DEFENDING VISUAL ADVERSARIAL EXAMPLES WITH SMOOTHOUT REGULARIZATION

By

WEITIAN LI

A thesis submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Master of Science

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Bo Yuan

And approved by

_____

_____

_____

New Brunswick, New Jersey

October 2019

ABSTRACT OF THE THESIS

Defending Visual Adversarial Examples With Smoothout Regularization

by Weitian Li

Thesis Director: Prof. Bo Yuan

In the past decades with unexpected and rapid development of computer vision, tremendous computer vision applications like face recognition, image recognition, object detection and so on. They present their powerful abilities to made life so convenient for humans. In the trend of computer vision, deep neural networks (DNNs) occupies a very essential role. Because relative applications are deploying in many critical fields such as autonomous car, authentication and so on. However, there exist many adversarial attacks that can result in huge model performance degradation. Deploying a robust and reliable DNN is becoming a crucial and necessary step for various applications. In this work, we introduce SmoothBlock, a novel regularization method to improve the model robustness against adversarial attacks. It can be directly utilized as a defense mechanism in inference phase to protect the pre-trained model. Besides, the proposed SmoothBlock can also be applied in both training and adversarial training to further improve the robustness against various adversarial attacks. Furthermore, we apply the proposed SmoothBlock with a self-ensemble method to improve the robustness of the system. We conduct extensive trials and detailed analysis on CIFAR-10 using Resnet20 model. Results show that the model robustness can be significantly improved by our method against FGSM, PGD and C&W L2 attacks under white-box scenarios.

ACKNOWLEDGEMENTS

I would first like to thank my advisor Professor Bo yuan for his patient and careful guidance on the supervision of my master thesis.

I would also show my great thanks to Yi Xie and Siyu Liao for offering many useful advise and help in the experiments and my thesis. It's proud and happy to be your collaborator. In particular, I want to thank Professor Sheng Wei, Professor Predrag Spasojevic for serving in my master thesis committee. Also, I would like to thank Zhongze Tang's template.

Last but not least, I would like to thank my friends, my girlfriend Hui Che and my family for supporting and inspiring me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Deep neural networks (DNNs) have achieved state-of-the-art results in many tasks, such as object recognition [1], natural language processing [2], semantic segmentation [3], autonomous vehicle [4], image recognition [5] and so on. While DNNs are becoming a powerful model and are deployed in real world platforms, related security problems have drawn people's attention recently in those security-critical scenarios.

1.1 Image Recognition

Image recognition refers to the technique of using a machine to process, analyze, and understand images to identify targets and objects in various modes. For a typical image recognition system, it accepts images and outputs classification of the images. The traditional process of image recognition is image collection, image prepossessing, feature extraction and image classification. Nowadays, DNN plays a significant role of it. One of the most popular DNN architecture is LeNet-5 [6],which can input the image of hand crafted number and outputs the classification of the image. The outline of LeNet-5 is depicted in Figure 1.1 and it is general flow diagram of image classification of DNN.

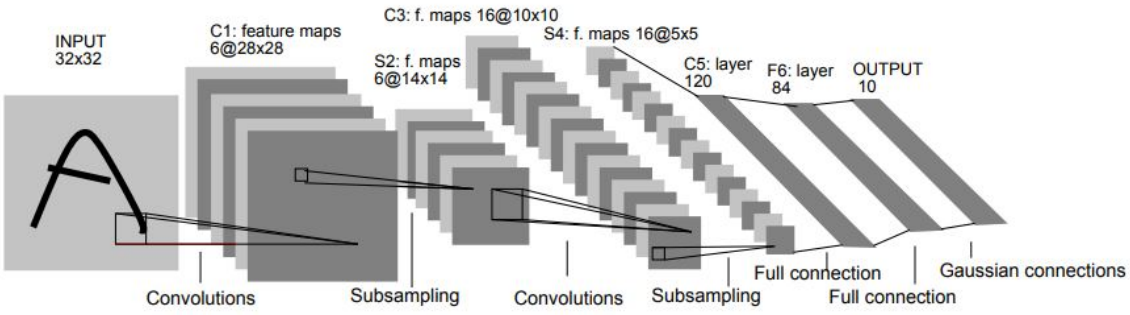
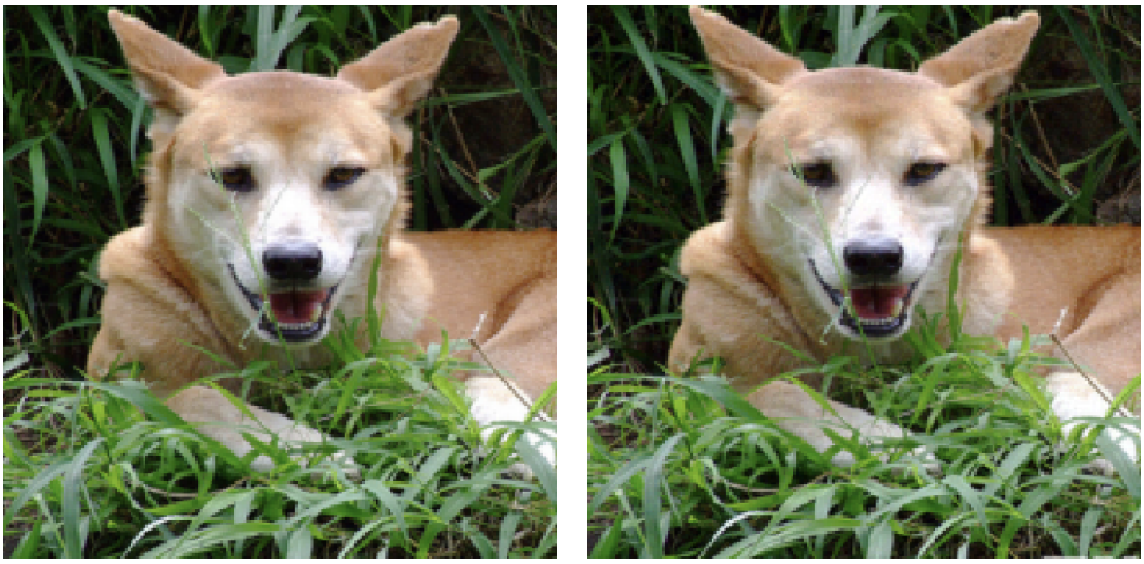


Figure 1.1: The outline of LeNet-5 [6]



(a) Classified as Dingo Dog (b) Classified as Mosquito Net

Figure 1.2: An Adversarial Example.

1.2 Adversarial Example

Recent study [7] firstly shows that DNN classifiers are vulnerable to data with small and special designed perturbations. They apply imperceptible perturbation to test images and successfully change DNN classification results, which is hard to distinguish for human. These perturbed input are called *adversarial examples*. As shown in Figure 1.2, the left image from ImageNet dataset [8] is classified as Dingo dogs by a given DNN. With small and special designed perturbation, the modified image on the right looks almost the same but now is classified as mosquito net.

What's more, recent study revealed that some adversarial examples have transfer-ability and are able to damage other DNNs [9]. That is to say, these perturbed inputs generated from given DNN can attack other DNNs even though they may have different architectures. Such transfer-ability is easy to find in *non-targeted* adversarial examples but not in *targeted* adversarial examples. The former means DNN models give wrong results for input while the latter further requires that DNN models misclassify the data to a designated class.

In general, classifier attacks can be categorized in three different classes [10], including

evasion attack, poisoning attack and exploratory attack. For the evasion attack, it only modifies testing data. But poisoning attack is about ”poisoning” training data to compromise a classifier performance. The exploratory attack aims at detecting underlying classifier algorithm and even the training data pattern.

