Neurocomputing 445 (2021) 61–71

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Binary thresholding defense against adversarial attacks

Yutong Wang [a,b], Wenwen Zhang [a,c], Tianyu Shen [a,b], Hui Yu [d], Fei-Yue Wang [a,*]

[a] *The State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China*
[b] *School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China*
[c] *School of Software Engineering, Xi'an Jiaotong University, Xi'an 710049, China*
[d] *School of Creative Technologies, University of Portsmouth, Portsmouth PO1 2DJ, UK*

ARTICLE INFO

Article history:
Received 10 February 2021
Revised 9 March 2021
Accepted 11 March 2021
Available online 18 March 2021
Communicated by Zidong Wang

Keywords:
Binary thresholding
Defense
Adversarial training
Adversarial attack

ABSTRACT

Convolutional neural networks are always vulnerable to adversarial attacks. In recent research, Projected Gradient Descent (PGD) has been recognized as the most effective attack method, and adversarial training on adversarial examples generated by PGD attack is the most reliable defense method. However, adversarial training requires a large amount of computation time. In this paper, we propose a fast, simple and strong defense method that achieves the best speed-accuracy trade-off. We first compare the feature maps of naturally trained model with adversarially trained model in same architecture, then we find the key of adversarially trained model lies on the binary thresholding the convolutional layers perform. Inspired by this, we perform binary thresholding to preprocess the input image and defend against PGD attack. On MNIST, our defense achieves 99.0% accuracy on clean images and 91.2% on white-box adversarial images. This performance is slightly better than adversarial training, and our method largely saves the computation time for retraining. On Fashion-MNIST and CIFAR-10, we train a new model on binarized images and use this model to defend against attack. Though its performance is not as good as adversarial training, it gains the best speed-accuracy trade-off.

1. Introduction

In recent years, many breakthroughs have been made in the field of deep learning [1–3]. Because deep learning outperforms other machine learning methods and even human observers in diverse computer vision [4–7] tasks, the applications of deep learning in many crucial situations such as autonomous driving [8,9], face recognition [10,11] and medical image analysis [12–14] are increasing. However, the security problem is not investigated until 2014. Since the adversarial attack proposed by Szetegy [15] in 2014, more and more recent work [16–19] has focused on this field. For an clean image, it can be correctly classified by a Convolutional Neural Network (CNN) [20,21]. Adversarial attack is to add some malicious perturbation to this clean image, and produce an adversarial example, which will be incorrectly classified by the same classifier. But for human, this adversarial perturbation does not affect their classification and they will correctly classify the adversarial example. Defense methods [22–24] are proposed to defend against these attacks and achieve correct predictions.

With the development of attack and defense methods, the main issues remain unsettled. There is not a fast, simple and strong defense method that can be deployed on any datasets with any classification model. Although adversarial training is an effective defense method which can be used on different datasets, it costs too much computing resources to obtain a robust model. Other fast defense methods [25,26] based on input transformation cannot be applied to different datasets.

To develop a new defense method, we first investigate the defense characteristics of adversarially trained model. We visualize the feature maps of adversarially and naturally trained model in same architecture on MNIST [27]. We find that different from naturally trained model, adversarially trained model enhances the semantic information and denoises the adversarial perturbation during forward propagation. By visualizing the histogram of the feature maps from adversarially trained model, we find that adversarially trained model performs binary thresholding on input images to achieve robustness.

Since the direct manipulation on model parameters is infeasible, we conduct binary thresholding on input image before feeding it to a classifier, and find that this method could effectively defend against adversarial attack on MNIST and achieve comparable performance without modifying the training process. It achieves

* Corresponding author.
E-mail address: feiyue.wang@ia.ac.cn (F.-Y. Wang).

https://doi.org/10.1016/j.neucom.2021.03.036
0925-2312/© 2021 Published by Elsevier B.V.

99.0% accuracy on clean images, and 91.2% accuracy on white-box adversarial images. This performance is slightly better than adversarial training, and our method saves the computation time for retraining. Besides, unlike adversarial training, our defense is more robust to different strength of perturbation, especially larger pertubations. Theoretically, for other digit datasets, simply performing our binary thresholding defense during testing is effective.

For CIFAR-10 [28], though simply performing binary thresholding on input images during testing could gain some robustness, it could not meet our requirements. Further study find that unlike in MNIST, the image in CIFAR-10 after binary thresholding is totally different from the original one. And this results in a reduction on accuracy. To overcome this problem, we train a new model on binarized images and use this model to defend against attack. The newly trained model has 81.3% accuracy on clean images, and 40.5% accuracy on adversarial images. This performance is about 7% lower than adversarial training, but we decrease the computation time ninefold and achieve the best speed/ accuracy trade-off on CIFAR-10. The overall pipeline of our defense method is shown in Fig. 1.

The main contributions of this paper are threefold.

- First, by visualizing the feature maps of naturally and adversarially trained model on MNIST, we find adversarially trained model performs binary thresholding to denoise and resist the adversarial perturbation.
- Second, inspired by this, without modifying the training process, we simply perform binary thresholding on input images during testing to defend against PGD attack on MNIST. This defense method proves to be fast, simple and strong. And we give a detailed description on how to obtain the optimal threshold regardless of dataset.
- Third, because MNIST is a gray-scale digit dataset, after binary thresholding the input image is just the same as the original one. While CIFAR-10 is a color dataset, we cannot directly perform binary thresholding to gain robustness. We train a new model on binarized images and use this model to defend against attack, though the performance is not as good as adversarial training, it gains the best speed-accuracy trade-off on CIFAR-10.

The rest of this paper is organized as follows. Section 2 introduces the related work of adversarial attack and defense methods, including PGD attack and adversarial training in detail. Section 3 illustrates the defense characteristic of adversarially trained model and based on this characteristic, we propose our binary thresholding defense. Section 4 describes the experimental settings and defense results. Section 5 draws a conclusion, lists the weaknesses of our method and puts forward future work that boosts the development of CNN robustness.

2. Related work

We first explain the related work of adversarial attacks and defenses. Then we detail PGD attack and adversarial training, which we will mention in the following sections.

2.1. White-box attack and black-box attack

When the network architecture and the parameters of target model are known, we can conduct white-box attack based on this target model. But if we do not know the network architecture and the parameters of target model, we need to use a substitute and perform attack based on this substitute. This is called black-box attack. Black-box attack relies on the transferability of adversarial examples between different models, which means an adversarial example generated on one model could attack another model at certain probability.

Generally, white-box attack is more powerful than black-box attack. But in fact, it is always impossible to acquire the network architecture and the parameters of target model. So we need to conduct black-box attack as an alternative. For a defense method, white-box and black-box performance are important metrics. In this paper, we will use the accuracy under white-box and black-box attack to verify the effectiveness of our defense method.

2.2. Adversarial attacks

Given the architecture and parameters of the target model, early gradient-based attack methods are developed to solve the constrained optimization problem

$$\max_r L(x+r, y; \theta) \|r\|_\infty \leqslant \epsilon, \tag{1}$$

where L is the loss function, θ is the model parameters, x is the clean image, y is its ground-truth label, r is the adversarial perturbation added to x and ϵ is the allowed maximum perturbation. Contrary to standard training process, where we search for θ to minimize the loss function and make the output of the network closer to y. Here, we maximize the loss to make the output far away from y thus it causes misclassification. Meanwhile, we need to limit the maximum size of r, and make the adversarial perturbation invisible for human observer.

