

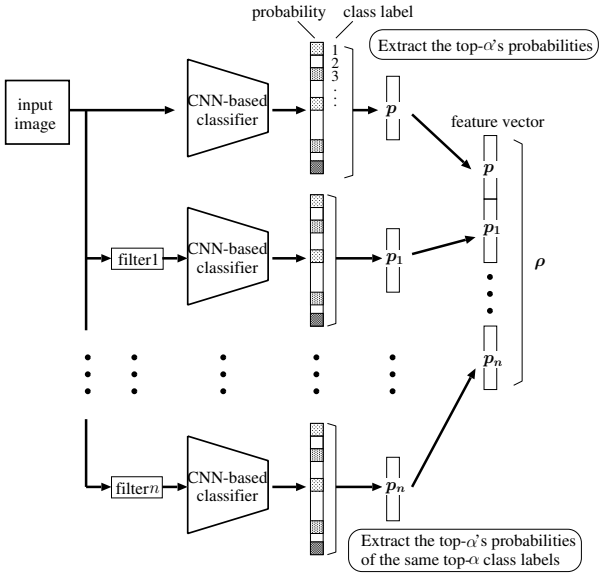

Fig. 1. Procedure of extracting feature vector $\boldsymbol{\rho}$.

A. Basic Idea

The adversarial images are created by adding noise to images so that a CNN-based image classifier can be falsely detected. Therefore, if a noise removal filter is applied to an image, the output is likely to vary.

As investigated in [9], low-pass filters can be used to remove noise in adversarial images. However, if the window size and filter parameters of the low-pass filter are known, the same low-pass filter can be used to create an adversarial image that is not subject to the noise removal effect. As a countermeasure, we change the strength of the filtering operation to observe the transition of the outputs from a CNN-based image classifier. Even if an attacker adjusts an attack tool according to a specific filter, we expect that the operation-oriented characteristics is still valid for the classification of adversarial images.

B. Extraction of Feature Vector

When a noise removal operation is performed, their probabilities are changed. In case of adversarial images, the probability of the top class label is drastically dropped. Depending on an exploited adversarial attack, the amount of changes in the probabilities of some top class labels are different. Especially for the case of adversarial images, the changes are completely different from normal images.

Suppose that a target image is first input into a CNN-based image classifier to obtain the top-α ($\alpha \geq 1$) class labels and their probabilities which are the output of softmax function. The top-α class labels and the probabilities are changed after a filtering operation. Our interest is to observe the transition of the probabilities of the same top-α class labels. The procedure of extracting the feature vector $\boldsymbol{\rho}$ is illustrated in Fig. 1.

The vector of the top-α class labels is denoted by

$$\boldsymbol{\ell} = (\ell_1, \ldots, \ell_\alpha), \tag{3}$$

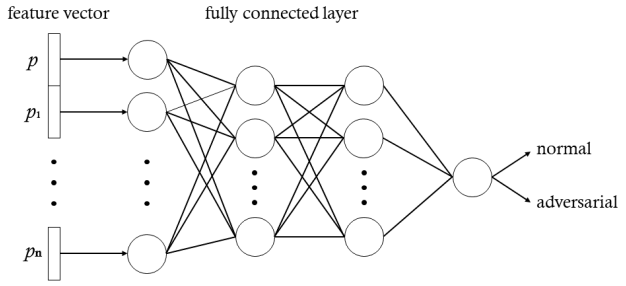

Fig. 2. Example of the classifier in the proposed method.

and their corresponding probabilities are

$$\boldsymbol{p} = (p_{\ell_1}, \ldots, p_{\ell_\alpha}). \tag{4}$$

A filtering operation with n different strength is applied to the target image. For i-th filtering operation, the probabilities of the top-α class labels are changed to $\boldsymbol{p_i}$:

$$\boldsymbol{p_i} = (p_{i,\ell_1}, \ldots, p_{i,\ell_\alpha}),\ (1 \leq i \leq n). \tag{5}$$

Finally, the original probability $\boldsymbol{p}$ and the n probabilities $\boldsymbol{p_i}$ are summarized to obtain an feature vector $\boldsymbol{\rho}$:

$$\boldsymbol{\rho} = (\boldsymbol{p}||\boldsymbol{p_1}||\cdots||\boldsymbol{p_n}), \tag{6}$$

where $||$ means a concatenation.

C. Classifier

The number of elements in the feature vector $\boldsymbol{\rho}$ is $\alpha(n+1)$. We construct a simple classifier based on a neural network. As the difference among some elements in $\boldsymbol{\rho}$ retains a certain level of characteristics of adversarial images, we use γ fully connected layers as the classifier. The illustration of the classifier with 3 layers is depicted in Fig. 2.

In order to train the classifier, we collect several feature vectors from normal images and their adversarial images by using some typical attacks. In addition, the feature vectors of filtered versions of such adversarial images are also collected as supervised data.