Note that this thesis mainly focuses on defending against evasion attack. In this case, attacking methods greatly depend on attackers' knowledge about given DNN. Thus, evasion attack can be further categorized into white box and black box attack. White box attack means attackers know the DNN architecture and even training data patterns. Instead, black box attack means they are lack of such information. More specifically, it is called non-adaptive black box attack if they know the training data distribution with which a similar DNN can be trained. Due to the transfer-ability, adversarial examples can be generated with this self-trained DNN under white box attack and then used to attack the unknown DNN. Adaptive black box attack means attackers can use the target model but don't have any related information.

A natural and straightforward defense method is training DNN classifiers together with adversarial examples, namely, *adversarial training*. This is effective for specific adversarial attacks [11] but it may be not practical due to the extra training time to adjust existing DNNs [12].

Inspired by the effectivenss of DropBlock [13], we find that smoothing feature maps with less variance works better than increasing mean and variance when testing over both clean and adversarial examples. Thus, instead of using dropping and scaling, we replace dropped blocks with the overall average feature maps to smooth the activation. Figure 1.3 shows feature maps of an adversarial example when applying DropBlock and SmoothBlock. Feature maps after DropBlock have large variance since values are distributed either close to 1 or close to 0. Instead, SmoothBlock doesn't scale up and the whole feature maps look smooth.

In this work, we propose a flexible defense algorithm called SmoothBlock. It can be

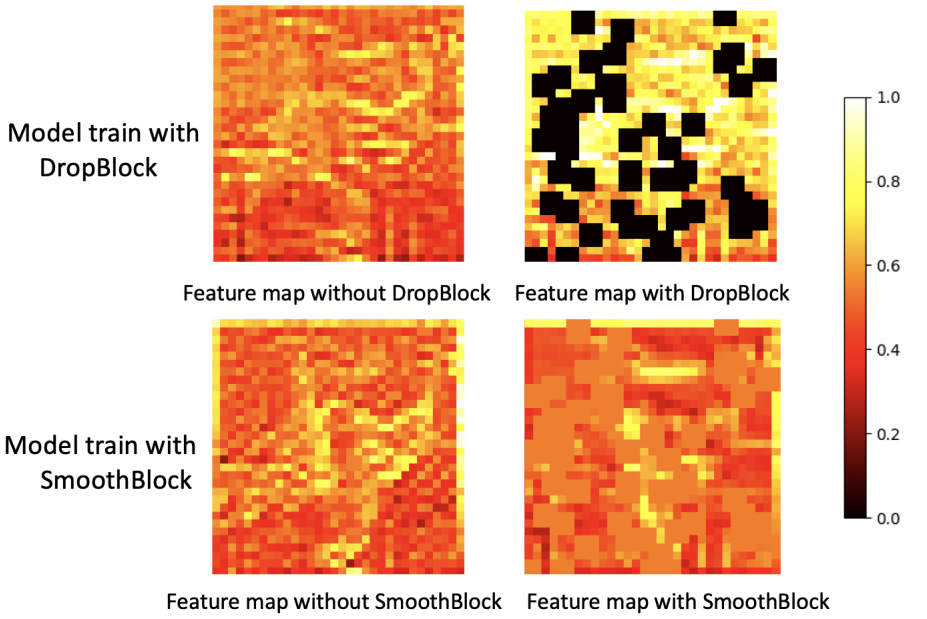


Figure 1.3: Visualization of DropBlock and SmoothBlock.

directly applied to an existing DNN model at testing phase and improve the robustness against adversarial attacks. When adversarial training is feasible, it can also serve as a regularization method during training. Moreover, we combine SmoothBlock with adversarial training and apply a self-ensemble method [14] to further improve the model robustness. Contributions of this work are summarized as below:

- We propose a flexible defense algorithm, SmoothBlock, that is directly deploy-able to protect pre-trained DNN models against adversarial examples while able to serve as regularization for training.

- While SmoothBlock works well with adversarial training, the self-ensemble algorithm can further improve the model robustness when computation cost is not in concern.

- Extensive experiments are conducted to demonstrate the effectiveness of the proposed method and results show that SmoothBlock works better than DropBlock under different adversarial settings.

1.3 Thesis Structure

Our proposed regularized defense against adversarial examples method for deep neural network allows the model to gain higher robustness when facing adversarial attacks at inference phase, while it can be additional defense mechanism on already pre-trained model and directly applied on model in training and adversarial training phase.

The structure of this thesis is as follows:

Chapter 1 introduces image recognition, adversarial examples and our findings of this thesis.

Chapter 2 discusses related works on adversarial attack, defense methods, the relative concepts of our topic and our inspiration source.

Chapter 3 presents the tools and platform for crafting practical image recognition model, implementation for SmoothBlock, the image dataset and adversarial examples for following experiment.

Chapter 4 descries our proposed method, explanation, mechanism and relative experiment details.

Chapter 5 presents the future work of this thesis and improvement.

Chapter 6 draws the conclusion.

CHAPTER 2
RELATED WORK

The security problem of DNN models is firstly proposed by [7], and it shows that the smoothness assumption of DNN model doesn't hold. In this case, the smoothness means that introducing imperceptibly perturbations to the image should not change its category. The existence of adversarial examples of DNN raises questions on the generalization and discontinuity of nonlinear input-output mappings learned by DNN models. In this section, we introduce basic concepts on DNNs and adversarial attacks used in our experiments. We focus on DNN image classifiers under both targeted and non-targeted attacks.

2.1 Adversarial Attack

Linearity inside the design of DNNs like ReLU [15] turns out exploitable [12]. A fast gradient sign method (FGSM) is proposed to quickly generate adversarial examples for given DNNs. Small perturbation are generated based on signs of input image's gradients. Moreover, iterative FGSM with clipping [16] generates stronger adversarial examples, which, however, turns out to be less transferable.

Image gradients can also be utilized to help determine which input pixel perturbation affects most to the given DNN. In [17], a salient map is computed to model pixel effect on classification results. For a target class, it modifies the most effective pixel from salient map each time and repeatedly does so till successfully changing the classification result.

[18] works on finding small perturbations with different distance metrics, including l_0, l_2 and l_∞ norm. Given the difficulty of high non-linearlity of DNNs, the objective function is re-formulated so as to be optimized easily and it turns out to be a powerful attack method.

On the other hand, [19] firstly practically presents an adaptive black box attack method. They are limited to query target DNN with input images. Firstly, a local DNN is trained

with query results from target DNN, and it is also used to generate Jocobian-based synthetic inputs to approximate the target DNN decision boundaries. Then they craft adversarial examples on this local DNN using the method of [12] and [17]. As a result, this method successfully attacks online DNN based service in real world.

2.2 Adversarial Defense

In order to smooth the DNN models, [20] learns from the distillation technique [21] that is proposed to produce a compact DNN architecture by training a small network with soft labels generated from a large network. Their intuition comes from the fact that distillation may help DNN generalize well so as to defend against perturbed input. In this case, soft labels come from the pre-trained DNN and are then used to help train a distilled network.

Inspired from game theory, stochastic activation pruning (SAP) is proposed as an adversarial defense strategy [22]. Activation units at each layer are randomly pruned with probabilities proportional to their absolute values. Rest units will be scaled up to keep the overall magnitude.

On the other hand, it is natural to associate testing image with training image and classify based their combination [23]. Given a test image, its peer training images are found by the cosine distance of their feature maps. The output feature map is computed as the element-wise weighted results of these images. This method turns out effective for both targeted and non-targeted white box attacks.

Unlike adversarial training, [24] directly added Gaussian noise as a regularization method during training which is called Parametric-Noise-Injection (PNI). In both training and testing phase, such noise can be added at inputs, activation units and even weight parameters. Surprisingly, this method improves DNN performance over both clean data and adversarial data.