Based on Eq. (1), Goodfellow et al. [29] first propose an efficient, one-step method, named Fast Gradient Sign Method (FGSM) to generate perturbation based on the gradient of loss function with regard to the image pixel. Considering the large one-step method can be coarse, Kurakin et al. [30] propose a finer optimization, Basic Iterative Method (BIM), also known as Iterative FGSM (I-FGSM). BIM performs FGSM for several iterations with smaller steps and constrains the L_∞ norm in each iteration. To further improve attack method, Madry et al. [31] propose PGD. Instead of starting at the original input point, PGD starts at a random point in the allowed L_∞ norm ball, and avoids local maximas. Among all theses first-order attack methods, PGD is proved to be the strongest. In each iteration the update is

$$x^{t+1} = \mathrm{Clip}_{x,\epsilon}(x^t + \alpha \mathrm{sgn}(\nabla_x L(x, y; \theta))), \tag{2}$$

where $\mathrm{Clip}_{x,\epsilon}(x')$ function clips x' in ϵ-L_∞ neighborhood of x, α is the step size that moves in each iteration, $\mathrm{sgn}(.)$ is the sign function, and $\nabla_x L$ is the gradient of the loss function with regard to image pixel values. Note that besides the ϵ restriction, we also need to restrict the pixel value in the range of [0, 1] or [0, 255].

2.3. Defenses

To defend against adversarial attacks, many defense methods are proposed to improve the robustness of neural networks. The most intuitive and effective method is adversarial training [15,29,31,32]. As a universal defense method, adversarial training requires to discard the standard model, and retrain the network on adversarially perturbed images until the network learns to classify them correctly. Essentially, it solves a min–max problem

$$\begin{aligned} &\min_\theta \max_r L(x+r, y; \theta) \\ &\qquad \|r\|_\infty \leqslant \epsilon, \end{aligned} \tag{3}$$

that is, we first optimize r to generate adversarial examples remaining θ unchanged. Then, we optimize θ using the adversarial examples generated in the first step. Usually, we use PGD as the

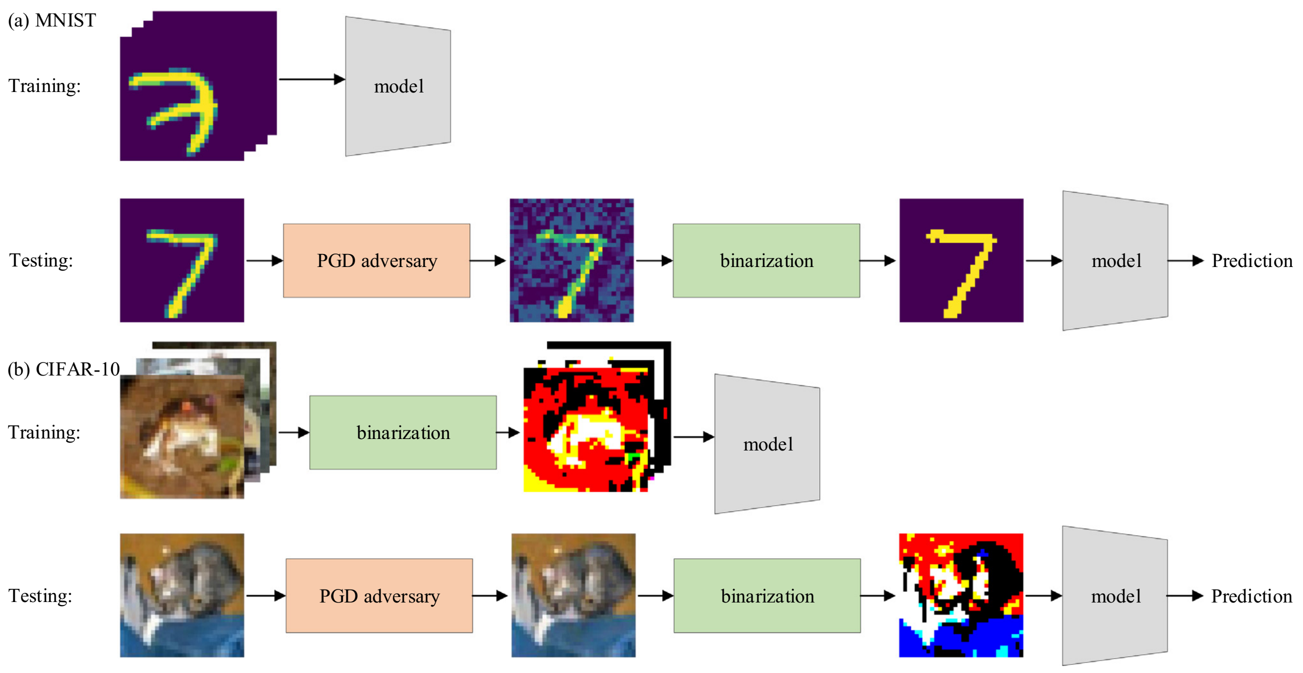


Fig. 1. Block diagram describing the differences between the defense pipeline on MNIST and CIFAR-10. For MNIST, we train the network on clean images (a). For CIFAR-10, we train the network on binarized images (b); and for both datasets, we test the networks on binarized adversarial images.

white-box attacker for adversarial training in the first step. Since a n-step PGD needs n back-propogations, the total amount of computation for adversarial training is approximately $n\times$ bigger than standard training. Also, it usually needs more training epochs to converge. Though the defense performance is pretty good, it needs substantial time to retrain the model. Especially when training on a large-scale dataset such as ImageNet [33], the computation time can be exploded. For an experimental environments where GPUs are inadequate, the defense cannot be implemented at all.

To overcome the weakness of adversarial training, there are other defenses working on simple image preprocess to defend against adversarial examples. Their advantages are threefold. First, they requires very few additional computations. Second, they can be deployed with any classification model. Third, they do not modify the classifier architecture or training process. Guo et al. [25] propose five image transformation methods to counter adversarial examples. These transformations include image cropping and rescaling [34], bit-depth reduction [35], JPEG compression [36], total variance minimization [37] and image quilting [38]. They combine different input transformations and effectively defend against attacks. They reduce images to 3 bits to perform bit-depth reduction, set quality level 75 (out of 100) to perform JPEG compression without any clues about how to obtain these parameters. Xie et al. [26] use random resize and random padding to achieve robustness. They resize the input shape from $299 \times 299 \times 3$ to $rnd \times rnd \times 3$, where rnd is an integer randomly sampled from the range of [299, 331). Then they randomly pad the resized image to the shape of $331 \times 331 \times 3$ empirically. They do not give a clear description about how to obtain the parameter either. And for sure, if the dataset is changed, these parameters are ineffective.

We propose a defense method that nearly avoids all these weaknesses and achieves the best speed-accuracy trade-off. We perform a simple binary thresholding with very few additional computations, and give a detailed description on how to obtain the optimal threshold in our defense regardless of dataset. It also can be deployed on any datasets with any classification model.

3. Proposed method

3.1. Denoising operations in adversarially trained model

To explore the differences between adversarially trained and naturally trained model, we feed a clean test image and its adversarial counterpart to these two models and visualize their feature maps after the first convolutional layer, as in [39]. The naturally trained model has more powerful adversarial examples, as verified in [31]. So based on a clean image in MNIST, as shown in Fig. 2a, we perform PGD attack on the naturally trained model to produce an adversarial example, as shown in Fig. 2b. The ground-truth label for the image is ‘7’, but the naturally trained model misclassifies the adversarial image as ‘9’, while the adversarially trained model correctly classifies the adversarial image.

The feature maps of the adversarially trained model on clean and adversarial image after the first convolutional layer are shown in Fig. 3. The feature maps after the first convolutional layer, ReLU layer, and max-pooling layer are shown in Fig. 4. We can see that the denoising operation occurs after the ReLU layer, and removes a significant chunk of the adversarial perturbation. The convolutional layer and ReLU can be formulated as

$$\mathrm{ReLU}(w^T x + b) \tag{4}$$

where $\mathrm{ReLU}(x) = \max(0, x)$. This can filter pixels with value less than

$$-\frac{wb}{\|w\|^2} \tag{5}$$

Feature maps after the second convolutional layer have blurry outlines. It is hard for human observer to determine if denoising operation is performed or not. So we select to investigate the feature maps after the first convolutional layer rather than the feature maps after the second convolutional layer.

