Mitigating Adversarial Attacks for Deep Neural Networks by Input Deformation and Augmentation

Pengfei Qiu[1,2,3,†], Qian Wang[3,†], Dongsheng Wang[1,2], Yongqiang Lyu[2*], Zhaojun Lu[3], Gang Qu[3]
[1]Department of Computer Science and Technology, Tsinghua University, Beijing, China.
[2]Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing, China.
[3]Dept. of Electrical and Computer Engineering & Institute for Systems Research, Univ. of Maryland, College Park, USA.
qpf15@mails.tsinghua.edu.cn, {wds, luyq}@mail.tsinghua.edu.cn, {qwang126, lzj77521, gangqu}@umd.edu

Abstract— Typical Deep Neural Networks (DNN) are susceptible to adversarial attacks that add malicious perturbations to input to mislead the DNN model. Most of the state-of-the-art countermeasures concentrate on the defensive distillation or parameter re-training, which require prior knowledge of the target DNN and/or the attacking methods and hence greatly limit their generality and usability. In this paper, we propose to defend against adversarial attacks by utilizing the input deformation and augmentation techniques that are currently widely utilized to enlarge the dataset during DNN's training phase. This is based on the observation that certain input deformation and augmentation methods will have little or no impact on DNN model's accuracy, but the adversarial attacks will fail when the maliciously induced perturbations are randomly deformed. We also use the ensemble of decisions to further improve DNN model's accuracy and the effectiveness of defending various attacks. Our proposed mitigation method is model independent (i.e. it does not require additional training, parameter fine-tuning, or any structure modifications of the target DNN model) and attack independent (i.e., it does not require any knowledge of the adversarial attacks). So it has excellent generality and usability. We conduct experiments on standard CIFAR-10 dataset and three representative adversarial attacks: Fast Gradient Sign Method, Carlini and Wagner, and Jacobian-based Saliency Map Attack. Results show that the average success rate of the attacks can be reduced from 96.5% to 28.7% while the DNN model accuracy is improved by about 2%.

I. Introduction

Deep Neural Network (DNN) has shown exceptionally good performance in a wide range of applications, including image classification, speech recognition, and object recognition [1]. However, recent studies have demonstrated that DNN is extremely vulnerable to adversarial attacks [2-4]. The attackers can maliciously force a DNN model to generate their desired consequences by introducing small perturbations to the input that are imperceptible to human. Several powerful approaches have been proposed to generate effective adversarial examples in the literature. For example, Fast Gradient Sign Method (FGSM)) [5], Carlini and Wagner (C&W) [6], and Jacobian-based Saliency Map Attack (JSMA) [7].

Improving the robustness of DNN models and defending them against the aforementioned adversarial attacks are equally important [1, 8]. Existing defense is achieved by either revising the structure of the DNN model [2, 9-12] or adding extra noises on the training data to simulate the conceivable adversarial examples [5, 13, 14]. However, these approaches all require the defenders to know the details of the target DNN models and/or the attacker's method to construct the perturbation. Such assumption might not be practical and hence greatly limit their generality and usability. In this paper, we propose a more general method to prevent the adversarial attacks by employing the input deformation and augmentation mechanisms. We consider the target DNN model as a black-box and do not assume any knowledge on how the adversarial examples are generated. Especially, our method is conceptually different from the existing researches that rely on the input manipulations, in which the basis representation of the input is modified by Principal Component Analysis (PCA), low-pass filter, image compression, resizing, padding, or bit-depth reduction [15-17].