2.3 Deep Neural Network

Let $\mathbf{X} \in \mathbb{R}^{c \times w \times h}$ denote an input image, where c, w and h are the number of channels, image width and image height, respectively. DNN is a discriminative model $L(\boldsymbol{\theta}; \mathbf{X}, y) = p(y|\mathbf{X}, \boldsymbol{\theta})$ that classifies input image. $\boldsymbol{\theta}$ are weight parameters of DNN model and y is a certain image class. The class with the maximum probability is chosen as the classification result of DNN. Adversarial examples mean that there exist small perturbation $\boldsymbol{\delta} < \epsilon$ such that $\mathbf{X} + \boldsymbol{\delta}$ is imperceptible but $p(y|\mathbf{X} + \boldsymbol{\delta}, \theta) \neq p(y|\mathbf{X}, \theta)$. Therefore, the DNN classifier is attacked and its accuracy gets much lower.

2.4 FGSM

It[12] generates adversarial examples using $\nabla L(\mathbf{X})$, which only takes one step to compute. The perturbation is set as the sign of $\nabla L(\mathbf{X})$ and it is bounded under the infinity norm, i.e., $||\boldsymbol{\delta}||_\infty < \epsilon$ for some $\epsilon \in \mathbb{R}$. A nice property of infinity norm is that it won't scale up with the input dimension. Then all pixels are modified with the perturbation:

$$\mathbf{X}' = \mathbf{X} + \epsilon \cdot \text{sign}(\nabla L(\mathbf{X})), \tag{2.1}$$

where ϵ implies the strength of add-on perturbations and sign$(\cdot)$ is an element-wise sign function. $\mathbf{X}'$ is the adversarial example. Moreover, this method can be incorporated into the loss function as a regularization term for adversarial training.

2.5 C&W

C&W [18] attack explores different perturbations under l_0, l_2 and l_∞ norms. It generates adversarial examples by solving following problem:

$$||\boldsymbol{\delta}|| + c \cdot L(\mathbf{X} + \boldsymbol{\delta}),\ 0 \leq \mathbf{X} + \boldsymbol{\delta} \leq 1 \tag{2.2}$$

where $||\cdot||$ can be different norms and c is a constant scalar. It is suggested to choose small c such that $L(\mathbf{X}+\boldsymbol{\delta}) \leq 0$. The constraints on perturbed input falling into range 0 to 1 can be solved by different approaches, including projected gradient descent, clipping gradient descent and changes of variables. Then DNN optimizers are applicable to the re-formulated problem and Adam [25] turns out to converge much faster than others.

2.6 PGD

Iteratively applying FGSM can be regarded as project gradient descent (PGD) [11] process. First application of FGSM is the updated perturbed input and the rest FGSMs work like a projection function. This is formulated as follows:

$$\mathbf{X}_0 = \mathbf{X},\ \mathbf{X}_{t+1} = \text{clip}_{\epsilon}(\mathbf{X}_t + \alpha \cdot \text{sign}(\nabla L(\mathbf{X}_t))), \tag{2.3}$$

where α is a constant scalar and the function clip$(\cdot)$ ensures that each pixel is valid and also bounded by $\pm\epsilon$ as defined in [16]. A valid pixel value means that its value is between 0 and 255 in terms of eight-bit color depth. The iteration number is chosen to reach the ϵ-ball boundary while saving the experiment computation cost.

2.7 DropBlock

Images has spatial locality as well as its feature maps at outputs of each layer of a DNN [13]. Randomly dropping single activation unit is not enough to remove semantic information since neighboring units are highly related and may compensate those dropped units. As a result, dropping a block of activation units can be much more effective in training DNN model and improving classification accuracy. This method is called DropBlock. After dropping, rest units will be scaled up to maintain the overall activation magnitude. However, DropBlock is proposed to be a regularization method to generalize dropout [26] to convolution layers. It is unknown whether it can improve the DNN robustness and help

defend adversarial examples or not.

CHAPTER 3
TOOLS AND PLATFORM

3.1 Pytorch

In our work, we use Pytorch as our programming Python framework. It is an open source Python library that allows users to have tensor computation with strong GPU acceleration and deep neural networks built on a tape-based autograd system [27]. The Pytorch version we used is 0.4.1.

3.2 CIFAR-10

In this work, we apply our SmoothBlock and following experiments on Canadian Institute For Advanced Research 10 dataset (CIFAR-10) [28]. It was collected by Canadian Institute For Advanced Research, which is one of the most popular and worldwide used datasets for machine learning and computer vision research. In CIFAR-10, there is 60000 32x32 pixels color images in 10 class, with 6000 images per class [28]. 50000 of them are training images and another 10000 images are test images. In this dataset, there are five batches for training and one batch for testing. Each batch contains 10000 images. Here is the sample of CIFAR-10 in Figure 3.1.

3.3 Adversarial Robustness Toolbox

As mentioned below, there are many adversarial attack methods such as FGSM, C&W, PGD and so on. They are main and powerful adversarial attack methods used for evaluating robustness of DNN. In this work, we use Adversarial Robustness Toolbox, an open source Python library from IBM company. It is developed for rapid crafting adversarial examples and analysis of attacks and defense methods for machine learning models [29].

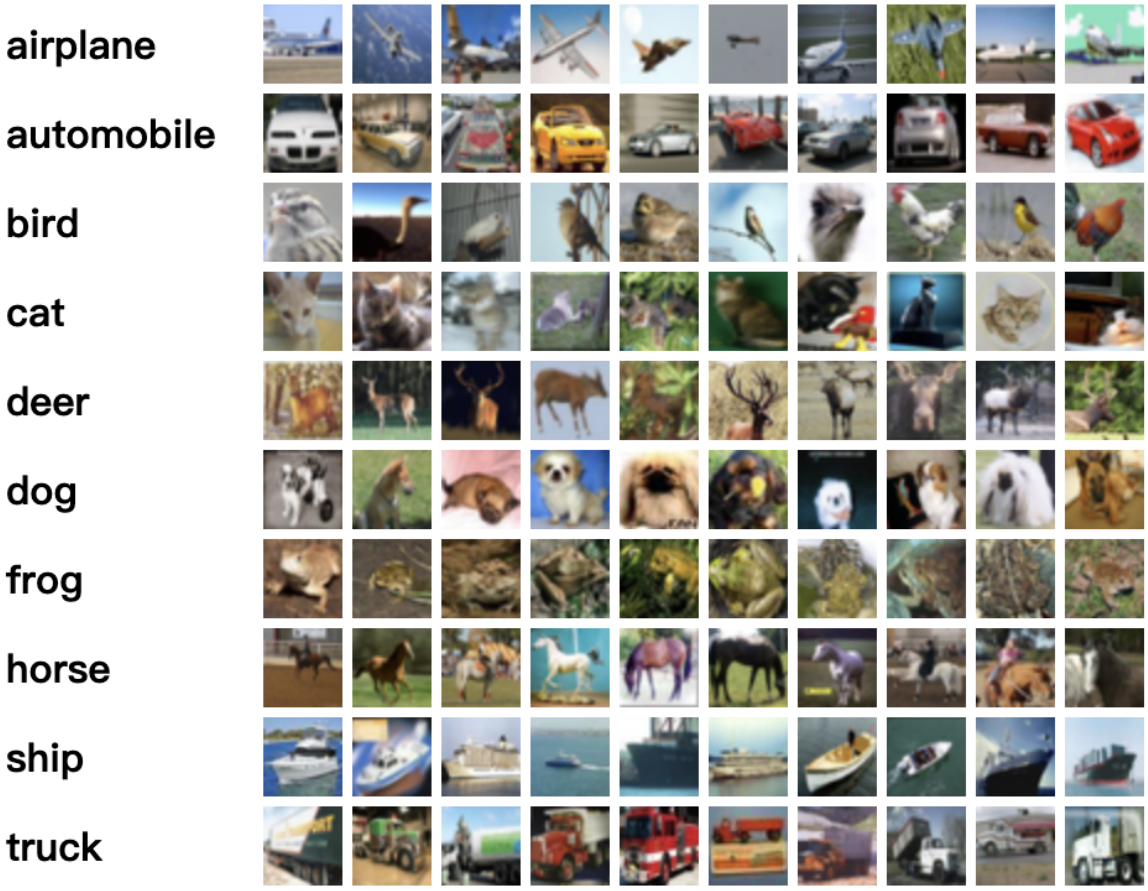


Figure 3.1: The sample of CIFAR-10 dataset. [28]

The Adversarial Robustness Toolbox we used is 0.4.0.