The feature maps of the naturally trained model after the first convolutional layer, ReLU layer, and max-pooling layer, are shown

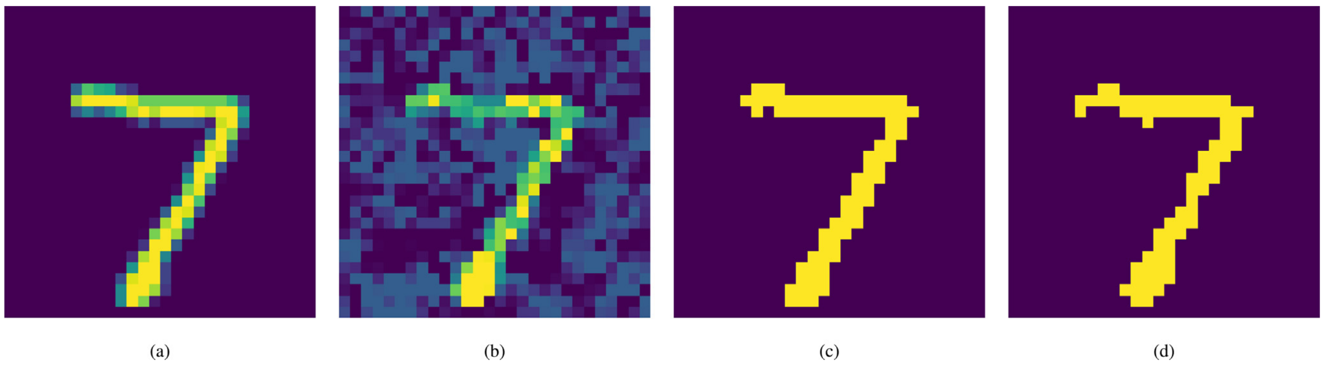


Fig. 2. The original clean image (a) and its adversarial counterpart (b). The binarized clean (c) and adversarial image (d). The ground-truth label is '7'. While the naturally trained model could correctly classify (a) as '7', it misclassifies (b) as '9'. (c) and (d) look very similar, and they are roughly identical to (a). These two binarized images do not cause misclassification any more.

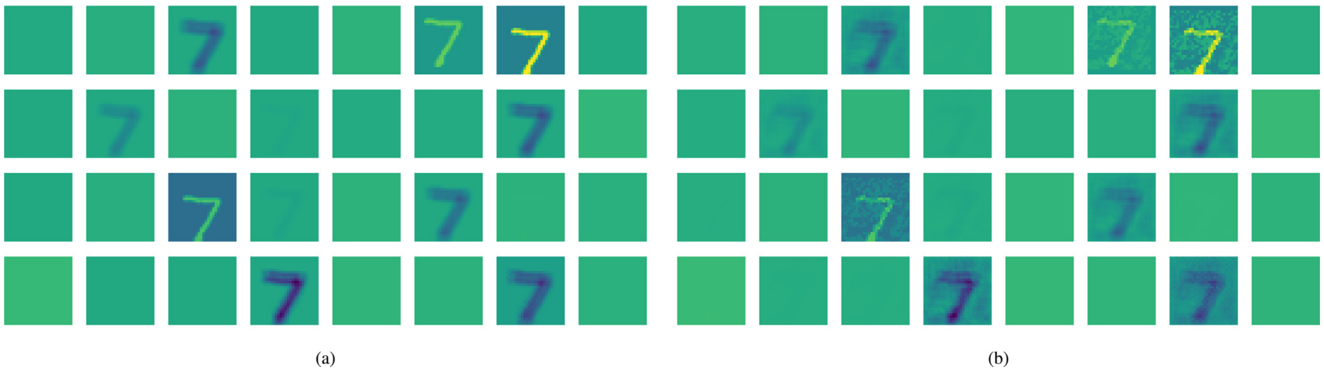


Fig. 3. The feature maps of the adversarially trained model on clean (a) and adversarial image (b) after the first convolutional layer. The adversarial perturbation in (b) is not removed.

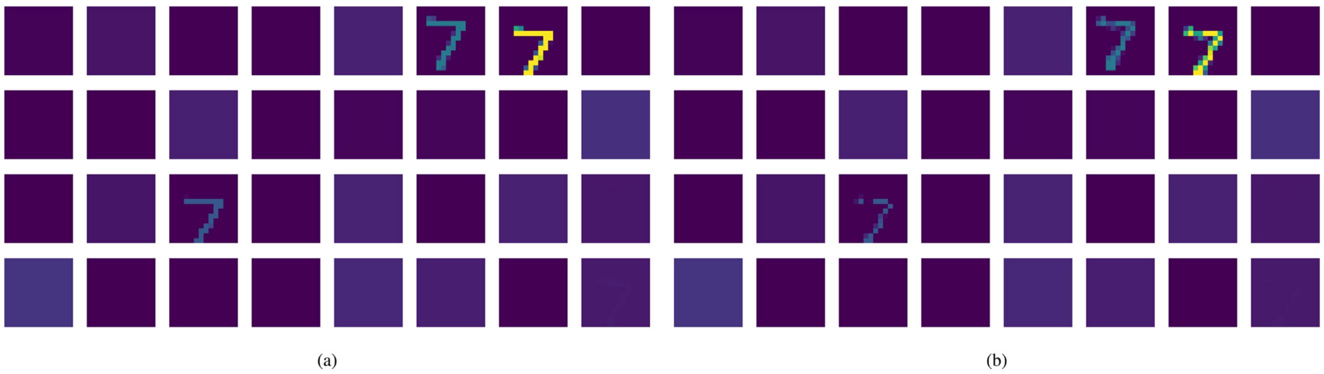


Fig. 4. The feature maps of the adversarially trained model on clean (a) and adversarial image (b) after the first convolutional layer, ReLU layer, and max-pooling layer. Only three channels have semantic activations on input, and these channels enhance the semantic information and resist the adversarial perturbation.

in Fig. 5. Compared with the naturally trained model, the feature maps of the adversarially trained model have clearer outlines and are more resistant to adversarial perturbation. It means the adversarially trained model enhances the semantic information and denoises the adversarial perturbation, while the naturally trained model amplifies the perturbation.

3.2. Binary thresholding in adversarially trained model

As shown in Fig. 4, only three channels play important roles in forward propagation and denoise the adversarial perturbation. We find the first convolutional and ReLU layer jointly perform as a threshold filter. To validate this operation, we visualize the histogram of the feature maps from the same channel in both models on clean and adversarial example. Here, we select the most activated channel to visualize. Since in the adversarially trained model, only three channels have semantic outputs regardless of input class, as shown in Fig. 4. Other channels have zero activations on both clean and adversarial image, and we cannot find if denoising is performed or not. Moreover, the strongest activation is always used to show the patterns learned by the network. It is generally considered that the strongest activation has the clearest information to decide the class of the input[40,41].

The histogram of the feature maps from both models are shown in Fig. 6. For the adversarially trained model, the most activated channels on clean and adversarial image are the same. But for

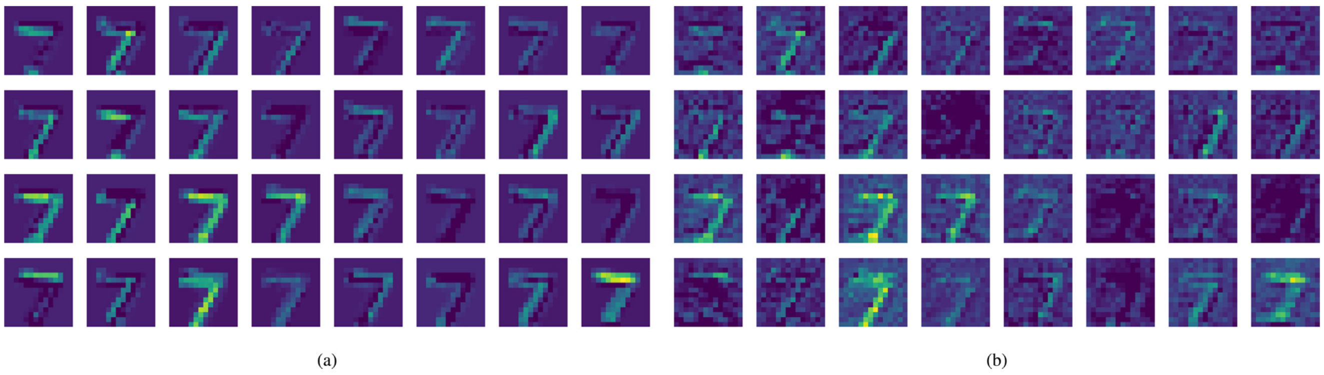


Fig. 5. The feature maps of the naturally trained model on clean (a) and adversarial image (b) after the first convolutional layer, ReLU layer, and max-pooling layer. The semantic information in (b) is severely obscured by adversarial perturbation.