The enabling technique of our proposed approach is input deformation and augmentation. Since the accuracy of a DNN model normally increases with the size of the training dataset. Data deformation and augmentation has been widely adopted to expand the size of a training dataset by artificially constructing new data from existing data. We observe that DNN models are able to recognize variations of an input obtained from certain data deformation and augmentation methods such as *width-shift*, *height-shift*, *zoom*, *rotation*, and *shear*. This is because that the DNN model infers results from the high-level abstractions of the input and hence can still recognize the variations generated from the normal examples as long as the variations have negligible feature loss. However, the DNN models have not been trained by the variations of the perturbations added by the adversary. Therefore, if we deform an input with malicious perturbations, the DNN model may not be able to recognize the perturbation and won't mislead the model to produce the attacker's desired output. This is the basic idea behind our mitigation approach which conceptually works as follows: for each input, we randomly create multiple variations and use the DNN model to produce decision for each variation, then the ensemble of decisions is used to generate the final output. Our proposed mitigation approach has the following features:

1) The proposed mitigation method is model independent (i.e. it does not require additional training, parameter fine-tuning, or any structure modifications of the target DNN model) and attack independent (i.e., it does not require any knowledge of the adversarial attacks). So it can be applied to defend any DNN model against any perturbation-based attacks.

[†]Pengfei Qiu and Qian Wang are co-first authors.
[*]Corresponding author.

978-1-7281-4123-7/20/$31.00

2) The proposed method uses multiple variations of the input for the final decision making. It improves the accuracy of the DNN model. We exploit the *accuracy threshold*, *probability summation*, and *majority voting* mechanisms and discuss the parameters that influence the defense success rate and the network accuracy.
3) The above claims are validated empirically to demonstrate the effectiveness of our method. On the CIFAR-10 dataset, we reduce the average success rate of three representative attacks (FGSM, C&W, and JSMA) from 96.5% to 28.7% while the DNN model accuracy is improved by about 2%.

The remainder of this paper is organized as follows. Section II introduces the essential preliminaries of DNN, adversarial examples, and the data augmentation technique. Section III describes the details of our proposed defense method. Section IV gives the real implementations of our method and presents the experiment results. At the end of the paper, Section V discusses the security, limitation, and the future work, section VI concludes the paper.

II. Preliminaries

In this section, we introduce the fundamental preliminaries corresponded to the DNN, adversary examples, and the data augmentation applied in the neural networks, respectively.

A. DNN

A DNN can be viewed as a highly complex function that is capable of non-linearity mapping and exhibits high effectiveness in modeling. It usually consists of multiple successive convolutional or sub-sampling layers, followed by one or more fully connected layers. The convolutional layers create a feature map from the original input images which can be further aggregated at various locations to form a matrix of lower dimension in width and height, but deeper in depth. The following fully connected layers use the feature matrix formed from previous layers to generate a class label.

B. Adversary Examples

The first concept of adversary examples is proposed by Szegedy et al. [18]. They successfully caused the network to misclassify an image by applying a certain perturbation, while the human eyes cannot recognize the differences. Suppose we have a trained deep learning model F and an original input data sample X. The objective of an adversary is to generate an adversarial example $X_* = X + \delta_X$ whose label will not be the same as the original one. The construction for the perturbation δ_X can generally be formalized as a box-constrained optimization problem shown in equation 1.

$$\arg\min_{\delta_X} \{\|\delta_X\| : F(X + \delta_X) \neq F(X)\} \tag{1}$$

This optimization problem minimizes the perturbation while misclassifying the prediction. Different techniques are developed to find an approximate solution to this optimization problem. Here we describe some well-known techniques mentioned in the recent literature.

Fast Gradient Sign Method (FGSM): Goodfellow et al. proposed the FGSM to construct the adversarial examples [5], which utilizes the fast sign methodology to calculate the gradient of the cost function. The adversarial examples can be generated by the equation 2.

$$X_* = X + \epsilon * sign(\nabla_x J(X, Y)) \tag{2}$$

Here, J is the cost function of the trained model and $\nabla_x(*)$ represents the gradient of the model with respect to a normal sample of X. The perturbation amplitude is controlled by the parameter ϵ. FGSM tries to change the pixel values based on the gradient of the cost function.

Jacobian-based Saliency Map Attack (JSMA): Papernot et al. introduced a different approach to find the sensitivity direction by using the forward derivative [7]. This method constructs a saliency map with the gradients of the output with respect to each input pixel. JSMA allows selecting the pixels which have the most likelihood of labeling the images to target class (the maximum gradient) based on the saliency map and chooses to perturb them. As a consequence, it could achieve a high success rate to misclassify the targets.

