ROBUST DYNAMICAL STEP ADVERSARIAL TRAINING DEFENSE FOR DEEP NEURAL NETWORKS

A Thesis

by

YUKUN HE

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2018

Major Subject: Computer Engineering

ABSTRACT

Although machine learning (ML) algorithms show impressive performance on computer vision tasks, neural networks are still vulnerable to adversarial examples. Adversarial examples typically stay indistinguishable to human, while they can dramatically decrease the classifying accuracy of the neural network. Adversarial training generates such examples and train them together with the clean data to increase robustness. Researchers has stated that the ”projected gradient descent” (PGD) adversarial training method specifies a concrete security guarantee on the neural network against adversarial attacks. The model trained with PGD adversaries performs robust against several different gradient based attack methods under l_∞-norm. This work proposes a *Dynamical Step Adversarial* (DSA) training method to generate adversaries for training by dynamically adjusting the length of step during each iteration. The paper demonstrates the robustness of DSA adversarial training model against different gradient-based attacks. The performance of the DSA training model under different l_∞-norm measurement attacks is compared with other protection methods. Finally, DSA with different numbers of steps are compared under Fast Gradient Sign Method (FGSM) and PGD attacks.

DEDICATION

To my parents for their unconditional love, my friends for their concern and support

ACKNOWLEDGMENTS

I would like to thank my thesis advisor Dr. Peng Li for his contributing advice and continuous guidance for the research. Also, I would like to thank my committee member Dr. Alex Sprintson and Dr. Xia Hu for being on my committee and suggestion given by them through whole research progress. Special thanks to my research colleague Tingjui Chang for lots of discussion and co-working through out the research, and help received from him. Also I shall express my thanks for the Texas A&M University for the great time of both in classes and entertainments I got here, from which I obtained a lot of knowledge and wonderful memories. After all, I would like to thank my family and friends for their supports and help.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Dr. Peng Li and Dr. Alex Sprintson of the Department of Electrical & Computer Engineering and Dr. Xia Hu of the Department of Computer Science & Engineering.

The analysis depicted in Chapter 2 and Chapter 3 were conducted with the help from Tingjui Chang of the Department of Electrical & Computer Engineering.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

Graduate study was supported as student technician under Dr. Peng Li's advising.

NOMENCLATURE

AI	Artificial Intelligence
NN	Neural Network
CNN	convolutional Neural Network
BP	Backpropagation
CONV	Convolutional Layer
FC	Fully Connected Layer
ReLU	Rectified Linear Unit
FGSM	Fast Gradient Sign Method
IFGSM	Iterative Fast Gradient Sign Method
MIFGSM	Momentum Iterative Fast Gradient Sign Method
PGD	Projected Gradient Descent
GAN	Generative Adversarial Network
DSA	Dynamical Step Adversary
MNIST	Modified National Institute of Standards and Technology
C&W	Carlini and Wagner Attack
CIFAR	Canadian Institute For Advanced Research

TABLE OF CONTENTS

LIST OF FIGURES

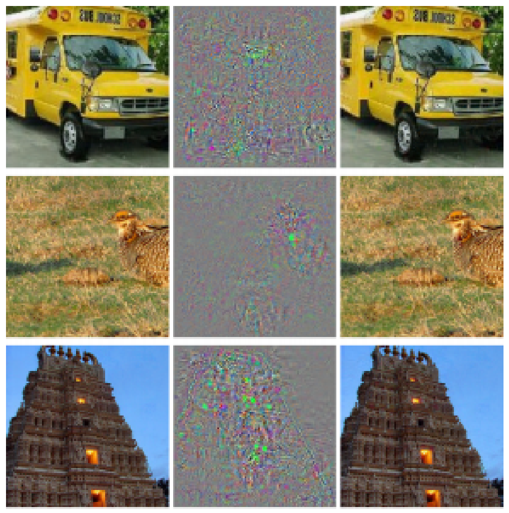
LIST OF TABLES

1. INTRODUCTION

1.1 Background

Deep neural networks have reached a high performance in many different machine learning tasks. However, a well-trained network classifier can be easily attacked by adding an artificial perturbations on the input data. Adversarial examples generated through many distinguished methods have high probability to be misclassified by the neural network while still indistinguishable to human [3]. This phenomenon is discovered in nearly all distinct kinds of network structures. Researchers show that under the circumstances that the parameters of a neural network is unknown to the attackers, it is still vulnerable by applying adversaries generated on some other well-trained substitute models. The phenomenon is also known as the transferbility of adversaries. This provides an insight of similarity between different classifiers working on the same tasks. Studies are done on the explanation of original cause of the adversaries existence through the input dimension and non linearity of the neural network model. Some studies are on the instability measurement of the given model. In addition, the usage of adversarial examples is further than only make the network misclassify. Recent work shows that the neural network can be forced to perform absolutely different functions on the adversarial input, which called the adversarial reprogramming of neural networks [4]. Thus, studies on explaining the existence of adversaries and how to prevent them are becoming more important nowadays.

Adversarial attacks show the undetected weakness in tradition neural networks, as shown in Figure. 1.1. Many defence methods are proposed to protect against adversaries. Many studies are focused on using more complex structures to improve the robustness of the main classifier; 2. adversarial training: add adversarial examples into training dataset [5]; 3. increasing generalization ability of networks by applying unsupervised and semi-supervised learning algorithms.


(a)

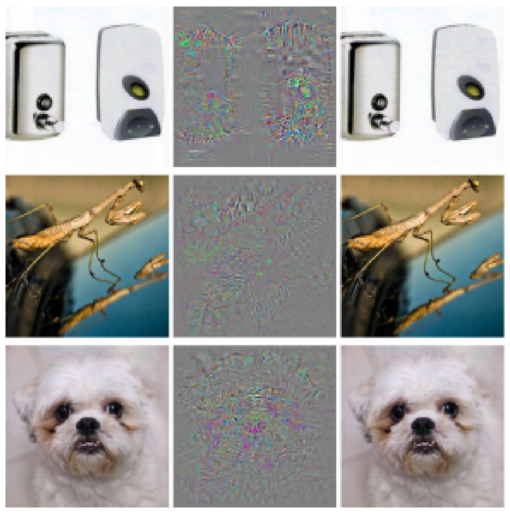

(b)

Figure 1.1: An illustration of adversarial examples ((reprinted from) [1], use permitted under the Creative Commons Attribution License CC BY 3.0) generated on AlexNet [2]. All images in the right columns are misclassified, with only small perturbation added on the corresponding images in the left columns.

1.2 Motivation

Current research shows that the threaten levels of different attack methods are commonly not the same. Fast-gradient sign method (FGSM) is one of the well-known adversarial attack methods based on gradient calculation. Using adversarial training, the model can perform well against the adversaries generated by FGSM. However, when applying another more threatened attack, projected gradient descent (PGD) method, which bases on the iterative multi-step FGSM, the model will be successfully broken. Experiments show that the model using adversarial training with adversaries generated by PGD can provide a great protection against several different gradient based attack methods. However, the adversarial training process becomes very slow due to the request of multiple times gradients computations when generating PGD adversaries during each training iteration.

This work discusses the existence of adversarial examples, digs into how single step and multi-step gradient based adversarial attacks work on the target convolutional network. The paper proposes a new adversarial training method, called Dynamical Step Adversarial (DSA) training, which only requires 2 steps gradient calculation to generate adversaries for training. Along with other adversarial training techniques, DSA performs well against FGSM and PGD attacks. Moreover, DSA provides good protections for the neural network against black-box attacks.