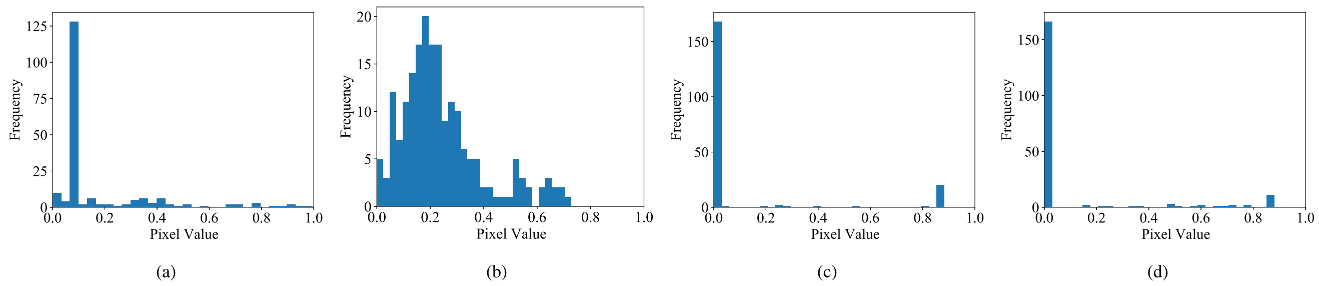


Fig. 6. The histogram of the feature maps from the same channel in the naturally trained model on clean (a) and adversarial image (b). The histogram of the feature maps from the same channel in the adversarially trained model on clean (c) and adversarial image (d). For the naturally trained model, the distributions of the feature maps on clean (a) and adversarial image (b) are totally different, mainly because of the adversarial perturbation added to the clean image, and this results in different predictions for these two images. For the adversarially trained model, the distributions of the feature maps on clean (c) and adversarial image (d) are roughly the same, surprisingly unaffected by perturbation, and the classifier could correctly classify both images.

the naturally trained model, the most activated channels on clean and adversarial image are different. As the naturally trained model could not correctly classify the adversarial image, we select to visualize the most activated channel on the clean image. For the naturally trained model, the distributions of the feature maps on clean and adversarial image are totally different, mainly because of the adversarial perturbation added to the clean image, and this results in different predictions for these two images. For the adversarially trained model, the distributions of the feature maps on clean and adversarial image are roughly the same, surprisingly unaffected by perturbation, and the classifier could correctly classify both images. And the feature maps are binary images. According to this, we deem the adversarially trained model performs binary thresholding on input images to achieve robustness.

3.3. Performing binary thresholding on input images

Since we cannot directly manipulate the model parameters to perform threshold filtering, we can achieve this before the image fed into the network. To do so, we need to specify the threshold value, and we assign 0 to the pixels whose value less than the threshold, 1 to the pixels whose value more than the threshold

$$dst(x,y,c) = \begin{cases} 1, & \text{if } src(x,y,c) > thresh \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

in which $src(x,y,c)$ is the pixel value at the position of (x,y) in c color channel before binarization, $dst(x,y,c)$ is the pixel value after binarization and *thresh* is the threshold value.

We should note that since each input image has distinct pixel distribution, the optimal threshold for each image is different. If we assign the same threshold for each image in the test set, that could lead to a decline of defense performance due to the unfit threshold. However, we cannot produce the optimal threshold for each image by line search, since the class of the image in test set is unknown.

Since we cannot obtain the label of test set, and the training set and test set are from a same distribution, we search the optimal value on training set. We put forward two strategies to obtain the optimal threshold value. The first strategy is we use a fixed threshold value to conduct binary thresholding over test set. Here, to find the optimal threshold, we first generate adversarial examples over the training set. Then we search the optimal threshold in the range of [0, 1] and obtain the defense performance on these adversarial examples. The best defense performance corresponds to the optimal threshold value, as shown in Fig. 8a. Similarly, the second strategy is we use a fixed percentile value over entire test set and compute the corresponding threshold value for each image. As the fixed threshold strategy does, we line search the optimal percentile in the range of [0, 1] over adversarial training dataset, as shown in Fig. 8b. To implement this, we do not need to generate adversarial images each time for different threshold value. To save the computation time of generating adversarial images, we only generate adversarial images once, and then apply different threshold to evaluate the performance.

For the fixed threshold strategy, at the optimal threshold 0.427, the accuracy on clean test images is 99.0%, and the accuracy on adversarial test images is 91.2%. For the fixed percentile strategy, at the optimal percentile 0.79, the accuracy on clean test images is 97.9%, and the accuracy on adversarial test images is 87.0%. The performance of optimal threshold is slightly better than optimal percentile. For both strategies, the accuracy on adversarial images starts from 0 and increases to the maximum and then decreases. The accuracy on clean images remains unchanged and decreases sharply in the end. In addition, for the fixed threshold strategy, in the range of around [0.3, 0.7], any threshold value could achieve pretty good defense performance. As for the fixed percentile strategy, only at around 0.8, the percentile could achieve good defense performance. And once the percentile is larger or smaller than 0.8, the performance drops quickly. So we choose to use the fixed threshhold to perform binary thresholding.

To validate the effectiveness of our strategy, we also search over test set to compare the results. We find that the optimal threshold value is 0.4 for test set. Using this threshold, the accuracy on clean images is 99.0%, and the accuracy on adversarial images is 90.1%. Considering the randomness of adversarial accuracy, the performance of optimal threshold over training set and over test set is identical. We can use the optimal threshold over training set to perform binary thresholding on test images, without significant performance decline and prior knowledge about the test set.

Though this fixed threshold strategy does not obtain the optimal defense method, it achieves compatible defense performance on test set, as shown in Table 1. Compared with adversarial training, our defense achieves better performance on clean images and on adversarial images. Note that we not only perform white-box attack on our defense, but also perform black-box attack. As we know, adversarial examples transfer across different models, which means an adversarial example generated from a model could also attack a different model [42–44]. As mentioned in [31], adversarial examples generated by naturally trained model are more powerful than generated by adversarially trained model. To conduct black-box attack on both our defense method and adversarial training, we train a new network using the same architecture and setups as the naturally trained model with independent initialization. The accuracy of the new naturally trained model on clean images is 99.2%, and the accuracy on adversarial examples is 0.0%, just the same as the naturally trained model. As we can see, the black-box attack successfully attacks the naturally trained model, its accuracy on black-box attack is 0.0%. Both our defense and adversarial training defend against the black-box attack, and achieve 95.6% and 96.1% accuracy, respectively. Our defense performance is just slightly lower than adversarial training.

If we cannot obtain the whole training set, we could also select a pretty good threshold for test set. We visualize the histogram of clean input image and its adversarial counterpart before and after binary thresholding, as shown in Fig. 7. As we can see, before binarization, the clean image is a binary image. Unlike the clean image, in the adversarial image, many pixels fall in the range of [0, 0.3], especially at around 0.3. This corresponds to the maximum perturbation $\epsilon = 0.3$. And once we set the threshold value slightly larger than 0.3, we could gain robustness on the adversarial image and make the distribution of adversarial image as in Fig. 6d. The clean image and its adversarial counterpart after binary thresholding is shown in Fig. 2c and d, as for human, they are just the same; as for classifier, both images could be correctly classified.