3.4 Resnet20

ResNet (Residual Neural Network) was proposed by Kaiming He et al. [30] (Microsoft Research Institute). Through the use of ResNet Unit, the 152-layer neural network was successfully trained and won the championship in ILSVRC2015. The error rate on top5 was 3.57%. At the same time, the parameter quantity is lower than VGGNet, and the performance is very outstanding [30]. The structure of ResNet can accelerate the training of neural networks very quickly, and the accuracy of the model is also greatly improved. At the same time, ResNet is very popular and can even be used directly in the Inception-Net network. There are many variant such as ResNet20, ResNet32, ResNet44, ResNet56, ResNet110, ResNet1202 and so on. In this work, we choose ResNet20 as our exam network for SmoothBlock. Here is the sample structure of ResNet in Figure 3.2.

3.5 SmoothBlock

In this work, we argue that dropping blocks of activation units and scaling up rest ones will increase mean and variance for rest activation units. This is limited in improving the DNN robustness. Instead, using the average of feature maps can help defend against adversarial examples, which will smooth these perturbed feature maps with the same mean value but less variance.

We present our algorithm in psedu-code as in Algorithm 1. Input contains layer activation $\mathbf{X}$, block size b and probability setting p which is used to randomly select block centers C. p determines how likely to keep the activation unit so it is actually keeping probability as in [13]. The neighboring features maps is defined by a function $N(\cdot)$ which returns neighbors as decided by b. In this work, these are square blocks all with the same height and width. Set B contains all activation units within blocks generated based on center points in C. For each activation unit in B, it is replaced either by the overall average m.

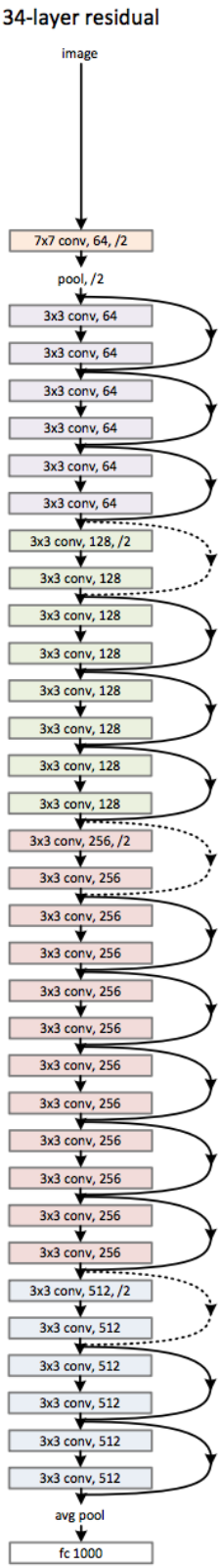


Figure 3.2: The sample structure of ResNet. [30]

Rest activation units are kept the same as original.

Algorithm 1 SmoothBlock Layer

Input: layer activation $\mathbf{X}$, size b, keeping probability p
Output: layer activation $\mathbf{Y}$
initialize empty set $C = \{\}$
for $x_i \in \mathbf{X}$ **do**
    uniformly sample $\gamma \in [0, 1]$
    if $\gamma < \frac{1-p}{b^2}$ **then**
        $C = C \cup \{x_i\}$
initialize empty set $B = \{\}$
for $c_i \in C$ **do**
    for $x_i \in \mathbf{X}$ **do**
        if $x_i \in N(c_i)$ **then** $B = B \cup \{x_i\}$
$m \leftarrow \text{mean}(\mathbf{X})$
initialize $\mathbf{Y}$ the same dimension as $\mathbf{X}$
for $x_i \in \mathbf{X}$ **do**
    if $x_i \in B$ **then**
        $y_i \leftarrow m$
    else
        $y_i \leftarrow x_i$
return $\mathbf{Y}$

Since p stands for keeping probability, $1 - p$ will be the probability to take the point as the dropping center. We scale this probability down with the block size b because the dropping center will drop together with b^2 number of neighboring units.

Besides, the above neighboring function $N(\cdot)$ can be implemented as creating square blocks with certain height and width that are centered by the given point. All activation units inside the block will be "smoothed" with the average value m.

3.6 Comparison with Dropblock

In this section, we discuss the differences between SmoothBlock and Dropblock, which are concluded in following aspects:

- SmoothBlock "smoothed" dropped units with overall average value rather than zeros.
- SmoothBlock doesn't scale up rest activation units.

- SmoothBlock can be directly applied to testing phase without fine-tune but it can also serve as a regularization method for training.

- SmoothBlock makes DNN model more robust than DropBlock in practice.

However, SmoothBlock maintains the same feature map magnitude as DropBlock while with less variance. Let $\mathbf{X}'$ be activation of perturbed input. Table 3.1 presents feature maps magnitude in terms of their Frobenius norm. Dropped units in DropBlock method have zero magnitude since they are set as zero and rest units are scaled up by constant c which is chosen as $\frac{1}{p}$ [13]. SmoothBlock keeps the same magnitude for both dropped and rest units. As a result, the overall magnitude is maintained in both methods. Mean and variance of DropBlock are both scaled up in terms of rest units. Variance of SmoothBlock is smaller since many blocks of units are now with same values.

Table 3.1: Feature Maps Magnitude.

	Dropped Units	Rest Units
DropBlock	0	$\|\|\mathbf{X}'\|\| * p * c$
SmoothBlock	$\|\|\mathbf{X}'\|\| * (1-p)$	$\|\|\mathbf{X}'\|\| * p$

CHAPTER 4
EXPERIMENTS

4.1 Experiment setup

We evaluate the robustness of SmoothBlock against adversarial examples on CIFAR-10 [31] dataset using ResNet20 [30] model is evaluation. This baseline model is trained for 200 epochs using Adam optimizer, with batch size of 128, and initial learning rate as 0.001 that decreases at epoch 80 and 180 with a factor of 0.1 and 0.5, respectively. The model can achieve 90.03% accuracy on clean test data.

We defend against FGSM attack, PGD attack as well as C&W l_2 attack under white-box scenario. For FGSM attack, the adversarial perturbation magnitude is set as $\epsilon = \{1, 2, 4, 8, 16, 32\}$ with respect to [0-255] pixel scale. Configuration for non-targeted PGD is the same as [11], i.e., $\epsilon = 8$ and 7 iterative steps with size $\alpha = 2$. As for C&W, we perform l_2 norm based target attack on first 1000 images of CIFAR-10 test data, using 10 steps of binary search for constant c. The final attack success rate (ASR) for C&W can reach at 100%, ASR implies the percentage of test data being mis-classified to the target wrong label.

The proposed SmoothBlock is systematically compared with DropBlock [13]. We for the first time apply DropBlock in test phase to exam the model robustness against adversarial examples. It is also worth mentioning that the idea of DropBlock is similar with another competing defending method SAP [22] when the block size is set as 1. Both SmoothBlock and DropBlock use shared dropped blocks across feature channels in our experiments.