C&W: C&W is a stronger iterative attack method proposed by Carlini et al. [19], which formulates the problem of finding an adversarial instance as equation 3.

$$\begin{aligned} &\text{minimize } D(X, X + \delta_X) \\ &\text{such that } F(X + \delta_X) = Y_*, X + \delta_X \in [0, 1]^n \end{aligned} \tag{3}$$

Since the constraint to express the prediction of a neural network is highly non-linear and difficult for existing optimization algorithms, they replaced the constraint with an objective function f such that $F(X + \delta_X) = Y_*$ if and only if $f(X + \delta_X) \leq 0$. This problem could be solved almost exclusively by choosing proper object function f.

C. Data Augmentation

Training deep learning neural network might require more data to tune a large number of parameters in order to get good performance. However, acquiring more real training data and organizing them need much more labor work and sometimes is not practical. Besides, in the real world scenario, we may have a dataset of images taken in a limited set of conditions. However, the target application may exist in a variety of conditions, such as different orientation, location, scale, brightness, etc. Therefore, the augmentation technique is proposed to artificially create variations of the images that can improve the ability to fit the models [20, 21].

One baseline of the data augmentation is that the created new images should be consistent with the original images. For example, a horizontal flip of a picture of a cat may make sense, because the photo could have been taken from the left or right. A vertical flip of the photo might not make sense and probably would not be classified appropriately, such as an upside down cat. The most commonly used transformations of images include shifts, flips, zooms, and rotations. The guarantee to apply the augmentation successfully is that the deep learning algorithms can learn features that are invariant to the location.

III. Mitigating the Adversarial Attacks with Input Deformation and Augmentation

In this section, we present our countermeasure of applying the deformation and augmentation techniques onto the input data to defend against the adversarial attacks.

A. Threat Model

We define the threat model as the attackers attempt to break the trained DNN with self-constructed adversarial examples. The goal of the attack is to entice the DNN to output specified consequences. In the side of defenders, they have no information about the internal structure of the target DNN and have no permissions to change the network parameters. Besides, they are not aware of how the adversarial examples are created.

B. Assumptions

In this work, we have two prerequisite assumptions.

- We assume the target DNN is vulnerable to adversarial attacks and the attackers have successfully created the adversarial examples that can fool the DNN.
- We assume the defenders have abilities to operate the inputs and observe the outputs but cannot manipulate any intermediate data of the model. In this assumption, the DNN is treated as an invariable black-box for the defenders.

C. Defense Overview

The overview of our proposed defense is illustrated in Fig. 1, which mainly consists of three parts: input data pre-processing, DNN, and the output decision fusion. For an input, the pre-processing unit will first create multiple variations using the data deformation and augmentation techniques. Next, the DNN will infer each variation with trained parameters. Finally, the output decision component determines the ultimate result from the decisions of the DNN. The example DNN in Fig. 1 owns two convolution layers and two dense layers. However, the defense is effective for any models as it only regulates the inputs and outputs of the DNN.

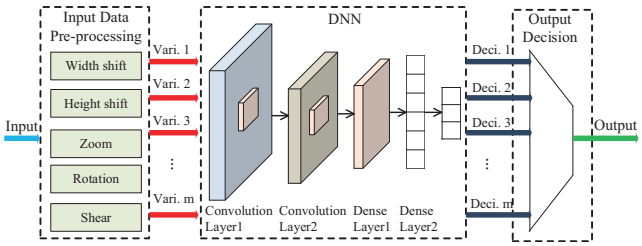

Fig. 1. The overview of our proposed defense on an example DNN. The data pre-processing unit plays a role to generate multiple variations from an input. The DNN receives every variation and recognizes it. The output decision element makes the final result for the input.