As mentioned above, if we search the optimal threshold on training set, we only need to generate adversarial examples on training set once, evaluate the defense performance on different threshold and find the optimal threshold. We do not need to retrain the model, as shown in Fig. 1(a).

3.4. Training on binarized images

While our binary thresholding defense is considerably effective on MNIST, when it is exploited on CIFAR-10, it results in poor performance. Intuitively, we can foresee this problem. MNIST is a gray-scale digit dataset, while CIFAR-10 is a color dataset. In MNIST, after binary thresholding operation, the image is just the same as the original one, as shown in Fig. 2c and d. So binary thresholding is directly useful for MNIST.

But for CIFAR-10, after binary thresholding operation, the input image is quite different from the original one, as shown in Fig. 1b. Note that, for gray datasets like MNIST, we use one optimal threshold for the single channel. For RGB datasets like CIFAR-10, we also use one optimal threshold for the three channels. Even if we use the optimal threshold of 121, the accuracy on clean test set is 38.8%, and the accuracy on adversarial test set is 28.1%. Though this indicates that our method has defensive ability, the defense performance is not good enough compared to adversarial training. As shown in Table 3, for adversarial training on CIFAR-10, the accuracy on clean test set is 87.3%, and the accuracy on adversarial test set is 47.2%.

To overcome this problem, we consider training a new model on binarized images and use this model to perform defense, as shown in Fig. 1(b). We select the optimal threshold 121 and use this threshold to perform binarization over training dataset. Using the same architecture and settings, we train a new model on these binarized images. At test time, we perform PGD attack on this model, and evaluate the performance of the new model on binarized images. We find that the accuracy on clean test set is 81.3%, and the accuracy on adversarial test set is 40.5%, as shown in Table 3. The performance is about 7% lower than adversarial training. As in MNIST, to conduct black-box attack, we train a new network using the same architecture and setups as the naturally trained model with independent initialization. The accuracy of the new naturally trained model on clean images is 95.0%, and the accuracy on adversarial examples is 0.0%, just the same as the naturally trained model. As we can see, the black-box attack successfully attacks the naturally trained model, and the accuracy on black-box attack is 16.1%. Both our defense and adversarial training defend against the black-box attack, and achieve 72.8% and 85.9% accuracy, respectively. Our defense performance is 13% lower than adversarial training. But we decrease the computation time ninefold and achieve the best speed/ accuracy trade-off on CIFAR-10.

The reason why the performance of our method is lower than adversarial training may be that, CIFAR-10 is not a binary dataset. The images in MNIST almost like binary images, and the majority of the pixel value is 0 and 1. After binary thresholding, the adversarial perturbation about 0.3 can be removed completely. But for CIFAR-10, the images are not binary images, and the pixel values spread over the range of [0, 255]. Simply performing binary thresholding may obscure some important features and cannot remove all the adversarial perturbation.

3.5. A high level understanding of our defense

As described in [31], the clean data points can be easily separated by a naturally trained model with a simple decision boundary. As for adversarial examples which are close to the decision boundary, the naturally trained model cannot separate them correctly. So, to separate them carefully, adversarial training needs to learn a complicated decision boundary and will need a larger capacity network and much more training time. On the contrary, we do not pursue the accurate complicated decision boundary, we map the adversarial examples to data points where are far away from the simple boundary and can be correctly separated.

In this way, our defense is similar to Defense-GAN [45] and PixelDefend [46] in essence. Song et al. devise PixelDefend, using a PixelCNN generative model [47,48] to project potential adversarial example back onto the regions of training data before feeding it into a classifier. Samangouei et al. propose Defense-GAN, leveraging a Generative Adversarial Network (GAN) [49,50] to project samples onto the range of the generator before classifying them.

Table 1
MNIST: Speed/ accuracy trade-off for the naturally trained model, the adversarially trained model and the naturally trained model with our defense on clean test set, white-box and black-box PGD adversarial test set (BT denotes binary thresholding).

Model	Natural	White-box PGD	Black-box PGD	Training time (FPS)	Inference time (FPS)
Standard training	99.2	0.0	0.0	23,000	86,410
Standard training w BT	99.0	91.2	95.6	23,000	83,500
Adversarial training	98.5	90.3	96.1	597	86,410
Adversarial training w BT	98.1	93.7	96.3	597	83,500

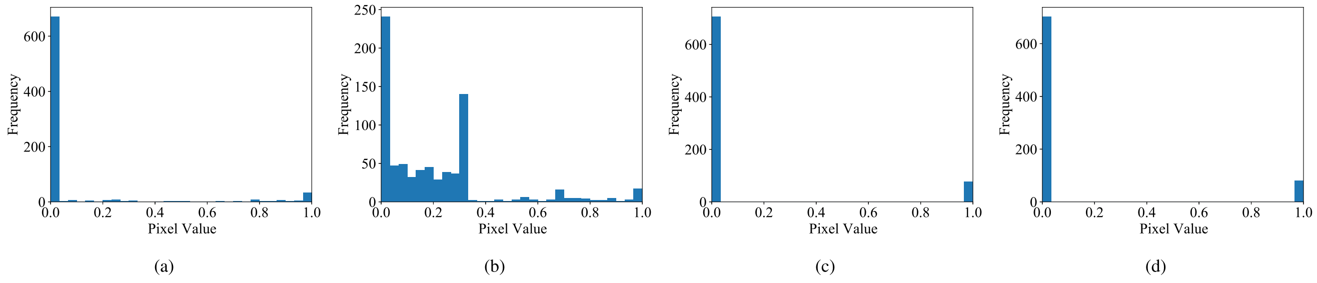


Fig. 7. The histogram of the original clean (a) and adversarial image (b). The histogram of the clean (c) and adversarial image (d) after binarization. Before binarization, the distributions of the clean (a) and adversarial image (b) are totally different, mainly because of the adversarial perturbation added to the clean image, and this results in different predictions for these two images. After binarization, the distributions of the clean (c) and adversarial image (d) are roughly the same, and the classifier could correctly classify both images.

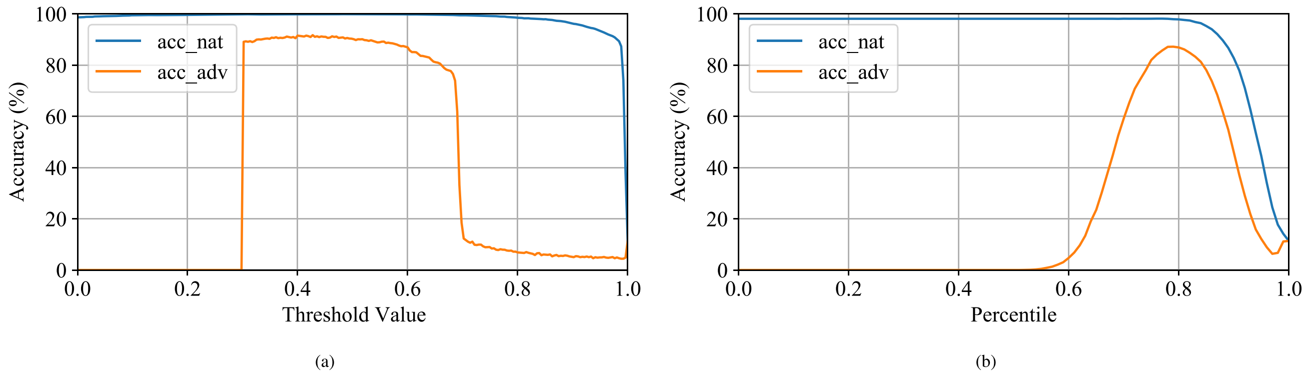


Fig. 8. The accuracy on clean and adversarial test set with different threshold value (a). The accuracy on clean and adversarial training set with different percentile (b).