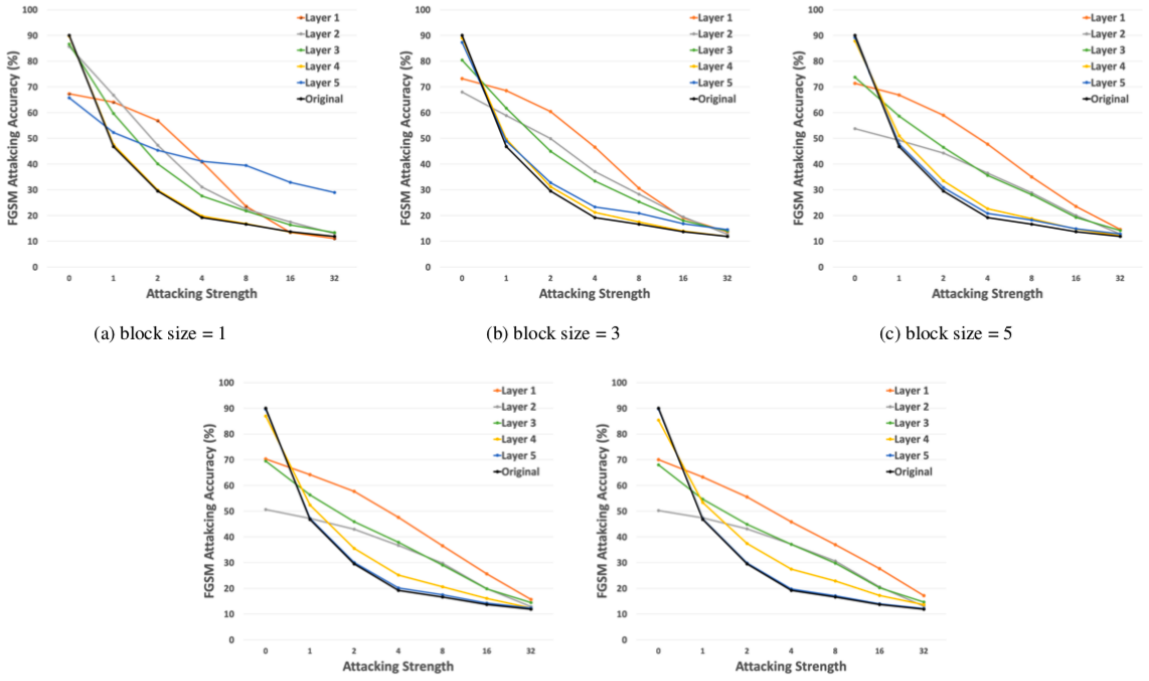
4.2 Post-Hoc Application to Pre-trained Model

We directly apply our proposed SmoothBlock on pre-trained ResNet20 model against adversarial attacks without any fine tuning. ResNet20 consists of a convolutional layer with

a batch normalization, followed by 3 residual blocks, an average pooling, and a fully-connected output layer. Extensive experiments are conducting to figure out the impact of block size b, applied location of SmoothBlock as well as keeping probability p.

Comparison results of test accuracy on ResNet20 model against FGSM attack are depicted in Fig. 4.3. The details of comprehensive experimental results are showed in Fig.4.1. We inject the defensive DropBlock/SmoothBlock regularization in ResNet20 at 5 different positions: layer 1 denotes applying DropBlock/SmoothBlock after the activation of the first convolutional layer, layer 2-4 denote applying DropBlock/SmoothBlock after each residual layer respectively, and layer 5 denotes placing DropBlock/SmoothBlock after the last pooling layer. Results for different block size b settings with top performance are shown in Fig. 4.3a and 4.3b. For both DropBlock and SmoothBlock, the model robustness has a larger improvement for layer 1 compared to others. Corresponding accuracy results are extracted to be shown in Fig. 4.3c for a detailed comparison. It clearly shows that using our proposed SmoothBlock in testing phase could make the model much more robust than original baseline and DropBlock. However, utilizing SmoothBlock could also eliminate legitimate spatial features, since the original model is trained without this regularization term, thus the test accuracy on clean data is deducted from 90.03% to 77.34%as shown in Fig. 4.3c.

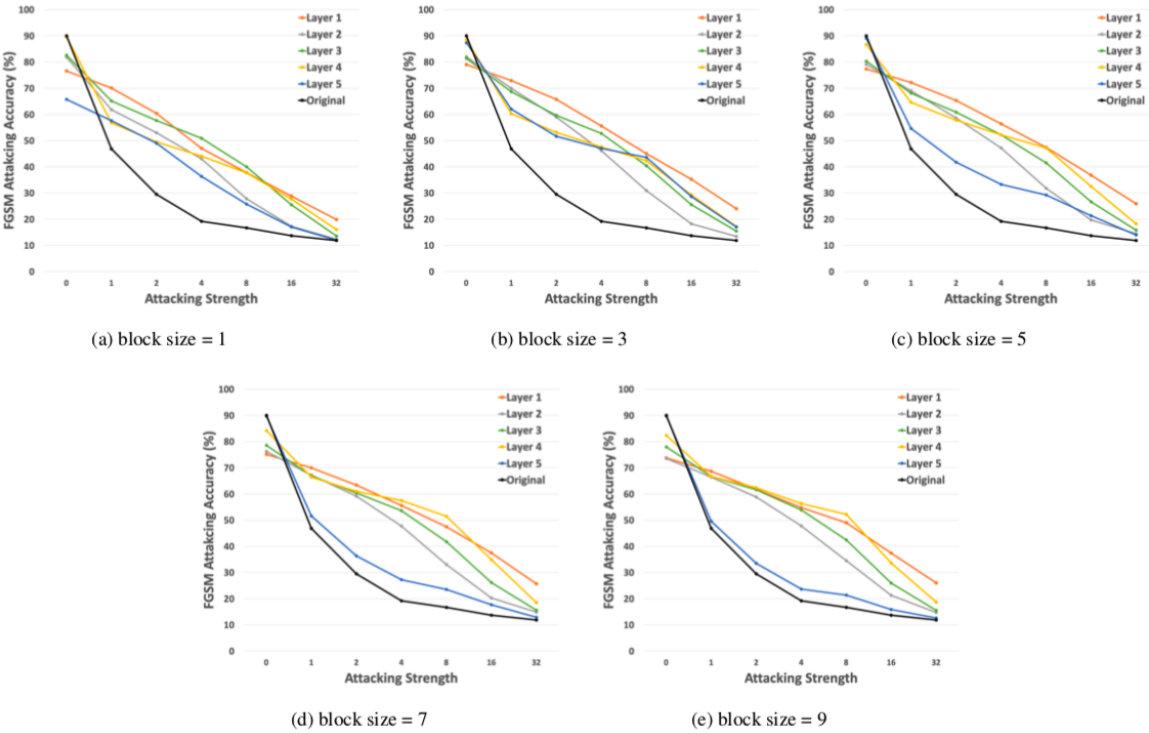
We also evaluate defending against PGD attack and C&W attack that are powerful non-targeted and targeted attack respectively. We have tried various configurations for block size b and applied location for both SmoothBlock and DropBlock. The details of extensive experiment are depicted in Fig.4.2. Best evaluation results are shown in Table. 4.1. It suggests that when applying defensive regularization, resistance of the model has been improved against PGD attack. Particularly, the proposed SmoothBlock allows the model to achieve much better performance than DropBlock. In addition, applying both SmoothBlock and DropBlock terms can effectively protect the model against C&W attack while getting significantly lower attack success rate (ASR) from 100% to 7.87%.



(a) block size = 1 (b) block size = 3 (c) block size = 5

(d) block size = 7 (e) block size = 9

(a) Comparison of CIFAR10 test accuracy plots on pre-trained ResNet20 model against FGSM attack when testing DropBlock is located in different layers with different block sizes.