When building the malicious perturbations, one base requirement is that they should be as small as possible so that the human cannot notice them in eyes. The gradient algorithms are widely exploited in adversarial attacks, which will adjust the parts of the input data that are critical to the output. Therefore, the perturbations will be different for disparate inputs. In our design, it is the deformations of the adversarial examples that are inputted into the DNN, however, the current perturbations will not match the new input, which cannot complete the attackers' goals. In general, the deformation and augmentation methods do not change the high-level abstractions of an input, consequently, the DNN can still be available to classify the normal examples.

D. Input Data Pre-processing

Generally, it will be resource saving if we only utilize one random variation to identify the adversarial examples. However, the variation may be insufficient to remove the side effects of the perturbations. Moreover, the network accuracy may decrease a lot, especially when the size of the trained data is small. Therefore, we utilize the input data pre-processing unit to create m variations.

Analogously, transforming the input data by fixed parameters is not a good choice. In this work, we measure the possible deformation and augmentation parameters one by one and observe its effects on the high-level abstractions of the input data. Consequently, we choose to create the variations from the combination of five variables as they are able to hold the high-level abstractions of the input data: 1) *width-shift* (d_w), the horizontal location of every element is shifted; 2) *height-shift* (d_h), the vertical site of every component is shifted; 3) *zoom* (d_z), focus on a specific part of the input data; 4) *rotation* (d_r), maps the position of an element in the input onto a position by rotating it through an angle; 5) *shear* (d_s), displaces each point in the fixed direction.

In order to enhance the difficulties for attackers to form the adversarial examples for our defense, all of the parameters for the five aspects are adjusted randomly and dynamically. This will increase the availability of the defense with negligibly negative impacts on the functions of the DNN because the features extracted from normal examples would not change even though the inputs are randomly deformed.

E. Output Decision

In our design, we utilize the simplest but the most exploited method, the minority is subordinate to the majority, to decide the final result o from outputs of DNN for different variations. We employ three mechanisms to acquire the conclusive result.

Accuracy threshold. For an input $\mathbf{x}$ that has a label y, such as a normal test instance or an adversarial example, we can construct a small dataset from the variations of the example and test the network accuracy acc on the dataset. For a normal input, the network accuracy will be very high due to most of the variations will be recognized error-freely. On the contrary, we can conclude that the input is an adversarial example if the network accuracy is less than a predefined threshold of τ. This mechanism can be used to rapidly evaluate the model accuracy on the test dataset. Equation 4 demonstrates the method.

$$o = \begin{cases} y, & if\ acc \geq \tau \\ adversarial\ example, & if\ acc < \tau \end{cases} \tag{4}$$

Probability summation. For each input, the softmax function of the DNN would regulate the output to a categorical probability, which tells us the probability that any of the classes are true. Then, we calculate the summation of all the probabilities for every category and choose the one that has the biggest probability as the DNN result. Formally, suppose p_{ik} denotes the probability of the ith variation belongs to category k, the size of the total classes is N, this method will be illustrated as equation 5.

$$o = j,\ where \sum_{i=1}^{m} p_{ij} \geq \sum_{i=1}^{m} p_{ik},\ \forall k \in [1, N]\ and\ k \neq j \tag{5}$$

Majority voting. If the output of the DNN is the class of the input instead of the probabilities, which is also the most common situation, we select the category that possesses the most variations as the output. Suppose the n_k represents the total amount of variations that are classified as category k, we demonstrate this mechanism as equation 6.

$$o = j,\ where\ n_j \geq n_k,\ \forall k \in [1, N]\ and\ k \neq j \quad (6)$$

IV. Evaluation and Results

In this section, we introduce the implementation algorithm and experiment results about our defense.

A. Implementation Algorithm

We utilized the Algorithm 1 to implement our defense for protecting the target DNN from being attacked by adversarial examples. The algorithm first constructs a dataset with the variations of the input and then exploits the DNN to identify every variation. Finally, it creates an output based on the different output decision mechanisms.

Algorithm 1: The realization algorithm for our defense.