These two methods are very similar, except they use different generative models. Though these defenses are shown to be feasible mechanism against adversarial attacks, it is still a challenge to train and tune generative model.

Compared with adversarial training, PixelDefend and Defense-GAN, we only need to conduct an extra binary thresholding and thus save a lot of computations at both training and inference time. At last, we need to emphasize the main advantages of our defense method. 1) Our method achieves compatible defense performance with adversarial training. 2) We perform a simple binary thresholding with very few additional computations during training and testing, and give a detailed description on how to obtain the optimal threshold in our defense regardless of dataset. 3) Our method can be deployed with any classification model on any dataset.

4. Experiments and results

4.1. Experimental settings

Three classification datasets are used in our experiments. MNIST has 60,000 training images and 10,000 test images. Each image is a 28×28 gray-scale image having a label from 1 of 10 classes. Fashion-MNIST [51] is just like MNIST. CIFAR-10 consists of 60,000 images, where 50,000 used for training and 10,000 for testing, and each image is a 32×32 color image with a label from 1 of 10 classes.

For MNIST, we use the same experimental settings as in [31][1] based on TensorFlow [52]. We use two convolutional layers with 32 and 64 filters in 5×5 size, each followed by a ReLU and a 2×2 max-pooling, and a fully connected layer of size 1024 to build the network. The naturally trained model is trained on clean images for 25 epochs, and the adversarially trained model is trained with PGD-made adversarial images for 100 epochs. The learning rate is 10^{-4}, and the batch size is 50. For PGD-made adversarial images used for training and testing, we run $n = 40$ iterations with a step size of $\alpha = 0.01$, and the maximum perturbations for each pixel is $\epsilon = 0.3$. For Fashion-MNIST, we use the same experimental settings as in MNIST, except we train the binarized model for 5 epochs.

For CIFAR-10, we use the same experimental settings as in [31].[2] We use a wide residual network [53] WRN-30-10 architecture, which denotes a residual network with 30 layers and 10 times wider than original residual network. Both the naturally and the adversar-

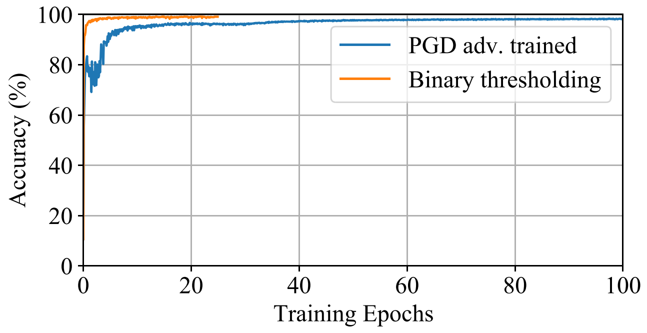
[1] https://github.com/MadryLab/mnist_challenge
[2] https://github.com/MadryLab/cifar10_challenge

Table 2
Fashion-MNIST: Speed/ accuracy trade-off for the naturally trained model, the adversarially trained model and the naturally trained model with our defense on clean test set, white-box and black-box PGD adversarial test set (BT denotes binary thresholding).

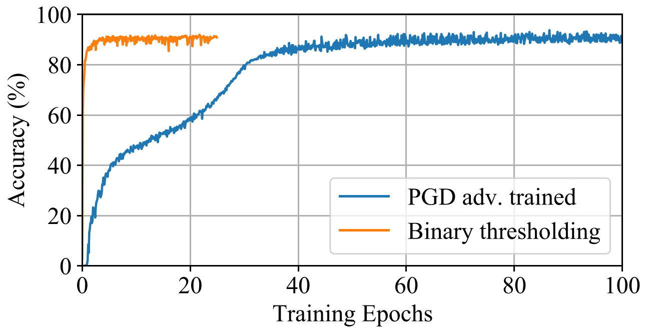
Model	Natural	White-box PGD	Black-box PGD	Training time (FPS)	Inference time (FPS)
Standard training	90.5	0.0	0.0	22,570	84,590
Standard training w BT	82.6	36.9	57.3	22,570	85,720
Binary training	86.2	47.0	67.5	21,600	85,720
Adversarial training	70.8	60.7	74.8	569	84,590
Adversarial training w BT	76.0	62.8	74.3	569	85,720

Table 3
CIFAR-10: Speed/ accuracy trade-off for the naturally trained model, the adversarially trained model and the naturally trained model with our defense on clean test set, white-box and black-box PGD adversarial test set (BT denotes binary thresholding).

Model	Natural	White-box PGD	Black-box PGD	Training time (FPS)	Inference time (FPS)
Standard training	95.0%	0.0%	16.1%	282	851
Binary training	81.3%	40.5%	72.8%	278	840
Adversarial training	87.3%	47.2%	85.9%	32	851



(a)



(b)

Fig. 9. Comparison of the convergence rate of our defense method and adversarial training on MNIST. (a) shows the accuracy on clean test set, while (b) shows the accuracy on white-box PGD test set.

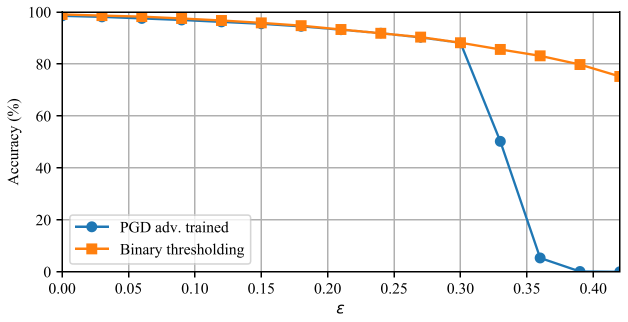


Fig. 10. Performance of our defense method against PGD adversaries of different strength on MNIST. The adversarially trained model is trained against $\epsilon = 0.3$. We observe that for ϵ smaller than 0.3, the adversarially trained model achieves equal accuracy to our defense method. But for ϵ more than 0.3, the performance of adversarial training drops sharply, while our defense remains stable.

ially trained model is trained for 70,000 iterations. The batch size is 128. The initial learning rate is 0.1, and we decrease it by 10× at the 40, 000 and 60, 000 iteation. For PGD, we use 7 steps of size 2, and a total $\epsilon = 8$.

All of our experiments run on an NVIDIA GeForce GTX 1080Ti. It is important to note that because of the random starting point in PGD, there is a randomness in mAP drop, i.e., the results of each attack are slightly different (generally ~1%). So for attack performance, we perform PGD three times and take the average as the final result.

4.2. Defense results

For MNIST, the defense results are shown in Table 1. The naturally trained model achieves 99.2% accuracy on clean test set, and 0% accuracy on adversarial test set at 23,000 FPS for training. The adversarially trained model achieves 98.5% accuracy on clean test set, and 90.3% accuracy on adversarial test set at 597 FPS for training. The training speed of standard training is roughly 40 times faster than adversarial training, and this corresponds to the PGD iteration number. For our defense method, we use the optimal threshold 0.427. The accuracy on clean test images is 99.0%, and the accuracy on adversarial test images is 91.2%. The accuracy on black-box attack is 95.6%, slightly lower than adversarial training. The overhead of our method is about 40× less than adversarial training, and our defense can be conducted online with any classifier without offline retraining process. The testing speed is roughly the same as adversarial training. We also find performing binary thresholding during evaluation help improve the robustness of adversarially trained model.

For fashion-MNIST, the defense results are shown in Table 2. The naturally trained model achieves 90.5% accuracy on clean test set, and 0% accuracy on adversarial test set. Performing binary thresholding during evaluation increases 36.9% accuracy of naturally trained model on adversarial test set. It also increases 2% accuracy of adversarially trained model on adversarial test set, and increases 5% accuracy on clean test set. We also train a new model on binarized clean images with the optimal threshold 0.325. The model achieves 86.2% accuracy on clean images and 47.0% on adversarial examples at 21,600 FPS for training. The overhead of our method is about 40$\times$ less than adversarial training.