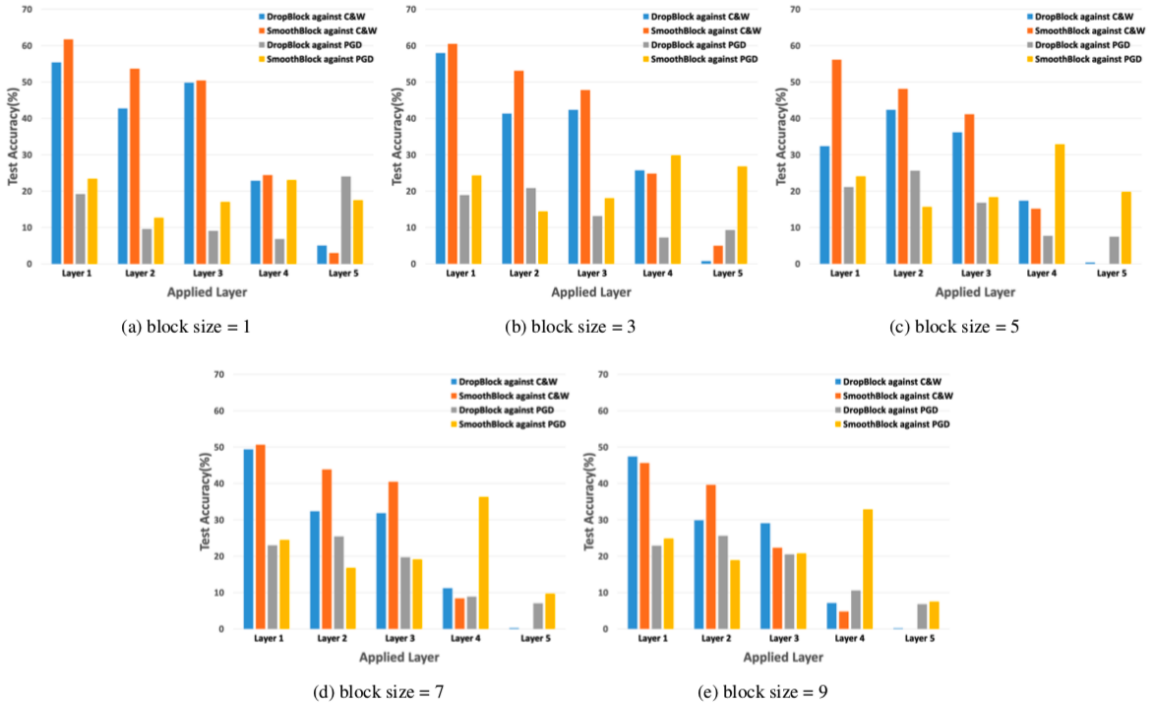


(a) block size = 1 (b) block size = 3 (c) block size = 5

(d) block size = 7 (e) block size = 9

(b) Comparison of CIFAR10 test accuracy plots on pre-trained ResNet20 model against FGSM attack when testing SmoothBlock is located in different layers with different block sizes.

Figure 4.1: Detailed Comparison of CIFAR10 test accuracy plots on pre-trained ResNet20 model against FGSM attack of DropBlock and SmoothBlock



(a) block size = 1 (b) block size = 3 (c) block size = 5

(d) block size = 7 (e) block size = 9

Figure 4.2: Comparison of CIFAR10 test accuracy on pre-trained ResNet20 model against C&W and PGD attack when testing DropBlock or SmoothBlock is located in different layers with different sizes.

Table 4.1: Robustness evaluation on pre-trained ResNet20 against PGD and C&W attack (%).

	PGD Acc	C&W ASR	C&W Acc
Raw	6.44	100	0.00
DropBlock	25.6	**7.82**	57.95
SmoothBlock	**36.31**	7.87	**60.5**

4.3 SmoothBlock Training on Clean Data

As investigated in 4.2, in most cases, SmoothBlock as a regularization term is embedded at layer 1 in ResNet20 with block size $b = 3$ and keeping probability $p = 0.7$, top defense performance will be reached. Therefore, we fix such configurations for other experiments. By training model with SmoothBlock using the same hyper-parameters mentioned in Section 4.1, the classification performance on clean data are comparable with original pre-trained model as shown in Figure 4.4. Besides, we also list the evaluation accuracy under FGSM attack on regularized models without any defensive method during inference, and results

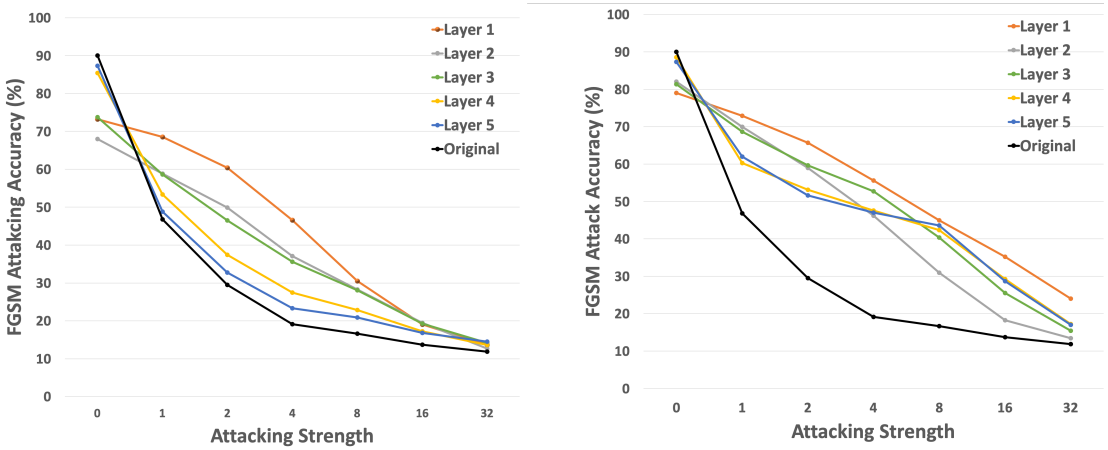


Figure 4.3: Comparison of test accuracy plots on pre-trained ResNet20 model against FGSM attack when testing DropBlock or SmoothBlock is located in different layers. (a) Test with DropBlock. (b) Test with SmoothBlock. (c) Comparison of DropBlock and SmoothBlock.

	Original Acc	FGSM 1	FGSM 2	FGSM 4	FGSM 8	FGSM 16	FGSM 32
Original Pre-trained	**90.03**	46.80	29.47	19.15	16.64	13.68	11.84
Train with DropBlock	89.19	**54.39**	34.02	23.11	18.14	13.03	12.85
Train with SmoothBlock	89.34	52.67	**34.32**	**25.06**	**19.46**	**15.83**	**12.99**