Input : Input data, $\mathbf{x}$; Target DNN, DNN; Deformation parameters, d=$\{d_w, d_h, d_z, d_r, d_s\}$; Number of variations, m; Decision method, v; Label, y; Accuracy threshold, τ.
Output: The classification result, o

$Variation[m]$;
$N \leftarrow Number\_of\_Classes$;
for $i \in [1, m]$ **do**
  $Variation[i] \leftarrow$ GENERATE_A_VARIATION($\mathbf{x}$, d);
end
if v == *Accuracy threshold* **then**
  $acc \leftarrow DNN$.EVALUATE($Variation$, y);
  $o \leftarrow y$ if $acc \geq \tau$ else $adversarial\ examples$;
else if v == *Probability summation* **then**
  $P \leftarrow DNN$.PREDICT($Variation$);
  $P\_sum[N]$;
  for $i \in [1, m]$ **do**
    P_sum += $P[i]$;
  end
  $o \leftarrow j, where\ P\_sum[j] \geq P\_sum[i]\ for\ i \in [1, N]$
else
  $C \leftarrow DNN$.PREDICT_CLASSES($Variation$);
  $C\_sum[N] \leftarrow 0$;
  for $i \in [1, m]$ **do**
    $C\_sum[C[i]]$++;
  end
  $o \leftarrow j, where\ C\_sum[j] \geq C\_sum[i]\ for\ i \in [1, N]$
Return o;

B. Datasets and Models

To evaluate the effectiveness of our proposed countermeasure, we trained the two DNN models provided by *Keras*[1] for CIFAR-10 classification task: one is with the data augmentation while training, the other model is without data augmentation. The model architecture is given in Table I. When performing the data augmentation, the Model II is a bit under fitting the dataset. To keep those models identical to accuracy, we didn't apply any further tuning on the models. After training, the model I achieves 79.4% accuracy and model II achieves 79.0% accuracy.

[1]Keras: https://keras.io/examples/cifar10_cnn

TABLE I
ARCHITECTURE OF NEURAL NETWORK MODELS

	Model I (w/o Augmentation)	Model II (with Augmentation)
$Conv2D$	64 filters with size (3×3)	32 filters with size (3×3)
$Conv2D$	64 filters with size (3×3)	32 filters with size (3×3)
$Maxpool$	kernel size (2×2)	kernel size (2×2)
$Conv2D$	128 filters with size (3×3)	64 filters with size (3×3)
$Conv2D$	128 filters with size (3×3)	64 filters with size (3×3)
$Maxpool$	kernel size (2×2)	kernel size (2×2)
$DenseLayer1$	256 units	512 units
$DenseLayer2$	256 units	N/A
$OutputLayer$	10 units	10 units

We generated the adversarial examples first using JSMA methods from the *Foolbox* tool[2]. In addition, we implemented FGSM and C&W attacks with the *CelverHans* package[3]. With the C&W attack, we first performed 20 iterations of binary search over the parameter *c*. For the selected optimal *c*, we run 1000 iterations of gradient descent with the Adam optimizer. For each method, we randomly generated 2000 adversarial examples.

C. Results

We evaluate our proposed defense method with two metrics: attack success rate and model accuracy. In this experiment, the output is decided by *probability summation* and the deformation parameters are $d_e = \{0.1, 0.1, 0.2, 10, 5\}$.

1) Attack success rate: We first test the attack success rate on Model I with the adversarial examples created by FGSM, JSMA, and C&W, respectively. The results are shown in Fig. 2. The x-asix represents how many transformed variations (m) are utilized in the one-time inference. We can observe dramatically drops for the adversarial attack success rate across all the three attack methods. The attack success rate of the FGSM attack decreases from 97.0% to 25.0%. For the JSMA attack, the drop rate is even larger, which is from 99.8% to 13.3%. As for the C&W attack, one of the most powerful adversarial attacks, our method still successfully defend it by decreasing the success rate from 92.8% to 27.7%. Next, we evaluate the attack success rates of the three adversarial attacks on Model II. The reductions of the attack success rate are very similar to them in the Model I, which are also very obvious. It is important to realize that the results between different numbers of variations (m) do not change too much. As a result, choosing a smaller augmentation number is desirable as saving delay time and computations.