2. EXISTING ATTACK METHODS

This chapter gives a view of many popular adversarial attacks focusing on fooling deep learning neural networks in computer vision tasks. The existence of small perturbation on the input images can easily make deep learning models into misclassification [1]. It is also claimed that there exists some kind of universal perturbations that can make a neural network misclassify a group of inputs simultaneously ([6]). This section gives a brief review of several widely used adversarial attacks.

2.1 Gradient Based Adversarial Attacks

2.1.1 Fast Gradient Sign Method (FGSM)

According to [3], an efficient method to compute an adversarial sample x_{adv} on the input image x is by adding a perturbation given by:

$$x_{adv} = x + \varepsilon sign(\nabla_x J(\theta, x, y)) \tag{2.1}$$

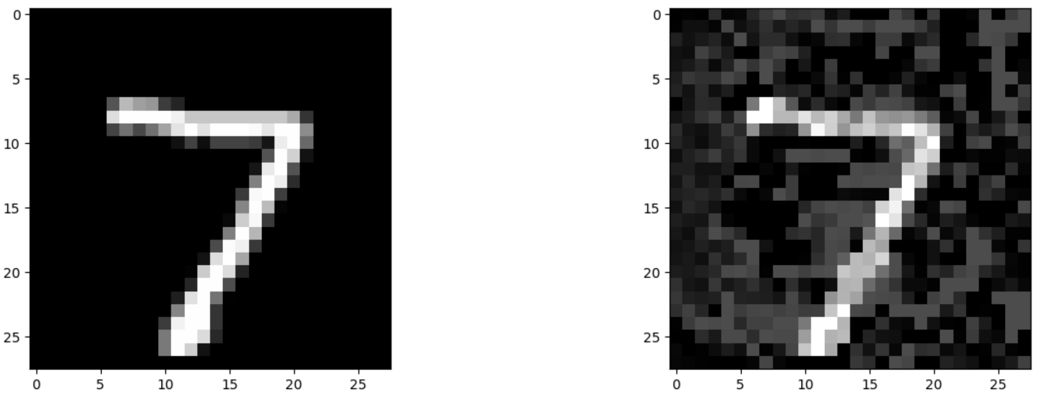
where ε is the size of perturbation. The attacker computes the gradient of the loss function respect to the input data x according to the current model parameters θ and target label y. The single step of gradient computation of this attack methods makes it very efficient but highly threaten to unprotected network. The chosen loss function is expected to be maximized by the computation of sign function under l_∞-norm measurement. The parameter ε represents the restriction of l_∞-norm of perturbation value in this place. FGSM can fool the network by solving the optimization problem of maximizing the loss function through calculating the gradients of the loss respect to the input image. Some work [7] also claims that using a specific label y_{target} as the target, the probability of predicting the output class to be y_{target} will also be maximized. They also suggests that y_{target} can be selected randomly to fool the network.

2.1.2 Iterative FGSM and Projected Gradient Sign Method (PGD)

To improve the attack level of one-step FGSM, an iterative multi-step search method called Iterative Fast Gradient Descent Method (IFGSM) is proposed . Set $x_0 = x$ which is the original clean image, the i'th perturbed image x_i is calculated iteratively as following:

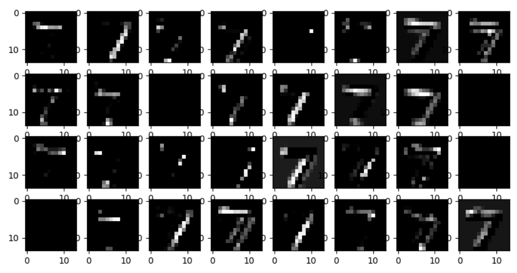
$$x_{i+1} = Clamp_{x,\varepsilon}(x_i + a \cdot sign(\nabla_x J(\theta, x, y))) \tag{2.2}$$

where the size of each step is determined by the hyperparameter a, normally set to be $a = \frac{\varepsilon}{k}$. To ensure that the total restrict l_∞-norm will be no more than ε, Clamp function clips the image into the range of no more than ε around the clean image. This process will iteratively repeat for k steps which is one of the hyperparameters setup before attacks. The value of a can also be set larger than $\frac{\varepsilon}{k}$ to enlarge the total magnitude of the whole iterative process. Another work [5] also proposes that by adding a uniform noise with size to be ε can universally increase the searching area of the IFGSM adversaries, performing a much higher threaten level on the target model.

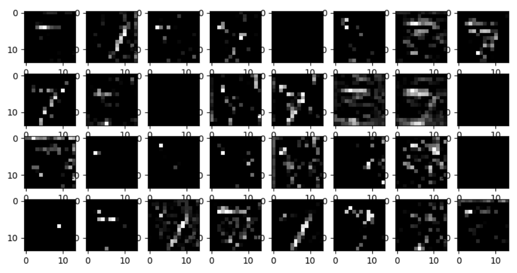


(a) (b)

Figure 2.1: A clean image and its corresponding adversarial example generated by PGD attacks on an unprotected well-trained model.



(a)



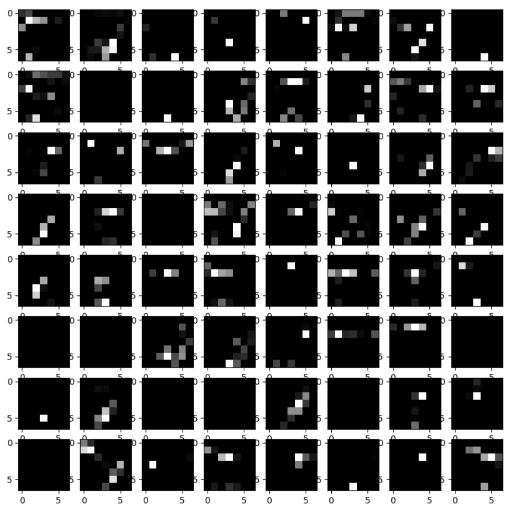
(b)

Figure 2.2: The output of the first convolutional layer and according to a clean image and its corresponding adversarial example generated by PGD attacks on an unprotected well-trained model.

To illustrate how PGD adversarial attack works on a single image, the MNIST handwriting dataset [8] is used for generating adversaries. A simple model with 2 convolutional layers and 2 fully-connect layer is well trained to obtain 99% accuracy on the test dataset. There are totally 32 filters at the first convolutional layer and 64 filters at the second layer, of each the kernel size is to be 5×5, and followed by ReLU activation function and 2d max-pooling layer. The fully-connect layers have 1024 hidden neurons. The model is only trained with clean MNIST training dataset. Figure 2.1 shows how the perturbation generated by PGD on the given model looks like. Human can easily describe that both images represent the digit "7", while the well train model can only predict the clean image correctly, with adversaries hardly classified.

Through the inner process of the chosen model, the output of the first convolutional layer (after ReLU activation function and max-pooling layer) is compared in Figure 2.2. It can be seen that the perturbation successfully goes cross the convolutional layer and max-pooling layer and distorted parts of channels, but the outline of the digit number can still be distinguished from parts of kernel outputs. Many stroke features are kept during this period. However, the output of the second convolutional layer has great difference from the clean one, shown in Figure 2.3.