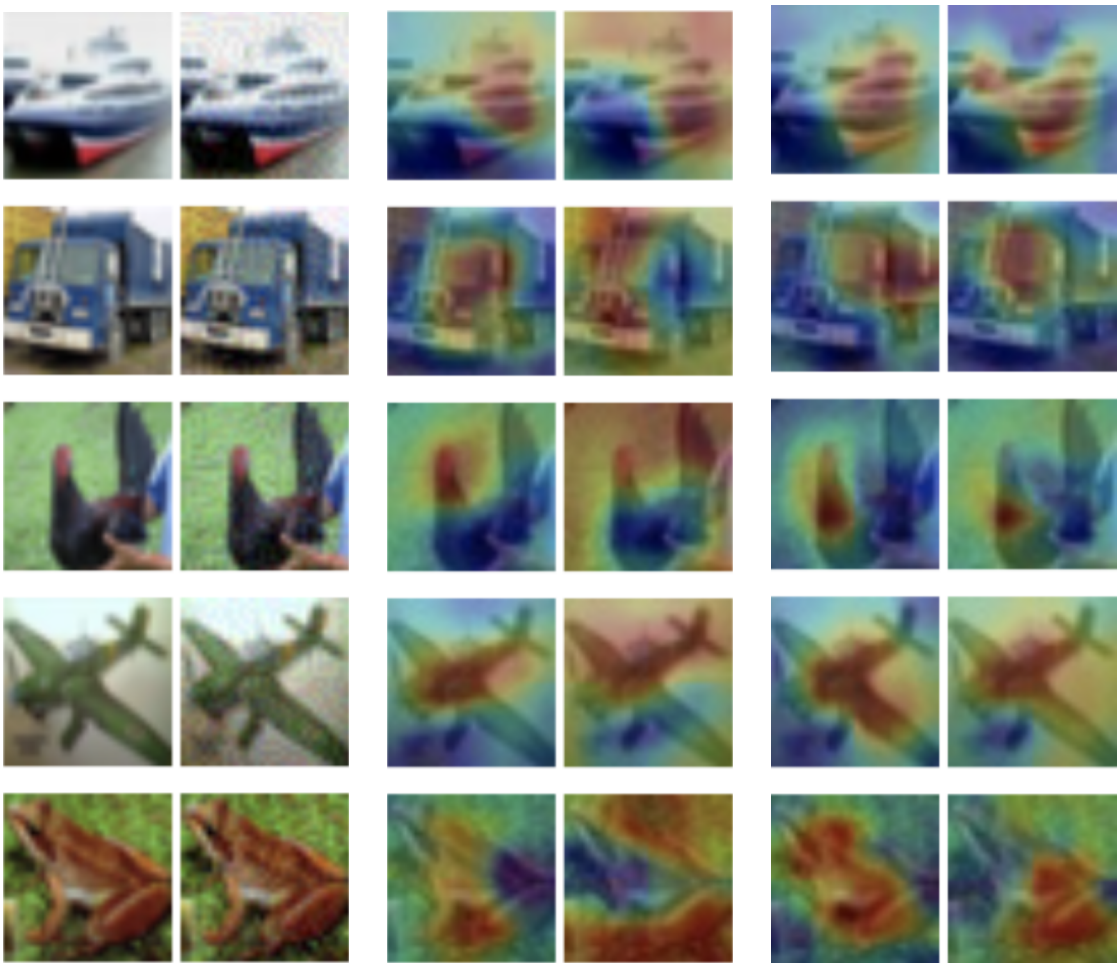
Figure 4.4: Comparison of evaluation results with no defense during inference on original pre-trained model and models trained with DropBlock/SmoothBlock (%).

imply the raw model's robustness. Particularly, as shown in Figure 4.4, with the proposed SmoothBlock regularization during training, we effectively defend the FGSM adversarial attack under the same ϵ setting when compared with the original model. This is because of injecting randomness and blurring some spatial features in training phase such that model is less overfitting to training distributions and can generalize well over simple one-step adversarial attack. The learn-able information becomes sparser, which allows the model less addicted to training distribution and achieves more capacity against simple one-step adversarial attack.

Furthermore, we utilize DropBlock or SmoothBlock in the inference phase towards the aforementioned regularized-trained models. The accuracy results against various attacks are compared in Figure 4.5. As mentioned in section 4.2, original accuracy is degraded when the DropBlock or SmoothBlock is directly applied in pre-trained model. Since many previous works [32] [24] have revealed that there is a trade-off between maintaining the non-perturbation data accuracy and defending the adversarial data, thus, this is another important metric to evaluate the effeteness of defense solutions against adversarial examples. To address this, as can be seen in Figure 4.5, re-training the model with SmoothBlock layer can solve this issue. Towards legitimate images, the classification performance does not pose a significant deterioration compared to the original model (only 0.69% drop). Moreover, defense results in Figure 4.5 gain considerable improvements against all listed attacks. In general, when our proposed SmoothBlock is applied in both training and testing phase, evaluation performance has the most significant enhancement which is comparable to state-of-art defense strategies against adversarial attacks [33] [34].

Train Method	Test Method	Original Acc	FGSM1	FGSM2	FGSM4	FGSM8	FGSM16	FGSM32	PGC Acc	C&W Acc
Original	N/A	90.03	46.80	29.47	19.15	16.64	13.68	11.84	6.44	0.00
DropBlock	DropBlock	89.07	70.66	49.83	32.59	23.54	16.83	12.99	7.09	76.50
	SmoothBlock	83.05	73.80	62.04	47.66	40.41	34.42	28.75	17.08	59.97
SmoothBlock	DropBlock	81.32	**75.12**	**66.00**	50.42	34.39	19.93	12.72	13.91	66.16
	SmoothBlock	**89.34**	70.91	59.73	**56.05**	**52.73**	**46.74**	**30.05**	**27.11**	**76.67**

Figure 4.5: Comparison of evaluation accuracy on regularized-trained ResNet20 models with different test methods against various adversarial attack (%).



(a) Image Examples (b) Original Model (c) SmoothBlock-trained Model

Figure 4.6: Examples of CAMs for original and adversarial images on different models. (a) Examples for clean and adversarial images. (b) (c) CAMs for both clean and adversarial images generated from pre-trained model and SmoothBlock-trained model, respectively.

To explore the underlying reasons for the model's robustness, we visualize the spatial distributed information of the model's attention by the class activation map (CAM) [35]. As shown in the left column of Figure 4.6a, we demonstrate several original image samples

Adv Train Method	Test Method	Original Acc	FGSM 1	FGSM 2	FGSM 4	FGSM 8	FGSM 16	FGSM 32
Vanilla	N/A	81.30	80.16	77.4	69.93	52.97	29.32	14.46
	SmoothBlock	54.61	54.70	54.11	53.60	49.29	33.12	23.36
Train with SmoothBlock	N/A	**85.22**	81.86	75.60	62.24	42.38	23.66	16.91
	SmoothBlock	84.54	**83.17**	**80.03**	**70.85**	**53.99**	**39.64**	**25.82**

(a)

	Original Acc	PGD Acc	Original Acc Test with SmoothBlock	PGD Acc Test with SmoothBlock
Vanilla Adv Training	81.30	39.12	54.61	47.58
Adv Training with SmoothBlock	**82.69**	37.97	82.23	**50.56**

(b)

Figure 4.7: The effeteness of our proposed regularized defense SmoothBlock working along with adversarial training. (a) Evaluation accuracy results against FGSM (%). (b) Evaluation accuracy results against PGD (%).

from CIFAR10 dataset that ResNet20 can successfully classify. However, if adding some adversarial perturbations to legitimate images, the model is easily fooled while the perturbation is imperceptible to people (right column). Figure 4.6b and 4.6c represent CAMs generated from original model and SmoothBlock-regularized model, respectively. By adding a highlighted mask on images, CAMs could localize class-specific regions learned by models. More specially, CAMs in the left column of Figure 4.6b and 4.6c correspond to the legitimate images. We can see that the model attentions mainly focus on objects to be classified. However, for original model under adversarial attacks (right column of Figure 4.6b), its attentions largely spread to other less significant regions which undermine the model's decision to incorrect class. On the other hand, Figure 4.6c shows CAMs for our proposed SmoothBlock-trained model against adversarial attacks. Despite with a slight distraction, the highlighted attention are generally maintain on the top of discriminated class-specific regions to drive model making correct decisions.

4.4 Adversarial Training

Adversarial training has been introduced in [12], and adopted by [11]. It is a form of data augmentation method which now can be considered as a well-known typical defense

method against adversarial examples. In this section, we embed our proposed SmoothBlock regularization into adversarial training to further improve the defense performance. We train the ResNet20 model on CIFAR10 by injecting adversarial examples as the vanilla adversarial training baseline. Following the identical configurations in Section 4.1, the original evaluation accuracy achieves 81.30%. As shown in Figure 4.7a, the model robustness against FGSM attacks has been increased compared to the original baseline without employing adversarial training in Section 4.2. Especially, improvements for attacks with smaller strengths are significant. However, such effect is eliminated when SmoothBlock is applied directly on vanilla adversarial training. Furthermore, we explicitly examine the proposed SmoothBlock with adversarial training by putting it in both training and testing phase. Results are shown in last two rows in Figure 4.7a. It can be seen that not only the model capacity on clean data is increased from 81.30% to 84.54%, but also the top evaluation performance against FGSM attacks is achieved by our proposed SmoothBlock solution across all attacking strengths.