2) Model accuracy: The ultimate aim for any defense method against adversary examples is to detect the attacks or decrease the attacker success rate with holding the original test accuracy. Here, we captured the model accuracy on the test dataset with different m. From Fig. 3, we can observe that when we applied the data deformation and augmentation simultaneously, the test accuracy does not drop for both models as long as $m \geq 3$. Due to the voting scheme, the test accuracy even increased from the baseline. With a larger augmentation number, which indicates several decisions are assembled together to generate the final result, the test accuracy increases up to 2%. This prominent result demonstrated the superiority of our method that it could address the

[2]Foolbox: https://github.com/bethgelab/foolbox
[3]Cleverhans library: https://github.com/tensorflow/cleverhans

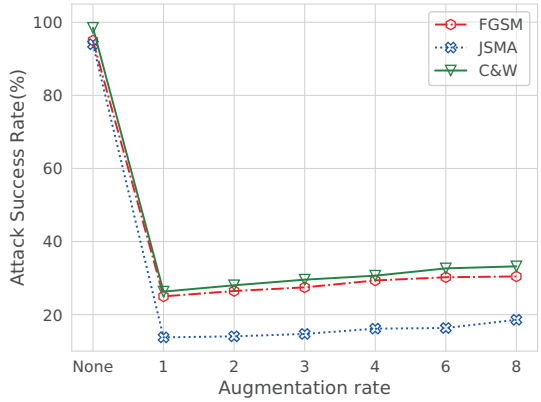

Fig. 2. The adversarial attack success rate for Model I with different m.

adversarial attacks by meantime boost the accuracy even when the target DNN model is trained without data augmentation.

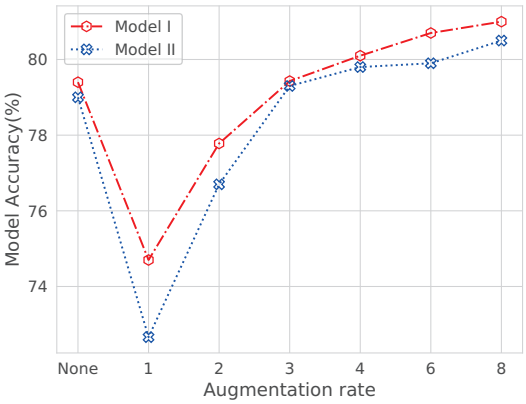

Fig. 3. The model accuracy with various m.

D. Other parameters

Besides m, we also explore the effects on the performance of our method taken by d and v. Without loss of generality, we take Model I as the target DNN in this experiment.

1) d: The similarity between the variations and the original data will influence the defense functions. On one hand, if the variations are closely similar to the original data, the perturbations may still be efficient to fool the DNN. On another hand, if the input is deformed drastically, the DNN may mistakenly recognize the variations for normal examples. Therefore, the parameter d is a significant factor in the method. We obtain the performance changes of the defense with different d are provided when addressing the JSMA attack, which can be found in Table II. In this experiment, the m is fixed to 6 and the output is decided by the *accuracy threshold* with $\tau = 0.5$. The model accuracy drops with the range of the deformation augments. The adversary attack success rate first decreases and then increases when the deformation scope heightens. The reason for the attack success rate increases is that the model accuracy lowers down.