The difference in Figure 2.3 is dominant. Most parts of the kernel output have different place of spotlight, which should be the corresponding features at that place. It is obvious that the fully connected layer following the second convolutional layer can hardly work on the distorted inner


(a)

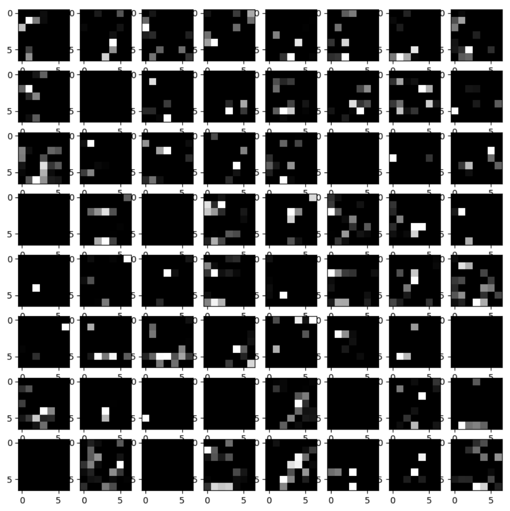

(b)

Figure 2.3: The output of the second convolutional layer and according to a clean image and its corresponding adversarial example generated by PGD attacks on an unprotected well-trained model.

result. Thus, the neural network makes a misclassification.

2.1.3 Adversarial Attacks with Momentum

Another adversarial training method proposed by [9] integrates the momentum into the IFGSM, which can strongly improve the success rate of attacks. The method uses momentum in IFGSM training strategies to escape from local maximum and searches for stronger adversaries. Instead of using gradients to determine direction straightforwardly, accumulating the velocity vector in the gradient direction is implemented to improve searching abilities:

$$g_{t+1} = \mu \cdot g_t + \frac{\nabla_{x_t} J(\theta, x_t, y)}{\|\nabla_{x_t} J(\theta, x_t, y))\|} \tag{2.3}$$

where g_t is a accumulating gradient at iteration t, initialized to be $g_0 = 0$ and x_t is the adversarial example at the t'th iteration. Then the next adversarial point x_{t+1}will be update by:

$$x_{t+1} = x_t + a \cdot sign(g_{t+1}) \tag{2.4}$$

This work can also be applied to an ensemble of models by combining the logits of all ensemble models to determine the loss function for attack. Also it discusses the transferability of MI-FGSM which is better among different substitute models by escaping the local maximum existing in the loss surface. The team won the first place in both the non-targeted attack and targeted attack competition in NIPS 2017 Adversarial Attacks and Defenses Competition.

2.2 Carlini and Wagner Attacks

Carlini and Wagner [10] introduce a group of adversarial attack methods by restricting l_0, l_∞ and l_2 norms to generate perturbation on the given input image, as known as C & W attacks. They demonstrate that the adversarial training does not significantly increase the robustness. The three distance measurement metrics are more effective against protected network compared with previous attack methods, due to that common methods normally focus on defend 1 or 2 of the three Lebesgue measurement attacks. As their work mentions, the general objective function can be formally defined as following:

$$\begin{aligned} &\text{minimize } D(x, x+\delta) \\ &\text{such that } C(x+\delta) = t \\ &\qquad\qquad x+\delta \in [0,1]^n \end{aligned}$$

They formulate an appropriate optimization instance which can be solved with existing optimization algorithms. A list of possible choices of different functions which are suit for solving the optimization problem is given. Also the quality of these functions are compared to demonstrate that some are better than others.Using thel_0, l_2 and l_∞ norms based attacks, they successfully fool the distilled network with 100% success rate, on which many other attack methods fail.

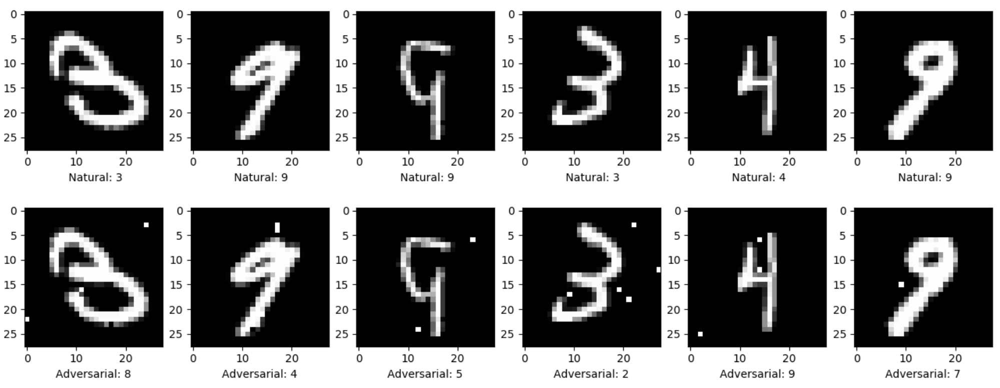


Figure 2.4: Pointwise attack experiments on the model using adversarial training with PGD adversaries (under 0.3 l_∞-norm and 30 steps) using MNIST dataset. The top row represents the clean images and the bottom row for the corresponding adversarial examples. Every distortion on a single image is under l_0-norm with no more than 5 pixels perturbed (either set to 0 or 1). The adversaries are easily distinguishable to human while all are misclassified by the well trained model.

2.3 Decision Based Attacks

Decision based attacks do not rely on the gradient information. A perturbation is added initially on the clean image and then the attacker will dynamically minimize the distance between the distorted image and the original image while keeps the distorted image adversarial to the classifier. Among typical decision based attack methods, boundary attack is proposed to minimize the l_2-norm between the adversarial image and the clean image [11]. Meanwhile, researchers argue that some simple noise can also successfully fool the classifier. For instance, salt&pepper noise can be simply used to generate pixel perturbation based on l_0-norm.

Another kind of decision based attack is Pointwise attack, mentioned in [12] using greedy algorithm to minimize the l_0-norm. The algorithm first generates salt&pepper noise on the clean image. Through probability adjustment, the total number of distorted pixels in salt&pepper noise will be reduced until the image can be classified correctly. Then A deep-first search will be applied through greedily setting each distorted pixel clean and verify whether result adversarial or not. The searching process will continue until exceeding the maximum searching steps. Commonly the output will have less distorted pixels than expectation (the number of distorted pixels). In the circumstance that the decided noise level can not be approached, an intermediate result with minimum number of distorted pixels will be chosen and randomly set parts of its distorted pixels clean so that total number of distorted pixels will be equal to the expect noise level. Then this version of the distorted image will be viewed as the corresponding adversarial example for the original one.

Figure 2.4 shows the Point Wise attack result on a neural network protected with Projected Gradient Descent (PGD) [5] adversarial training (one of the most popular protection methods by inserting online generated adversaries into training dataset to protect against attacks, which will be discussed in the following chapter). The distortion of the Point Wise attack is limited to 5 pixels. Practically, most of the adversarial images only have 2-4 distorted pixels, other than 5. All adversarial images are misclassified by the network. In conclusion, the experimental results reveal the vulnerability of the model while being attacked with a different norm measurement from the one applied in adversarial training.

2.4 Transferability and Black Box Attacks

Transferability is a property that adversarial examples generated for a substitute model concurrently has high probability to be misclassified by another model [1]. This indicates the existence of high vulnerability in the neural network even its parameters and structures are unknown to the attackers. There are many studies on this property ([1], [3], [13]). The transferability of adversarial examples can also perform a effect on different images. Some work shows the existence of universal perturbation which can distort a bunch of images simultaneously [6]. Targeted and non-targeted examples are also studied [14] over ImageNet [15], which is thought to be more complex than simple datasets like MNIST [8] and CIFAR-10 [16], in order to illustrate the strength of the current black-box attacks.

3. EXISTING DEFENSE METHODS AGAINST ADVERSARIAL ATTACKS

The defenses against adversaries can be categorized into three main directions according to the strategies:

- Using new training examples generated from original dataset, commonly are adversarial examples, as well as Data prepossessing, squeezing, transformation or compression.
- Modifying the structure of the network.
- Add co-network to detect, generate or squeeze data.