D. Noise Removal Filter

In this study, we consider a noise removal filter with arbitrary filter strength to realize a method for detecting adversarial images. The adversarial noise is regarded as noise which is less visible for human eyes. Such a signal can be removed effectively by lossy compression algorithms. So, we investigate the variability using JPEG compression, which is one of the nonlinear image processing filters that can gradually change the filter strength. Due to the lossy compression process in JPEG algorithm, non-useful information of image can be removed, and the strength can be easily controlled by its parameter of quality factor (QF).

In addition, we choose a scaling down filter which also reduce the entropy of an image, and vary the filter strength is easily changed by selecting the scaling factor. It is also interesting to consider the effects caused by the interpolation process when the reduced image is scaled up to the original image size. Then, we select the bilinear interpolation to reconstruct the image with the original size. It is because its low interpolating performance works as a noise removal operation.

IV. Experimental Results

We conduct experiments to evaluate the performance of the proposed method. In the experiments, we set $\alpha = 5$ to choose top-5 class labels and their probabilities. For the noise removal filters, we choose the JPEG algorithm and scaling down plus bilinear interpolation. The strengths of the filtering operations are controlled by setting $n = 7$ kinds of QF=$\{90, 90, 80, 70, 60, 50, 40, 30\}$ and the scaling factor $s = \{90, 80, 70, 60, 50, 40, 30\}$, respectively. As the feature vector $\boldsymbol{\rho}$ is composed of the probabilities of top-5 class labels for original and filtered images, the number of elements in is 40 in the experiment.

A. Experimental Setup

In this study, we generate adversarial images from normal images using the Foolbox library[1] with version 1.8.0 [13]. In the experiment, we assume a white box attack scenario such that an attacker can access to a target CNN-based image classifier. As the adversarial attacks do not always succeed to create an adversarial image, the success probability is dependent on the characteristics of a target image. For simplicity, we use default parameters for all attacks evaluated in this experiment. Data set used for verification of ILSVRC 2012 of ImageNet[2]. As for the image classifier, we select the trained model of VGG19 [17] and ResNet50 [18] which can classify into 1000 classes. From 1000 pieces in this data set, five targeted attacks, LBFGS [19], BIM [12], PGD[12], L1 and L2 distance minimization [13], are evaluated.

For non-targeted attacks, we used the six basic gradient attack method implemented in FoolBox [13], the fast gradient signed method (FGSM) [2], the ADef method proposed by Alaifari et al. [20], the DeepFool method [14], the Newton method [21], the Carlini and Wagner's method (C&W) [15], and the Saliency Map method [22].

We randomly selected 1,000 images were correctly classified (the ground-truth class labels were among the predicted top-5 results), and were added to the normal data set. Then, we performed the adversarial attacks on the selected images and selected misclassified ones which predicted top-5 results did not contain the previously predicted class labels). The misclassified adversarial images were then added to the adversarial data set. The number of samples successfully attacked in the Top-5 attacks for each attack and classifier is shown in Table I and II.

[1] https://foolbox.readthedocs.io/en/stable/

[2] http://www.image-net.org/

TABLE I
NUMBER OF SAMPLES SUCCEED TOP-5 ATTACK (TAGETED ATTACK)

	LBFGS	BIM	PGD	L1	L2
ResNet50	292	692	658	578	677
VGG19	711	830	866	573	751

TABLE II
NUMBER OF SAMPLES SUCCEED TOP-5 ATTACK (NON-TAGETED ATTACK)

	FGSM	Gradient	Deepfool	Newton	C&W	Saliency
ResNet50	841	841	798	762	639	247
VGG19	846	838	820	778	748	385

TABLE III
CLASSIFICATION ACCURACY OF DETECTING ADVERSARIAL EXAMPLES [%] (TARGETED ATTACK).

	FILTER	LBFGS	BIM	PGD	L1	L2	MIX
ResNet50	2 LPFs [9]	89.0	91.5	92.0	90.5	91.5	—
	JPG($n = 7$)	99.1	97.6	99.4	98.1	98.6	98.2
	SCL($n = 7$)	98.7	97.8	98.7	97.9	98.5	98.1
VGG19	2 LPFs [9]	82.4	80.7	81.4	81.1	80.4	—
	JPG($n = 7$)	99.1	97.3	98.8	97.8	96.7	98.3
	SCL($n = 7$)	99.3	97.6	99.0	97.3	96.8	97.4

TABLE IV
CLASSIFICATION ACCURACY OF DETECTING ADVERSARIAL EXAMPLES [%] (NON-TARGETED ATTACK).