TABLE II
THE PERFORMANCE CHANGES OF THE DEFENSE WITH DIVERSE d

	Changing rotation range d_r (degree)				
d_r	$d_r = 5$	$d_r = 10$	$d_r = 20$	$d_r = 30$	$d_r = 40$
Attack success rate	20.3%	17.9%	16.6%	18.0%	18.2%
Model accuracy	80.3%	79.8%	79.5%	77.7%	75.1%
	Changing zoom range d_z (fraction)				
d_z	$d_z = 0.1$	$d_z = 0.2$	$d_z = 0.3$	$d_z = 0.4$	$d_z = 0.5$
Attack success rate	17.6%	17.3%	17.9%	20.3%	20.7%
Model accuracy	80.5%	79.2%	78.8%	77.2%	75.4%
	Changing width shift range d_w (fraction)				
d_w	$d_w = 0.05$	$d_w = 0.1$	$d_w = 0.15$	$d_w = 0.2$	$d_w = 0.3$
Attack success rate	14.4%	15.4%	18.8%	22.5%	24.6%
Model accuracy	80.6%	79.7%	79.2%	75.8%	68.4%

2) v: We test the attack success rate and model accuracy with diverse accuracy threshold (τ) when the output decision method is *accuracy threshold* and $d = d_e$. The experiment results are illustrated in Table III. The notation n/m means that when n of m variations have the original label, we will finalize the result as the original label. We also employed *majority voting* to form the final decision and the results are very similar to Table III. It is clear to find out that the adversary accuracy changed a bit with different voting schemes.

It is natural that when we have the same number of votes m, larger n or τ would make it difficult to achieve the consistency but would lead the accuracy decreases. However, if we make the n is half of the m or $\tau = 50\%$, the accuracy would maintain to the baseline and the adversary success rate would decrease as using probability summations.

TABLE III
THE MODEL ACCURACY AND ATTACK SUCCESS RATE WITH DIFFERENT τ.

Augmentation	FGSM		JSMA		C&W	
Threshold (τ)	model	adversarial	model	adversarial	model	adversarial
7/8	58.2%	14.5%	58.2%	11.7%	58.7%	9.7%
6/8	67.2%	20.5%	68.4%	17.3%	67.4%	21.9%
5/8	73.5%	25.7%	74.9%	23.5%	76.5%	34.5%
4/8	79.0%	32.4%	79.9%	30.8%	81.6%	45.1%
5/6	50.8%	9.4%	51.3%	7.9%	50.8%	4.0%
4/6	72.5%	26.3%	74.3%	24.1%	72.4%	31.8%
3/6	79.8%	33.6%	82.5%	33.3%	82.2%	46.1%
3/4	70.4%	24.2%	71.3%	21.5%	71.2%	28.2%
2/4	82.5%	34.0%	81.3%	36.3%	81.2%	49.9%

V. DISCUSSION

In this section, we discuss the security and limitation of our proposed method and also look forward to some future extensions.

A. Security Analysis

The method mitigates the perturbations by randomly manipulating the input data with features are kept. It is impossible for attackers to predict the actual variations because the deformation and augmentation parameters are dynamically updated, therefore, they cannot construct adversarial examples for the variations that will be used in the DNN predication process. However, because the perturbations are not really removed, the method might not accurately extract the high-level abstraction as the original examples. This is a general problem when addressing adversarial attacks. We measured the number of adversarial examples that are classified as they are not combined with perturbations, which is 4524 among the 6000 adversarial examples.

B. Limitation

The proposed method can efficaciously prevent the adversarial attacks without accuracy loss. Besides, it ignores the structure of the DNN and the attack method. However, the DNN works on a set of variations rather than an example, which will take additional time overheads. Fortunately, our design is perfectly workable for parallel execution. The classifications for different variations can be performed on various concurrent threads to largely reduce the time overhead. Moreover, the number of variations (m) is also a fatal factor for the time overhead. Generally, the less of m is, the less time overhead will be. However, reducing m may be disadvantageous for the network accuracy. Therefore, the trade-off between the time overhead and the performance should be explored in advance.

C. Future Work

More extensions or technologies can be studied to further reduce the time overhead, augment the defense success rate, or lower the network accuracy loss. First of all, the time overhead for generating the variations can be lessened by inserting a preprocessing network layer before the first layer to complete the input deformation and augmentation instead of utilizing software algorithms. Besides, implementing the defense with hardware that supports parallel technology will also benefit for lowering the execution time. Secondly, conducting more modifications on the input without damaging the features may be operative for augmenting the defense success rate. Thirdly, accurately controlling the deformation directions or extents may be studied to drop the network accuracy loss.