3.1 Adversarial Training and Data Pre-processing

3.1.1 Related Works

For the first category, new training data is generated based on the given clean data input. An typical type among different generating methods is to introduce brute-fore adversarial algorithms during training process to generate unseen adversarial data for the current model. Some work ([3], [17]) argues the principles behind the adversarial training is to reduce the overfitting by regularizing the network. To improve robustness against bunch of different existing adversarial attacks, algorithms for generating stronger adversaries are introduced in many works. Also, Data augmentation like adding random noise or prepossessing can also provide some protection for the network through adversarial training. However, It is shown that adversarial examples can still be searched out in an adversarial-trained models [18].

3.1.2 Gradient Based Defense Methods

The major part of the adversarial training defense method is to generate adversaries according to the gradients. Researchers find that the adversaries generated through FGSM can efficiently protect the network [3]. The adversarial examples are produced on each mini-batch data and then included into the training process. Experiments show that FGSM adversarial training can easily protect the network from one-step gradient based attack.

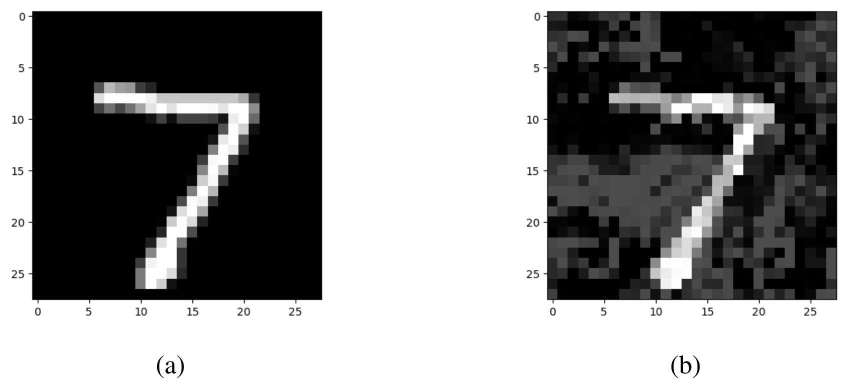


Figure 3.1: A clean image and its corresponding adversarial example generated by PGD attacks on the PGD adversarial training model.

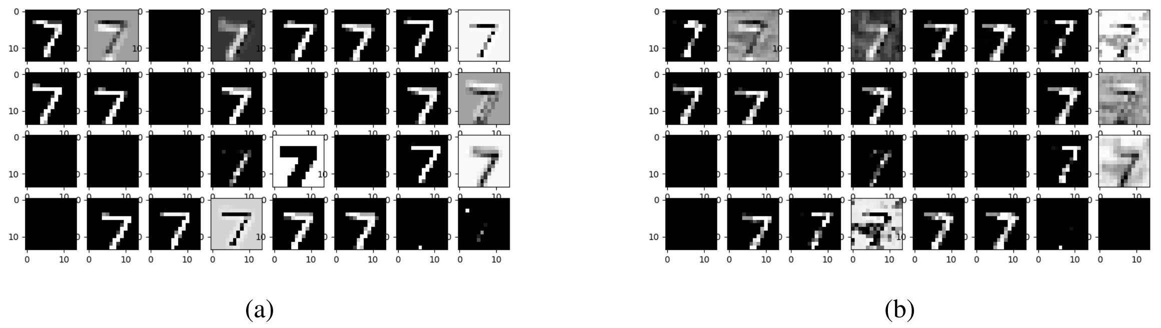


Figure 3.2: The output of the first convolutional layer and according to a clean image and its corresponding adversarial example generated by PGD attacks on the PGD adversarial training model.

Nevertheless, some work [5] claims that by generating strong adversarial examples using PGD attack method can protect neural network very well. Similar to the IFGSM adversarial attack method, PGD adds a random uniform noise on the starting cleaning image in the range of $[-\varepsilon, \varepsilon]$, then continues the iterative searching process. The random perturbed noise can easily bring the clean data to a wider region where adversaries may exist and hard to be found. The model trained with PGD adversaries stay robust against a group of gradient based attacks. As experiments showing, models trained with PGD performs no vulnerability under the attacks by FGSM, IFGSM and MI-FGSM.

To illustrate the protection ability of PGD on a simple CNN classifier, MNIST dataset and a neural network trained with PGD adversaries are used. Figure. 3.1 shows the same image which are misclassified in Chapter 1 by applying PGD method. In this experiment, the attack method is still PGD unless the model has been replaced with the one using PGD adversarial training. As it shown that the perturbation on the clean image is nearly the same as the one generated on unprotected model. However, the output of the first convolutional layer, shown in Figure. 3.2 from the adversarial example is nearly the same as the one from the clean version. The boundary of the digit with respect to the background at this level is much more bright than the one in unprotected network. The same phenomenon also occurs at the second convolutional layer output, shown in Figure. 3.3

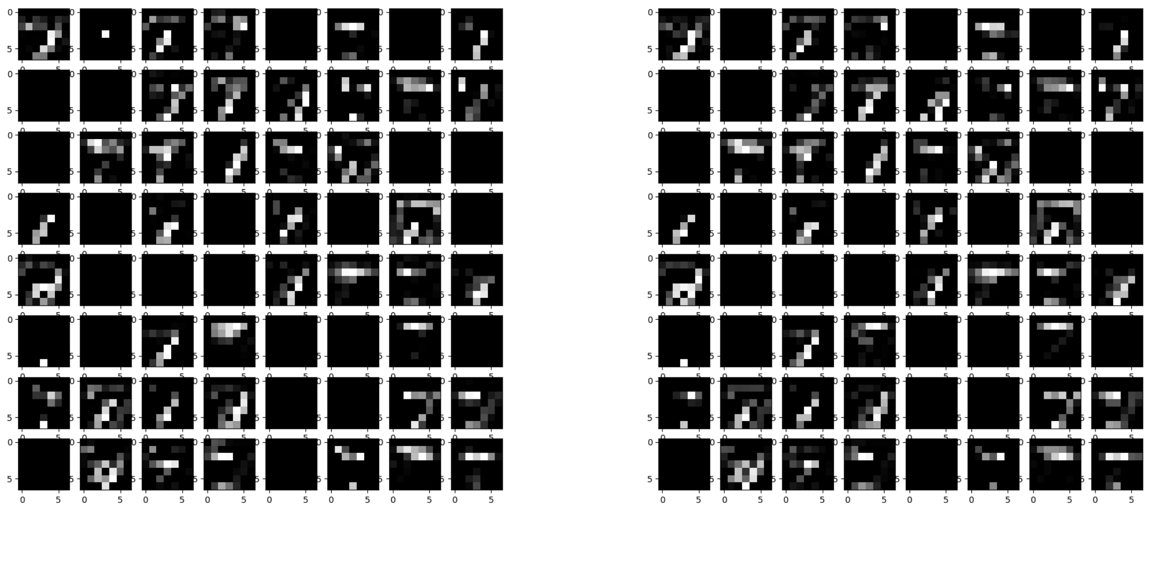

(a) (b)

Figure 3.3: The output of the second convolutional layer according to the clean image and its corresponding adversarial example generated by PGD attacks on the PGD adversarial training model.

After more precise comparison with unprotected network, several interesting differences are discovered. As discussed above, the output at first convolutional layer of the model using PGD adversarial training seems to have higher resolution value than unprotected network. However, the features here are less distinguishable compared with unproteced version, which seems to extract more high level features at this level. Instead, each filter of the PGD adversarial training one attempts to generate a clean version of the input image with some squeezing or minor transformation.