For CIFAR-10, the defense results are shown in Table 3. We train a new model on binarized clean images with the optimal threshold 121. Our defense achieves 81.3% accuracy on clean images and 40.5% on adversarial examples at 278 FPS for training. Adversarial training achieves 87.3% accuracy on clean images and 47.2% on adversarial examples at 32 FPS for training. Our performance is 7% lower than adversarial training, but we decrease the computation time ninefold and achieve the best speed/ accuracy trade-off on CIFAR-10.

4.3. Convergence rate

We compare the convergence rate of our defense method and adversarial training on MNIST, as shown in Fig. 9. Our method converges much faster than adversarial training both on clean and adversarial test set. Moreover, the training time of each batch for adversarial training is 40$\times$ longer than our method.

4.4. Resistance for different values of ϵ

Since we perform binary thresholding, we speculate our method could resist different values of ϵ when ϵ is smaller than the threshold. On MNIST, we use 100 steps with a step size of $2.5 \cdot \epsilon/100$ to make sure that we could reach the boundary of the ϵ-ball from any starting point within it. We compare the robustness of our defense method with adversarial training. Note that the adversarially trained model is trained against attacks with the maximum perturbation of $\epsilon = 0.3$. And the values of ϵ is in the range of [0, 0.42] as in [31], since much larger ϵ will make the perturbation obvious enough to cause misclassification by humans.

As shown in Fig. 10, for ϵ smaller than 0.3, the adversarially trained model achieves equal accuracy to our defense method. But for ϵ more than 0.3, the performance of adversarial training drops sharply, while our defense remains stable.

5. Conclusion and future work

In this paper, we propose a fast, simple and strong defense method that achieves the best speed-accuracy trade-off. In MNIST, we visualize the feature maps of the adversarially and naturally trained model on clean image and its adversarial counterpart, and find that the adversarially trained model performs binary thresholding to gain robustness. Inspired by this, we perform a simple binary thresholding on input images before feeding them into the classifier to defend against PGD attack, and give a detailed description on how to obtain the optimal threshold in our defense regardless of dataset. We can obtain robustness just by binarizing the images during testing. For Fashion-MNIST and CIFAR-10, we need to use the binarized images to retrain a model and use the new model to obtain robustness.

There are also some shortcomings of our defense method. First, as shown in Table 3, the defense performance on CIFAR-10 can be improved. We also plan to perform our defense on ImageNet. Note that our defense method is based on L_∞ norm attack, and is not applicable to L_1 or L_2 norm attack. Since the L_1 and L_2 norm attack could significantly change the pixel value, our threshold filter will fail in this situation. So in the future, we will investigate more general defense method that can defend against more kinds of attack with better performance.

CRediT authorship contribution statement

Yutong Wang: Investigation, Conceptualization, Methodology, Software, Formal analysis, Visualization, Writing - original draft, Writing - review & editing. **Wenwen Zhang:** Conceptualization, Methodology, Validation, Formal analysis. **Tianyu Shen:** Conceptualization, Methodology, Validation, Formal analysis. **Hui Yu:** Resources, Conceptualization. **Fei-Yue Wang:** Resources, Supervision.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Y. Ming, X. Meng, C. Fan, H. Yu, Deep learning for monocular depth estimation: A review, Neurocomputing 438 (2021) 14–33, https://doi.org/10.1016/j.neucom.2020.12.089.
[2] G. Sokar, D.C. Mocanu, M. Pechenizkiy, Spacenet: Make free space for continual learning, Neurocomputing 439 (2021) 1–11, https://doi.org/10.1016/j.neucom.2021.01.078.
[3] P. Singh, P. Mazumder, M.A. Karim, V.P. Namboodiri, Calibrating feature maps for deep cnns, Neurocomputing 438 (2021) 235–247, https://doi.org/10.1016/j.neucom.2020.12.119.
[4] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Communications of the ACM 60 (6) (2017) 84–90, https://doi.org/10.1145/3065386.
[5] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1026–1034, https://doi.org/10.1109/ICCV.2015.123.
[6] X. Li, F.-Y. Wang, Parallel visual perception for intelligent driving: basic concept, framework and application, Journal of Image and Graphics 26 (1) (2021) 67–81, https://doi.org/10.11834/jig.200402.
[7] H. Zhang, X. Li, F.-Y. Wang, The basic framework and key algorithms of parallel vision, Journal of Image and Graphics 26 (1) (2021) 82–92, https://doi.org/10.11834/jig.200400.
[8] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L.D. Jackel, M. Monfort, U. Muller, J. Zhang, et al., End to end learning for self-driving cars, arXiv preprint arXiv:1604.07316 (2016)..
[9] J. Wu, R. Ji, J. Liu, M. Xu, J. Zheng, L. Shao, Q. Tian, Real-time semantic segmentation via sequential knowledge distillation, Neurocomputing 439 (2021) 134–145, https://doi.org/10.1016/j.neucom.2021.01.086.
[10] O.M. Parkhi, A. Vedaldi, A. Zisserman, Deep face recognition (2015)..
[11] F. Schroff, D. Kalenichenko, J. Philbin, Facenet, A unified embedding for face recognition and clustering, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 815–823, https://doi.org/10.1109/cvpr.2015.7298682.
[12] T. Shen, C. Gou, F.-Y. Wang, Z. He, W. Chen, Learning from adversarial medical images for x-ray breast mass segmentation, Computer Methods and Programs in Biomedicine 180 (2019), https://doi.org/10.1016/j.cmpb.2019.105012 105012.
[13] T. Shen, J. Wang, C. Gou, F.Y. Wang, Hierarchical fused model with deep learning and type-2 fuzzy learning for breast cancer diagnosis, IEEE Transactions on Fuzzy Systems 28 (12) (2020) 3204–3218, https://doi.org/10.1109/TFUZZ.2020.3013681.
[14] T. Shen, C. Gou, J. Wang, F.Y. Wang, Simultaneous segmentation and classification of mass region from mammograms using a mixed-supervision guided deep model, IEEE Signal Processing Letters 27 (2020) 196–200, https://doi.org/10.1109/LSP.2019.2963151.
[15] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, in: Proceedings of the International Conference on Learning Representations, 2014.
[16] S.-M. Moosavi-Dezfooli, A. Fawzi, P. Frossard, Deepfool: a simple and accurate method to fool deep neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2574–2582, https://doi.org/10.1109/CVPR.2016.282.
[17] N. Carlini, D. Wagner, Towards evaluating the robustness of neural networks, in, in: Proceedings of the IEEE Symposium on Security and Privacy, 2017, pp. 39–57, https://doi.org/10.1109/SP.2017.49.
[18] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z.B. Celik, A. Swami, The limitations of deep learning in adversarial settings, Proceedings of the IEEE European Symposium on Security and Privacy (2016) 372–387, https://doi.org/10.1109/EuroSP.2016.36.
[19] Y. Wang, K. Wang, Z. Zhu, F.-Y. Wang, Adversarial attacks on faster r-cnn object detector, Neurocomputing 382 (2020) 87–95, https://doi.org/10.1016/j.neucom.2019.11.051.

[20] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778, https://doi.org/10.1109/CVPR.2016.90.

[21] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014)..

[22] N. Papernot, P. McDaniel, X. Wu, S. Jha, A. Swami, Distillation as a defense to adversarial perturbations against deep neural networks, in: Proceedings of the IEEE Symposium on Security and Privacy, 2016, pp. 582–597, https://doi.org/10.1109/SP.2016.41.

[23] D. Hendrycks, K. Gimpel, Early methods for detecting adversarial images, arXiv preprint arXiv:1608.00530 (2016)..

[24] D. Meng, H. Chen, Magnet: a two-pronged defense against adversarial examples, in, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 135–147, https://doi.org/10.1145/3133956.3134057.