VI. Conclusion

The DNN is demonstrating more and more performance advantages in various machine learning applications. However, the vulnerability of DNN to adversarial attacks has generated a lot of interests and concerns. The DNN can be deliberately misled to create malicious consequences with adversarial examples that contain additional noises. Most of the current security methods for defending against adversarial attacks assume the target DNN model is accessible or the parameters can be re-trained with the training data that contain the adversarial examples. In this paper, we propose a defense approach that is model-irrelevant and attack-irrelevant, in which the input data is deformed before it is fed to the DNN. Besides, we exploit three kinds of ensembles of decision for different applications to ensure the effectiveness and reduce the network accuracy loss. We successfully deploy the defense to efficaciously prevent the most popular adversarial attacks on the CIFAR-10 datasets. The experiments claim that our method can achieve a high defense success rate without network accuracy loss. At the end of the paper, we analyze the security, limitation, and possible extensions about this work.

Acknowledgements

We gratefully thank the anonymous reviewers for their valuable suggestions and comments. This work was supported in part by the National Key Research and Development Plan of China under Grant No. 2016YFB1000303 and the Guangdong Province Key Project of Science and Technology under Grant No. 2018B010115002.

References

[1] S. Wang, X. Wang, P. Zhao, W. Wen, D. Kaeli, P. Chin, and X. Lin, "Defensive dropout for hardening deep neural networks under adversarial attacks," in *Proceedings of the International Conference on Computer-Aided Design*, ser. ICCAD '18. New York, NY, USA: ACM, 2018, pp. 71:1–71:8.

[2] 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)*. San Jose, CA, USA: San Jose, CA, USA, May 2016, pp. 582–597.

[3] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *CoRR*, vol. abs/1607.02533, 2017.

[4] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *CoRR*, vol. abs/1607.02533, 2017.

[5] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," *arXiv e-prints*, p. arXiv:1412.6572, Dec 2014.

[6] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, ser. AISec '17. New York, NY, USA: ACM, 2017, pp. 3–14.

[7] 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)*. Saarbrucken, Germany: IEEE, March 2016, pp. 372–387.

[8] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*, 2018. [Online]. Available: http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2018/02/ndss2018_03A-4_Xu_paper.pdf

[9] N. Papernot and P. D. McDaniel, "On the effectiveness of defensive distillation," *CoRR*, vol. abs/1607.05113, 2016.

[10] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rJzIBfZAb

[11] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, Jul. 2018.

[12] Q. Liu, T. Liu, Z. Liu, Y. Wang, Y. Jin, and W. Wen, "Security analysis and enhancement of model compressed deep learning systems under adversarial attacks," in *Proceedings of the 23rd Asia and South Pacific Design Automation Conference*, ser. ASPDAC '18. Piscataway, NJ, USA: IEEE Press, 2018, pp. 721–726.

[13] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[14] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2107–2116.

[15] U. Shaham, J. Garritano, Y. Yamada, E. Weinberger, A. Cloninger, X. Cheng, K. Stanton, and Y. Kluger, "Defending against adversarial images using basis functions transformations," *arXiv preprint arXiv:1803.10840*, 2018.

[16] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. L. Yuille, "Mitigating adversarial effects through randomization," *CoRR*, vol. abs/1711.01991, 2017. [Online]. Available: http://arxiv.org/abs/1711.01991

[17] C. Guo, M. Rana, M. Cissé, and L. van der Maaten, "Countering adversarial images using input transformations," *CoRR*, vol. abs/1711.00117, 2017. [Online]. Available: http://arxiv.org/abs/1711.00117

[18] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations*, 2014. [Online]. Available: http://arxiv.org/abs/1312.6199

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

[20] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, March 2017.

[21] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding data augmentation for classification: when to warp?" in *2016 international conference on digital image computing: techniques and applications (DICTA)*. IEEE, 2016, pp. 1–6.