Notice that multiple filters try to generate anti-color version of the input image, by reversing the pixel value on background and digit. Consider the valid pixel value on original input, the only solution is to have a very high negative weight filters with a balanced bias. The pixel with value 1 which stay on the body of the digit will be convoluted to the negative domain, then set to 0 by the ReLU activation function. This will not occur on the background pixels, most of which are with value 0. Thus, the image will be reversed into anti-color. The bias will automatically increase the ratio of signal to noise at that output. This can probably explain the robustness of PGD adversarial training models. Also the second output still maintains the same feature as the

3.2 Structure Modification

For the second category, the typical methods are gradient regularization and masking by shrink the information leaking from gradients. This method is commonly combined with adversarial training which can perform good against attacks like FGSM [3]. In some cases, the gradient masking can fail due to the local gradient information when applying stronger attack methods which can solve more global optimization functions [5]. Another method is to use a sub-network with binary classification output as the detector of detecting the existence of the adversaries [19]. This method is proved to be vulnerable by applying adversaries fooling both sub-network and the main classifier. This can also be classified as the co-network according to the framework relation with main classifier.

3.3 Co-Network Defense Strategies

The third category contains mainly two kinds of co-network: Defense-GAN network and detector sub-network. Generative Adversarial Network (GAN) is proposed by Ian Goodfellow at 2014 [20], which soon became widely used in different areas of machine learning. Defense-GAN [21] can be used with any chose classification model and used to defense against any possible adversarial attacks. The basic idea is to train GAN to find input can minimize the distance between the output of GAN and the given image. The perturbation found by the GAN will be used as adversarial inputs to train the main network to correctly classify both of them. Detector network is similar to some defense strategies in the second category, but adding co-networks to detect adversarial existence [22]. Whenever the existence of adversarial perturbation in the input image is detected, the network will choose to either filter the noise or refuse to make prediction.

4. DYNAMICAL STEP ADVERSARIAL TRAINING DEFENSE METHOD

4.1 Motivation

Among all gradient based adversarial attacks, PGD introduces a high level of threaten. It is claimed [5] that PGD can be used to protect the network from broad gradient based attacks. However, traditional PGD and IFGSM require iterative backpropagation determined by the selecting of the number of step k. Typically k will be at least 20 which brings a higher cost of computation. Also some over-fitting problems exist in the model trained with PGD adversaries, which is still vulnerable to well-chosen attack methods. Experiments show that [12] PGD may be weaker than even FGSM under some other attack methods. This encourages to propose a more robust and efficient method for adversarial training.

4.2 Dynamical Step Adversarial Training

To solve the problem that the total L_{inf}-norm of the adversarial examples generated by PGD is not strongly reach the given perturbation size, it is wise to adjust the size of the perturbation at each step of searching next data point in PGD. For the fact that PGD goes with the constant size of perturbation during each iteration of searching the adversaries, it can goes very slowly and inefficiently through loss surface around the given data point. Adjust the size of the perturbation dynamically at each step will strongly accelerate the searching ability and strongly push the search direction to the perturbation boundary (the given L_{inf} size, typically 0.3 for the MNIST dataset).

To dynamical adjust the size of each step, the loss of each current starting point and the point on its searching direction will be calculated. Figure 4.1 shows the how this process will be done at each step.

The center point is the given data point and P_0 is the point starting from the given data with a constant size of random noise. This will be used as the starting point to generate adversaries for training. The size of the first step will be constantly set to $a = \frac{\varepsilon}{k}$ where k is the total number of steps the algorithm will go. The searching direction will be generated by calculating the gradients of the cross entropy loss respect to the start data point. Then the partner of adversarial points can be obtained by using single step of FGSM. The first search step is as following

$$x_1 = x'_0 + a \cdot \nabla_{x_0} L(y, f(x_0, \theta)) \tag{4.1}$$

where x'_0 is made by adding a cube noise on the given training data point, and x_1 is the adversarial data generated from the corresponding data point x'_0. The cube noise is added by randomly chosen values from $\{-\delta, \delta\}$ (either positive or negative) on each pixel of the given image. After the first step, the size of the step will be dynamically changed according to the loss of current starting point x'_s and the loss of its correspondingly adversarial data $\tilde{x_s}$. Scaling factor η will be applied on the step size a as

$$\eta = \frac{L_{\tilde{x_s}} - L_{x'_s}}{L_{x'_s}} \tag{4.2}$$

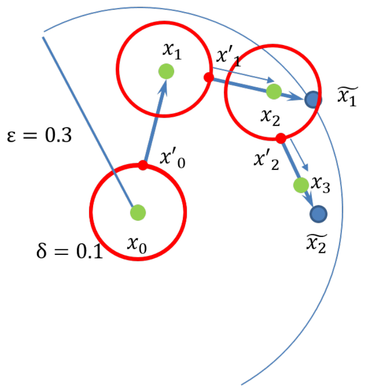


Figure 4.1: The training process of DSA. The total perturbation size is chosen to be $\varepsilon = 0.3$, and the size of noise add on each starting point is set to be $\delta = 0.1$. Each red circle represents the noise bound corresponding to the starting point x_i, and x'_i is the point with random generate noise around x_i. x_0 is the clean images which is prepare to be trained from a single minibatch of data. From x'_i, one-step FGSM is applied to find the corresponding adversarial point $\tilde{x}_i$. x_1 is the same as $\tilde{x}_0$. After that, each x_{i+1} will be dynamically generated with the loss of x'_i and $\tilde{x}_i$ according to the expression of η.

This dynamically changes the size of each step according to the magnitude of the loss difference between current starting point and its correspondingly adversarial point. The data points generated after first step will be

$$x_{s+1} = x'_s + \eta \cdot a \cdot \nabla_{x_s} L(y, f(x_s, \theta)) \tag{4.3}$$

This is an interpolation of creating new data point withing the range of known the loss of the starting point and ending point during a single FGSM process.

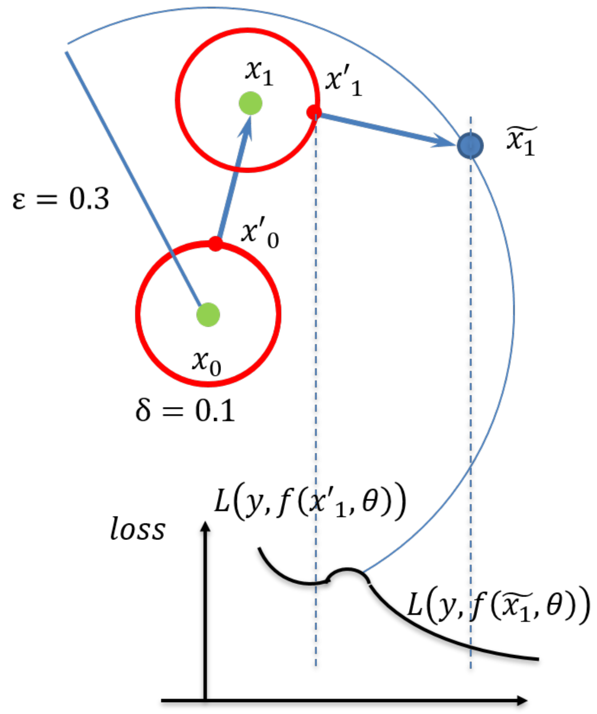


Figure 4.2: DSA can avoid local maximum around x'_i when $L_{\tilde{x}_s} < L_{x'_s}$, which may lead to a wrong searching direction by traditional gradient based adversaries.