	FILTER	FGSM	Gradient	Deepfool	Newton	C&W	Saliency	MIX
ResNet50	2 LPFs [9]	70.5	70.5	73.5	72.0	72.5	75.0	—
	JPG($n = 7$)	85.9	84.4	91.4	86.6	85.1	86.9	87.6
	SCL($n = 7$)	85.3	82.8	89.8	83.6	81.3	85.9	83.5
VGG19	2 LPFs [9]	56.3	56.2	58.5	57.3	58.6	64.4	—
	JPG($n = 7$)	82.2	80.7	89.4	79.8	81.3	81.7	82.4
	SCL($n = 7$)	79.6	79.6	85.2	77.1	78.7	83.1	80.0

B. Classification Accuracy

Using the images to which each adversarial attack is succeeded, we train the proposed classifier and measure the classification accuracy for each attack. It means that the supervised data are composed of normal images and the adversarial images which are produced by one specified attack. The results enumerated in Table III are classification accuracy against targeted attack. The two low pass filters (LPF) are the median filter with 3×3 window and the bit-depth reduction from 24-bit into 4-bit, which are used in [9]. In the table, "MIX" is the result for the case such that the supervised data involves all adversarial images produced by the five targeted attacks. From the results, we can say that the proposed method can detect the adversarial images with high accuracy.

The results for the non-targeted attacks are enumerated in Table IV. It is observed that the trained model of ResNet50 is better than that of VGG19 for calculating the probabilities of top-5 class labels. Compared with the case of targeted attacks, the accuracy is slightly lower. As the non-targeted attack finds a different class label close to the original one predicted by an image classifier, the changes in the probabilities in the top-5 labels can be controlled to be minimum. Thus, the detection of such adversarial images are much more difficult. The combination of some noise filters, e.g. JPEG plus Scaling down, will improve the performance, though it increases the computational complexity.

V. CONCLUSIONS AND FUTURE WORKS

In this study, we introduced operation-oriented characteristics, which is the sensitivities to a noise removal operation with some different strengths, and proposed a detection method based on the characteristics. The transition of outputs of a CNN-based image classifier helps us for the classification of normal images and adversarial images. Our approach can be used as a pre-processing operation to exclude adversarial inputs from inputting into a CNN-based system. From our experiments, the classification accuracy of the non-targeted attacks are slightly lower than that of the targeted attacks. For the improvement of the accuracy, one solution is to combine operation-oriented characteristics for some noise removal filters. The other possible solution is to replace the CNN-based image classifier to extract more useful feature vector. The investigation of such solutions are left for our future works.

Generally, the strength of noise can be modified according to a attacker's choice, and the adversarial attacks in the foolbox have such a parameter. In [10], the bypassing method against the feature squeezing had been presented by investigating the

sensitivities to the parameter. As the proposed method focuses on the transition of the outputs wit respect to the strength of filtering operation, the similar characteristics will be observed on the transition from the outputs with different strength. The effect of the number of filters on detection accuracy will be discussed in detail in a journal article. The analysis on such characteristics is also left for our future works.

ACKNOWLEDGMENT

This research has been partially supported by the JSPS KAKENHI Grant Number 19K22846.

REFERENCES

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *Proc. ICLR2014*, 2014.

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. ICLR2015*, 2015.

[3] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," in *ICLR 2014*, 2014.

[4] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *SP*. IEEE, 2016, pp. 582–597.

[5] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel, "On the (statistical) detection of adversarial examples," *arXiv preprint arXiv:1702.06280*, 2017.

[6] X. Li and F. Li, "Adversarial examples detection in deep networks with convolutional filter statistics," in *CVPR*, 2017, pp. 5764–5772.

[7] Z. Gong, W. Wang, and W.-S. Ku, "Adversarial and clean data are not twins," *arXiv preprint arXiv:1704.04960*, 2017.

[8] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," in *ICLR*, 2017.

[9] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: detecting adversarial examples in deep neural networks," in *Proc. NDSS2018*, 2018.

[10] Y. Sharma and P.-Y. Chen, "Bypassing feature squeezing by increasing adversary strength," *arXiv preprint arXiv:1803.09868*, 2018.

[11] R. Fletcher, *Practical methods of optimization (2nd ed.)*, John Wiley & Sons, 2000.

[12] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Proc. ICLR2017*, 2017.

[13] J. Rauber, W. Brendel, and M. Bethge, "Foolbox: A python toolbox to benchmark the robustness of machine learning models," *arXiv preprint arXiv:1707.04131*, 2017.

[14] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard, "Deepfool: a simple and accurate method to fool deep neural networks.(2016)," *arXiv preprint arXiv:1511.04599*, 2016.

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

[16] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS2014 Deep Learning Workshop*, 2014.

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR2015*, 2015.

[18] G. Huang, K. Q. Weinberger, and L. Maaten, "Densely connected convolutional networks," in *Proc. CVPR2017*, 2017.

[19] P. Tabacof and E. Valle, "Exploring the space of adversarial images," in *Proc. IJCNN2016*, 2016.

[20] R. Alaifari, G. S Alberti, and T. Gauksson, "Adef: an iterative algorithm to construct adversarial deformations," *arXiv preprint arXiv:1804.07729*, 2018.

[21] U. Jang, X. Wu, and S. Jha, "Objective metrics and gradient descent algorithms for adversarial examples in machine learning," in *Proceedings of the 33rd Annual Computer Security Applications Conference*. 2017, ACSAC 2017, p. 262–277, Association for Computing Machinery.

[22] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 2016, pp. 372–387.