Recent study [11] has developed adversarial training against PGD attack which is known as one of the most powerful attacks. We directly apply our proposed SmoothBlock on vanilla adversarial-trained model. Results are shown in Figure 4.7b. Although test accuracy for PGD is improved, the performance of legitimate data is severely dropped from 81.30% to 54.61%. To address this issue, we put SmoothBlock in training stage along with adversarial training. Accuracy results are reported for testing with and without SmoothBlock (second row in table). It can be easily observed that when our proposed SmoothBlock employed in both training and testing phase, the model robustness against PGD attack gains considerable increase to 50.56%. Besides, for clean data, evaluation accuracy is proportional to the vanilla adversarial training and even gets improved from 81.30% to 82.23%.

Compared to another recent work [24] as aforementioned in section 2.2, our work has similar effect of increasing model robustness by introducing regularization term in model.

Even though we have both achieved analogous performance against FGSM and PGD attacks, PNI is limited that it could not work properly without adversarial training while our method is more flexible. The proposed SmoothBlock can effectively improve model robustness, even when directly applied to pre-trained model as well as embedded into training phase with and without adversarial examples injection.

4.5 Ensemble

Many recent works [14] [24] have proved that introducing randomness is a promising solution against adversarial examples. As discussed, DropBlock and SmoothBlock blocks are randomly generated to sample on intermediate feature maps of DNN models. Notably, to maintain high classification accuracy, DropBlock is not applied during inference. Such randomness would pose uncertainty and bias for model decision, and the impact from DropBlock is worse than SmoothBlock. However, by initiating model ensemble, we can turn this drawback to benefits.

We obtain model predictions for multiple iterations and ensemble results to make final decisions. The model here is ResNet20 trained with SmoothBlock, by randomly picking one regularization term from DropBlock and SmoothBlock in each testing phase. We show ensemble results in Figure 4.8. Compared with one-time test result, the evaluation performance against small-strength FGSM is improved as well as C&W target attack. This contributes to the combination of DropBlock which reflects to our observation that DropBlock behaves better for small perturbations (Section 2.2). Ensemble performance fluctuates in a reasoning range and reaches an upper bound when we initiate more than 100 test iterations. As a result, if inference computation cost is not in concern, we suggest such ensemble strategy to further improve the model robustness.

	Original Acc	FGSM1	FGSM2	FGSM4	FGSM8	FGSM16	FGSM32	PGD Acc	C&W Acc	C&W ASR
One time	89.34	70.91	59.73	56.05	52.73	46.74	30.05	27.11	76.67	7.90
50 Iterations	89.98	74.11	61.27	56.86	53.94	47.49	30.28	27.43	80.58	3.14
100 Iterations	90.06	73.52	61.51	55.97	53.68	47.59	29.81	27.98	80.81	3.08

Figure 4.8: Model self-ensemble results against FGSM, PDG and C&W attaks (%).

CHAPTER 5
FUTURE WORK

With extensive experiments on selecting appropriate block size and location of SmoothBlock, we select the best configurations for our ResNet20 model. The well trained ResNet20 model with SmoothBlock achieves around 90% accuracy on legitimate CIFAR-10 data and even under the adversarial training scenarios it can achieve around 82% on legitimate CIFAR-10 data. With proposed SmoothBlock, the ASR of PGD and C&W hover around 7.8%, which is largely lower than when apply no defense method. It also achieves state-of-art performance when facing FGSM, PGD and C&W mainstream adversarial attack. Also, we apply self-ensemble method to futuer improve the performance.

By trial and error, we found the location, block size and setting configurations can largely influence the performance of the model with SmoothBlock. For example, we may place the two SmoothBlock layers with block size 5 and 3 in the model at the last residual block. We are also interested in different structures like VGG-net, DenseNet and other famous DNN structures, since different solutions of layer connection may affect the performance of SmoothBlock. Additionally, we intent to apply our proposed SmoothBlock on Resnet20 with larger dataset like ImageNet.

CHAPTER 6
CONCLUSION

In this work, we propose a regularized defense method against adversarial examples for DNN models. We conduct extensive experiments to illustrate the effectiveness of the proposed SmoothBlock under different attacking approaches (FGSM, PGD, C&W) in white-box scenarios. According to our convincing experiments, SmoothBlock can be easily embedded into pre-trained model to improve adversarial accuracy. Moreover, the model robustness achieves state-of-art performance when putting our proposed SmoothBlock in training and adversarial training, without degradation on clean data. Besides, we suggest an ensemble strategy to further improve DNN robustness. With our proposed SmoothBlock, the model is able to gain high robustness, reduce the negative impact from perturbed input data and achieve excellent classification performance with clean data.

REFERENCES

[1] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," in *Advances in neural information processing systems*, 2013, pp. 2553–2561.

[2] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for natural language processing," *arXiv preprint arXiv:1606.01781*, vol. 2, 2016.

[3] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[4] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[5] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," 2015. eprint: `arXiv:1512.00567`.

[6] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[7] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[9] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," *arXiv preprint arXiv:1611.02770*, 2016.

[10] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," *arXiv preprint arXiv:1810.00069*, 2018.

[11] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[12] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[13] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "Dropblock: A regularization method for convolutional networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 10 750–10 760.

[14] X. Liu, M. Cheng, H. Zhang, and C.-J. Hsieh, "Towards robust neural networks via random self-ensemble," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 369–385.

[15] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 315–323.

[16] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *arXiv preprint arXiv:1611.01236*, 2016.

[17] 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.

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

[19] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ACM, 2017, pp. 506–519.

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

[21] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[22] G. S. Dhillon, K. Azizzadenesheli, Z. C. Lipton, J. Bernstein, J. Kossaifi, A. Khanna, and A. Anandkumar, "Stochastic activation pruning for robust adversarial defense," *arXiv preprint arXiv:1803.01442*, 2018.

[23] J. Svoboda, J. Masci, F. Monti, M. M. Bronstein, and L. Guibas, "Peernets: Exploiting peer wisdom against adversarial attacks," *arXiv preprint arXiv:1806.00088*, 2018.

[24] A. S. Rakin, Z. He, and D. Fan, "Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack," *arXiv preprint arXiv:1811.09310*, 2018.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[27] A. Paszke, S. Gross, S. Chintala, and G. Chanan, "Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration," *PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration*, vol. 6, 2017.

[28] A. Krizhevsky, V. Nair, and G. Hinton, "The cifar-10 dataset," *online: http://www. cs. toronto. edu/kriz/cifar. html*, vol. 55, 2014.

[29] M.-I. Nicolae, M. Sinn, M. N. Tran, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, I. M. Molloy, *et al.*, "Adversarial robustness toolbox v0. 4.0," *arXiv preprint arXiv:1807.01069*, 2018.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[31] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[32] A. Athalye and N. Carlini, "On the robustness of the cvpr 2018 white-box adversarial example defenses," *arXiv preprint arXiv:1804.03286*, 2018.

[33] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, "Pixeldefend: Leveraging generative models to understand and defend against adversarial examples," *arXiv preprint arXiv:1710.10766*, 2017.

[34] T. Na, J. H. Ko, and S. Mukhopadhyay, "Cascade adversarial machine learning regularized with a unified embedding," *arXiv preprint arXiv:1708.02582*, 2017.

[35] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2921–2929.