Notice that η in DSA will be negative if $L_{\tilde{x}_s} < L_{x'_s}$, which indicates that the current searching direction predicted by the gradient may not be optimized. The negative direction can avoid the wrong searching caused by the existence of the local maximum around x'_i and expect to get a higher loss on the opposite direction, as shown in Figure 4.2. The local maximum misleads the increasing direction of the loss surface, while the calculation of η will avoid it and choose the opposite direction.

Algorithm. 1 shows the pseudocode of generating DSA adversaries during each training iteration and the expression of combining all intermediate results in the total loss function. For each image, a list of data points will be recorded during the DSA searching process, and the total loss function will be the sum loss of the original image and its corresponding list of data points along the searching path. Then back propagation will be computed according to the total loss, and the optimizer will be used to update the parameters of the neural network.

Algorithm 1 Dynamic-step Adversarial Training Loss

```
Input: Input batch data $(X, Y)$; model parameter $\theta$; searching parameters $k, \varepsilon, \delta$
Output: Total L for training $L_{total}$
 1: $L_{total} \leftarrow 0$
 2: for each $(x_i, y_i) \in (X, Y)$ do
 3:     for $s \in [0, k-1]$ do
 4:         $x'_{i,s} gets x_{i,s} + \delta \cdot rand([-1, 1]^N)$
 5:         $\tilde{x}_{i,s+1} \leftarrow x'_{i,s} + a \cdot sign(\nabla_{x'_{i,s}} CE(y_i, f(x'_{i,s}, \theta)))$
 6:         $L_{x'} = CE(y_i, f(x'_{i,s}, \theta))$
 7:         $L_{\tilde{x}} = CE(y_i, f(\tilde{x}_{i,s}, \theta))$
 8:         if i == 0 then
 9:             $\eta \leftarrow 1$
10:         else
11:             $\eta \leftarrow \frac{L_{\tilde{x}} - L_{x'}}{L_{x'}}$
12:         end if
13:         $x_{i,s+1} \leftarrow x'_{i,s} + \eta \cdot a \cdot sign(\nabla_{x'_{i,s}} CE(y_i, f(x'_{i,s}, \theta)))$
14:         $L_{total} \leftarrow L_{total} + CE(y_i, f(x_{i,s+1}, \theta))$
15:     end for
16: end for
```

5. EXPERIMENT RESULTS

5.1 Setups

MNIST handwriting dataset is used to measure the performance of different models by applying several widely adopted attack methods.. It contains 60,000 training images and 10,000 testing images, sampled from human volunteers' handwriting digits from 0 to 9. It is a simple but widely used dataset in image recognition tasks. A well-trained convolutional network has high performance on the test dataset with only 0.5% error rate. However, it can still be easily fooled under adversarial attack.

The machine learning platform chosen for training and attacking is Pytorch [23], which is an open source deep learning platform developed by Facebook. The optimizer used to update parameters for all models is chosen to be Adam with learning rate to be 0.001. The loss function used for measuring different between output and one-hot target on a single batch of data is chosen to be cross-entropy loss. All Models chosen to be compared are listed as following

- Models trained with FGSM adversaries generated by using perturbation of size $\varepsilon = 0.3, 0.4$
- Models trained with PGD adversaries with perturbation of size $\varepsilon = 0.3$ and step size of $k = 30$
- Models trained with Dynamic Step methods with perturbation of size $\varepsilon = 0.3$ and step size chosen to be $k = 2$, which leads a to be 0.15. The size of the cube noise is set to be 0.1.

The architecture of each training model is shown in Figure 5.1. All the models shares the same architecture: consisting of two convolutional layers with 32 and 64 filters respectively, each followed by a 2×2 max-pooling layer and ReLU activation function, and two fully connected layers to be $3136 \times 1,024$ and 1024×10 respectively. After the raw output of the last fully-connected layer, there is a softmax layer generating logits output. All models are trained with 128 epochs on MNIST training dataset and the minibatch is chosen to be 256 images randomly chosen

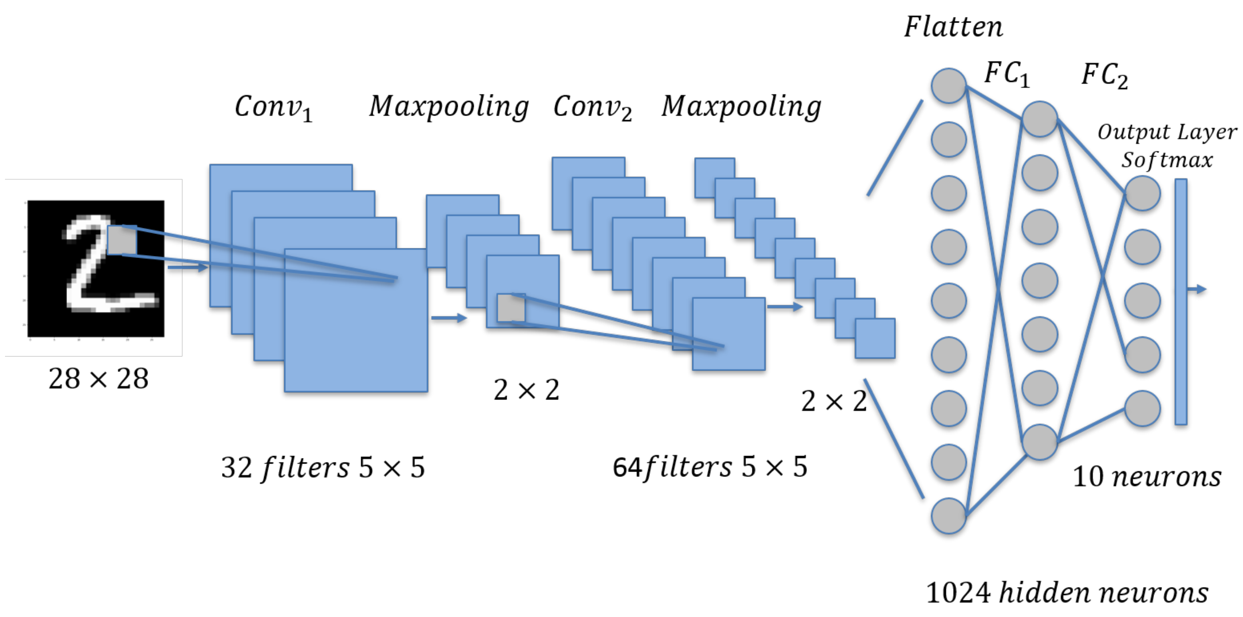


Figure 5.1: Architecture of the training model.

from the remaining part of the training dataset. After each training epoch, the training dataset will be randomly shuffled. During each training iteration, the loss functions of the models trained with FGSM and PGD adversaries consider both clean image and the adversaries generated by the corresponding attack methods. These two data points contribute simultaneously to the final loss function, for instance, the one used in FGSM adversarial training is:

$$L_{FGSM} = CE(y, f(x, \theta)) + CE(y, f(x_{FGSM}^{adv}, \theta)) \tag{5.1}$$

where x and x_{adv} represents the clean data and adversarial example respectively; y to be the target and θ the current parameters of the model; the loss function is chosen to be cross entropy loss CE. This guarantee that the generated adversaries will not distort the accuracy on clean image.

The adversarial attacks applied on all chosen models are:

- White-box attack with FGSM under perturbation of size $\varepsilon = 0.2, 0.3, 0.4$.
- white-box attack with PGD under perturbation of size $\varepsilon = 0.3, k = 30$.