[25] C. Guo, M. Rana, M. Cisse, L. Van Der Maaten, Countering adversarial images using input transformations, in, in: Proceedings of the International Conference on Learning Representations, 2018.

[26] C. Xie, J. Wang, Z. Zhang, Z. Ren, A. Yuille, Mitigating adversarial effects through randomization, in: Proceedings of the International Conference on Learning Representations, 2018.

[27] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324, https://doi.org/10.1109/5.726791.

[28] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images (2009)..

[29] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, in: Proceedings of the International Conference on Learning Representations, 2015.

[30] A. Kurakin, I. Goodfellow, S. Bengio, Adversarial examples in the physical world, in: Proceedings of the International Conference on Learning Representations Workshop, 2017.

[31] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, in: Proceedings of the International Conference on Learning Representations, 2018.

[32] C. Xie, Y. Wu, L. v. d. Maaten, A.L. Yuille, K. He, Feature denoising for improving adversarial robustness, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 501–509. doi:10.1109/CVPR.2019.00059..

[33] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, International Journal of Computer Vision (2015), https://doi.org/10.1007/s11263-015-0816-y.

[34] A. Graese, A. Rozsa, T.E. Boult, Assessing threat of adversarial examples on deep neural networks, in: Proceedings of the IEEE International Conference on Machine Learning and Applications, 2016, pp. 69–74, https://doi.org/10.1109/ICMLA.2016.44.

[35] W. Xu, D. Evans, Y. Qi, Feature squeezing: Detecting adversarial examples in deep neural networks, in: Proceedings of the Network and Distributed System Security Symposium, 2018.

[36] G.K. Dziugaite, Z. Ghahramani, D.M. Roy, A study of the effect of jpg compression on adversarial images, arXiv preprint arXiv:1608.00853 (2016)..

[37] L.I. Rudin, S. Osher, E. Fatemi, Nonlinear total variation based noise removal algorithms, Physica D: Nonlinear Phenomena 60 (1–4) (1992) 259–268, https://doi.org/10.1016/0167-2789(92)90242-F.

[38] A.A. Efros, W.T. Freeman, Image quilting for texture synthesis and transfer, in: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, 2001, pp. 341–346.

[39] Y. Wang, F.-Y. Wang, Finding patterns in adversarial training, Chinese Automation Congress 2020 (2020) 4130–4134.

[40] M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, European Conference on Computer Vision (2014) 818–833, https://doi.org/10.1145/3065386.

[41] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, T. Darrell, Decaf: A deep convolutional activation feature for generic visual recognition, in: International Conference on Machine Learning, 2014, pp. 647–655..

[42] N. Papernot, P. McDaniel, I. Goodfellow, Transferability in machine learning: from phenomena to black-box attacks using adversarial samples, arXiv preprint arXiv:1605.07277 (2016)..

[43] Y. Liu, X. Chen, C. Liu, D. Song, Delving into transferable adversarial examples and black-box attacks, in: Proceedings of the International Conference on Learning Representations, 2017.

[44] F. Tramèr, N. Papernot, I. Goodfellow, D. Boneh, P. McDaniel, The space of transferable adversarial examples, arXiv preprint arXiv:1704.03453 (2017)..

[45] P. Samangouei, M. Kabkab, R. Chellappa, Defense-gan, Protecting classifiers against adversarial attacks using generative models, in: Proceedings of the International Conference on Learning Representations, 2018.

[46] Y. Song, T. Kim, S. Nowozin, S. Ermon, N. Kushman, Pixeldefend, Leveraging generative models to understand and defend against adversarial examples, in: Proceedings of the International Conference on Learning Representations, 2018.

[47] A. v. d. Oord, N. Kalchbrenner, K. Kavukcuoglu, Pixel recurrent neural networks, arXiv preprint arXiv:1601.06759 (2016)..

[48] T. Salimans, A. Karpathy, X. Chen, D.P. Kingma, Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications, arXiv preprint arXiv:1701.05517 (2017)..

[49] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems, vol. 27, 2014, pp. 2672–2680..

[50] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein generative adversarial networks, in: Proceedings of the International Conference on Machine Learning, 2017, pp. 214–223.

[51] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, arXiv preprint arXiv:1708.07747 (2017)..

[52] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, et al., Tensorflow: Large-scale machine learning on heterogeneous distributed systems, arXiv preprint arXiv:1603.04467 (2016)..

[53] S. Zagoruyko, N. Komodakis, Wide residual networks, arXiv preprint arXiv:1605.07146 (2016)..

Yutong Wang received her bachelor degree from Harbin Engineering University in 2016. She is currently a Ph.D. student in University of Chinese Academy of Sciences as well as The State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences. Her research interests include computer vision and object detection.

Wenwen Zhang received his bachelor degree in software engineering from Xi'an Jiaotong University, Xi'an, China, in 2014. He is currently a Ph.D. student in the School of Software Engineering, Xi'an Jiaotong University as well as the State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences. His research interests include computer vision and deep learning.

Tianyu Shen received the bachelor of engineering and bachelor of management degree from the Xi'an Jiaotong University, Xi'an, China, in 2016. She is currently a Ph.D. candidate at the State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences as well as University of Chinese Academy of Sciences. Her research interests include computer vision and machine learning.

Hui Yu (SM'15) is a Professor with the University of Portsmouth, UK. His research interests include methods and practical development in vision, machine learning and AI with applications to human-machine interaction, virtual and augmented reality, robotics and geometric processing of facial expression. He serves as an Associate Editor of IEEE Transactions on Human-Machine Systems and IEEE/CAA Journal of Automatica Sinica.

Fei-Yue Wang (S'87-M'89-SM'94-F'03) received his Ph. D. degree in computer and systems engineering from the Rensselaer Polytechnic Institute, Troy, NY, USA, in 1990. He joined The University of Arizona in 1990 and became a Professor and the Director of the Robotics and Automation Laboratory and the Program in Advanced Research for Complex Systems. In 1999, he founded the Intelligent Control and Systems Engineering Center at the Institute of Automation, Chinese Academy of Sciences (CAS), Beijing, China, under the support of the Outstanding Chinese Talents Program from the State Planning Council, and in 2002, was appointed as the Director of the Key Laboratory of Complex Systems and Intelligence Science, CAS. In 2011, he became the State Specially Appointed Expert and the Director of the State Key Laboratory for Management and Control of Complex Systems. His current research focuses on methods and applications for parallel intelligence, social computing, and knowledge automation. He is a fellow of INCOSE, IFAC, ASME, and AAAS. In 2007, he received the National Prize in Natural Sciences of China and became an Outstanding Scientist of ACM for his work in intelligent control and social computing. He received the IEEE ITS Outstanding Application and Research Awards in 2009 and 2011, respectively. In 2014, he received the IEEE SMC Society Norbert Wiener Award. Since 1997, he has been serving as the General or Program Chair of over 30 IEEE, INFORMS, IFAC, ACM, and ASME conferences. He was the President of the IEEE ITS Society from 2005 to 2007, the Chinese Association for Science and Technology, USA, in 2005, the American Zhu Kezhen Education Foundation from 2007 to 2008, the Vice President of the ACM China Council from 2010 to 2011, the Vice President and the Secretary General of the Chinese Association of Automation from 2008-2018. He was the Founding Editor-in-Chief (EiC) of the International Journal of Intelligent Control and Systems from 1995 to 2000, the IEEE ITS Magazine from 2006 to 2007, the IEEE/CAA JOURNAL OF AUTOMATICA SINICA from 2014-2017, and the China's Journal of Command and Control from 2015-2020. He was the EiC of the IEEE Intelligent Systems from 2009 to 2012, the IEEE TRANSACTIONS ON Intelligent Transportation Systems from 2009 to 2016, and is the EiC of the IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SYSTEMS since 2017, and the Founding EiC of China's Journal of Intelligent Science and Technology since 2019. Currently, he is the President of CAA's Supervision Council, IEEE Council on RFID, and Vice President of IEEE Systems, Man, and Cybernetics Society.