- Black-box attack using three different well-trained models: trained only with clean images; trained with clean images and adversaries generated by FGSM with $\varepsilon = 0.3$; trained with clean images and adversaries generated by PGD with $\varepsilon = 0.3, k = 30$. Two attack methods will be applied to substitute models: FGSM with $\varepsilon = 0.3$ and PGD with $\varepsilon = 0.3, k = 30$.

All attacks are applied on the MNIST test data with 10,000 test images.

5.2 White-box Attacks

The performance of all trained models under two type of white-box attacks are shown in Table. 5.1. All models reach over 99% accuracy on clean dataset. Unsurprisingly, the performance of the natural model trained only with clean data drops a lot when applying either attack methods. PGD as a more offensive method, distort every image in the test dataset. The model obtained via FGSM adversarial training can easily protect against FGSM adversarial attacks. However, PGD can easily break FGSM trained model by decreasing its accuracy to about 10%. PGD adversarial training can maintain at very high accuracy both under FGSM and PGD attacks.

Table 5.1: Accuracy of different models under white-box adversarial attacks evaluated using MNIST. Rows are the attacks where “Clean Data“ in the first row represents the accuracy of clean dataset . Columns report the accuracy of different defense solutions where ”Natural” means the baseline model without any additional defense strategy, and “FGSM Adv. Train" and “PGD Adv. Train" mean the adversarial training with FGSM and PGD, respectively.

Attack		Natural	FGSM		PGD	DSA
			$\varepsilon = 0.3$	$\varepsilon = 0.4$	$\varepsilon = 0.3, k = 30$	$\varepsilon = 0.3, k = 2$
Clean Data		0.9942	0.9938	0.9921	0.9913	0.9915
FGSM	$\varepsilon = 0.2$	0.4427	0.9714	0.9494	0.9695	0.9654
	$\varepsilon = 0.3$	0.1741	0.9768	0.9658	0.9519	0.9595
	$\varepsilon = 0.4$	0.1027	0.9136	0.9732	0.8152	0.9364
PGD	k=30, $\varepsilon = 0.3$	0	0.0038	0.0298	0.9484	0.9006

The model trained with DSA algorithms easily protect the network against both FGSM and

PGD adversarial attacks. It reaches as high as the accuracy of PGD adversarial training model under FGSM attacks with different perturbation size. And the accuracy does not drop a lot when applying FGSM attack with $\varepsilon = 0.4$. Compared with FGSM adversarial training with the same l_∞-norm restriction (FGSM training of $\varepsilon = 0.3$), DSA still keeps a performance about 2% higher. Under PGD attack, the accuracy still maintains a high level accuracy above 90%, which makes it every competitive with the current PGD adversarial training. Notice that DSA

There are two interesting things in the results. One is that when applying FGSM attack with perturbation size of $\varepsilon = 0.2, 0.3, 0.4$ on the model trained with the adversaries generated by $\varepsilon = 0.4$ FGSM, the accuracy is higher at larger ε value. FGSM adversarial training with $\varepsilon = 0.4$ performs about 3% better under $\varepsilon = 0.4$ FGSM attack other than under even a smaller perturbation size FGSM attack (FGSM with $\varepsilon = 0.2$). This does not append in the natural model, PGD adversarial training model and DSA training model, which indicates the possibility of the existence of overfitting. The FGSM adversarial training only protects the model against matched adversarial attacks, that is, the attacker chosen the same size of perturbation ε as it used for generating adversaries during training process

Another thing is that the PGD adversarial training model, considered to be the most robust model against different gradient-based adversarial attacks, drops a lot while applying FGSM attack with $\varepsilon = 0.4$. Compared with FGSM adversarial training model with the perturbation size $\varepsilon = 0.3$, PGD adversarial training only reaches about 80%, which drops about 14% according to the performance under FGSM attack with $\varepsilon = 0.3$. Similarly, under FGSM with $\varepsilon = 0.3$, PGD adversarial training has the lowest accuracy through all adversarial training models. Possible explanations are that the total l_∞-norm of the PGD adversaries after the multi-step iteration can seldom reaches the boundary determined by ε. Due to the direction of the gradients on each step will are not strongly towards the boundary (l_∞-norm $which is \varepsilon = 0.3$), adversaries generated by PGD commonly stay insider the boundary.

Substitude Model	Natural	FGSM Adv. Train		PGD Adv.Train	DSA
		$\varepsilon = 0.3$	$\varepsilon = 0.4$	$\varepsilon = 0.3, k = 30$	$\varepsilon = 0.3, k = 2$
Natural Model	0.1738	0.9258	0.9314	0.9713	0.9656
FGSM $\varepsilon = 0.3$	0.8444	0.9768	0.9214	0.967	0.9647
PGD $\varepsilon = 0.3, k = 30$	0.8643	0.9324	0.9119	0.9519	0.9629

Table 5.2: Accuracy of different models under black-box FGSM attacks evaluated using MNIST dataset.

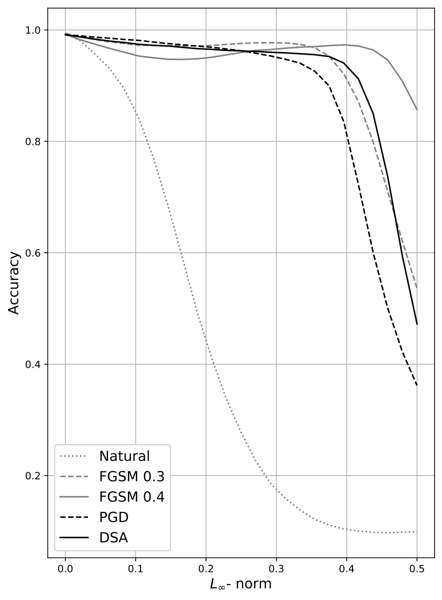
Substitude Model	Natural	FGSM Adv. Train		PGD Adv.Train	DSA
		$\varepsilon = 0.3$	$\varepsilon = 0.4$	$\varepsilon = 0.3, k = 30$	$\varepsilon = 0.3, k = 2$
Natural Model	0	0.952	0.9377	0.9805	0.9778
FGSM $\varepsilon = 0.3$	0.9425	0.0042	0.8582	0.979	0.9757
PGD $\varepsilon = 0.3, k = 30$	0.8592	0.9277	0.9157	0.9484	0.9627

Table 5.3: Accuracy of different models under black-box PGD attacks evaluated using MNIST.n

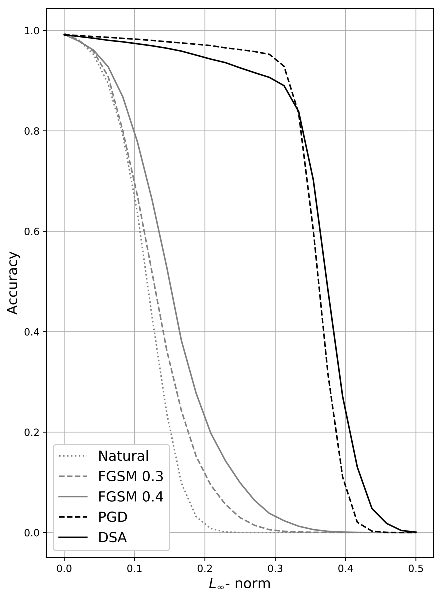
5.3 Black-box Attacks

To apply black-box attacks, several well-trained models are chosen as substitute model to generate adversaries: the model only trained with clean images; the model trained using adversaries generated by FGSM with perturbation size of $\varepsilon = 0.3$; the model trained using adversaries generated by PGD with perturbation size of $\varepsilon = 0.3, k = 2$. In order to observe the strength of different types of adversaries, two adversarial attacks are selected to apply on the substitute models respectively.

Table. 5.2 shows the result when applying FGSM attacks with $\varepsilon = 0.3$. Table. 5.3 shows the result when applying PGD attack with $\varepsilon = 0.3, k = 30$ on substitute models to generate adversaries used for attack the defense model. Compared with other models, the model trained with DSA keeps the highest level of accuracy under the adversaries generated from all substitute models. It can be seen that DSA and PGD adversarial training has higher accuracy than the model only trained with clean images and the model trained under different perturbation sizes.



(a)



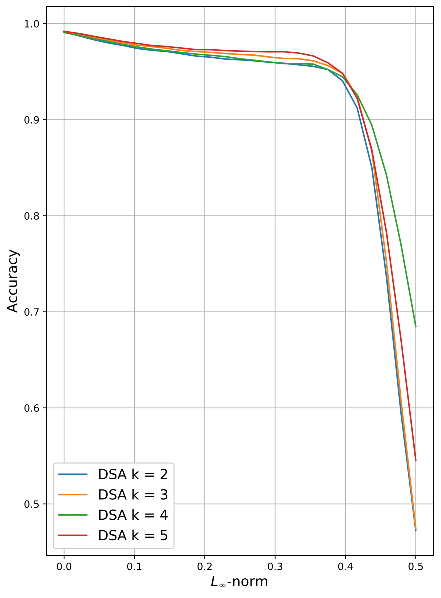
(b)

Figure 5.2: Accuracy changes according to the increasing of L_∞-norm under different attack methods . FGSM attack method is applied in (a) and PGD in (b). The model with DSA adversarial training stays at high accuracy through the wide region under L_∞-norm attacks.

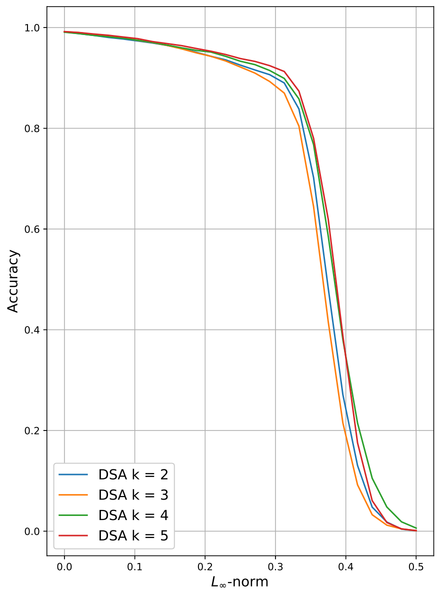
5.4 L_∞-norm Attack Illustration

To illustrate how the accuracy of the model changes respective to the perturbation size ε increases, the plot of accuracy vs. L_∞-norm is shown in Figure 5.2.

Under both attacks, PGD maintains a very high accuracy before L_0-norm reaches $\varepsilon = 0.3$ which is used as the size of perturbation during training process. However, PGD drops very fast as the noise level increasing, leading to very low accuracy at high perturbation. DSA, trained within 2 steps, still performs as high as PGD. Beyond $L_\infty = 0.3$, the dropping of DSA is not as quick as PGD, which provides alter-protection even the perturbation of noise is much larger than it used for adversarial training. Under FGSM attacks, the model trained using adversaries generated by FGSM with $\varepsilon = 0.4$ stays at a high accuracy level even the size of the perturbation reaches 0.5. However, an accuracy valley is observed withing range of $[0, 0.3]$ on this model. The accuracy



(a)



(b)

Figure 5.3: Accuracy of the models trained with the different number of steps according to the increasing of L_∞-norm in different attack methods. FGSM attack method is applied in (a) and PGD in (b).

drops and then returns back to a the same level, indicating the existence of vulnerability of this model under smaller size of perturbation adversarial attacks other than larger size of perturbation. DSA is flat through the region of $[0, 0.3]$, and keeps almost the same level of performance compared with the model trained under $\varepsilon = 0.3$ FGSM adversaries. Therefore, the model trained with DSA is much more robust through different region under l_∞-norm perturbation.

All the adversaries generated by DSA discussed above sets $k = 2, \varepsilon = 0.3$. To illustrate the searching ability of DSA with different numbers of steps, k is set from $2, 3, 4, 5$and both FGSM and PGD attacks are applied on the well-trained model. The size of the cube noise is set to be $\frac{2}{3}a$ under different k. Figure 5.3 shows the accuracy vs. L_∞-norm under different k. It can be seen that at $\varepsilon = 0.3$, the model training with larger k will have higher accuracy. The model under $k = 5$ has 92% vs. 90% of the model under $k = 2$.

6. CONCLUSIONS

In this thesis, the existence of adversarial examples in the convolutional neural networks is discussed. Several modern adversarial attack methods and effective defense methods are introduced. Also, an insight of how gradient based attack methods work on the convolutional neural network is made through visualization on intermediate results of each layer. Project gradient descent method, as one of the powerful attack and adversarial training defense methods, is discussed. Then, in order to solve the high computation complexity of searching PGD adversaries, an efficient multi-step gradient based adversarial training algorithm is proposed. The algorithm dynamically changes the step size according to the loss of the points. The robustness of the model trained with DSA is shown by applying different adversarial attacks on the training model. MNIST dataset is setup to be the benchmark for training of the classifier and applying adversarial attacks. Two main gradient based attack methods are implemented as the white-box attacks and several well-trained models with different defense strategies are set up to be the substitute models in black-box attacks. Experiments and comparisons have been made through DSA and several other traditional adversarial training methods, illustrating the effectiveness and robustness of the new proposed defense methods. This paper also compared the performance of the neural network under different l_2-norm measurement attacks, illustrating that DSA is more robust within a wider measurement region. Moreover, DSA with different numbers of steps are trained additionally to show the improvements while increasing the total searching step numbers.

REFERENCES

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *CoRR*, vol. abs/1312.6199, 2013.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[3] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015.

[4] G. F. Elsayed, S. Shankar, B. Cheung, N. Papernot, A. Kurakin, I. J. Goodfellow, and J. Sohl-Dickstein, "Adversarial examples that fool both computer vision and time-limited humans," 2018.

[5] 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.

[6] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, P. Frossard, and S. Soatto, "Analysis of universal adversarial perturbations," *arXiv preprint arXiv:1705.09554*, 2017.

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

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

[9] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," *arXiv preprint*, 2018.

[10] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," *arXiv preprint arXiv:1608.04644*, 2016.

[11] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," *arXiv preprint arXiv:1712.04248*, 2017.

[12] L. Schott, J. Rauber, M. Bethge, and W. Brendel, "Towards the first adversarially robust neural network model on mnist," 2018.

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

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

[15] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[16] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," tech. rep., Citeseer, 2009.

[17] S. Sankaranarayanan, A. Jain, R. Chellappa, and S. N. Lim, "Regularizing deep networks using efficient layerwise adversarial training," *arXiv preprint arXiv:1705.07819*, 2017.

[18] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," *arXiv preprint*, 2017.

[19] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," *arXiv preprint arXiv:1702.04267*, 2017.

[20] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[21] P. Samangouei, M. Kabkab, and R. Chellappa, “Defense-gan: Protecting classifiers against adversarial attacks using generative models,” *arXiv preprint arXiv:1805.06605*, 2018.

[22] D. Meng and H. Chen, “Magnet: a two-pronged defense against adversarial examples,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 135–147, ACM, 2017.

[23] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” 2017.