Improving Robustness of Neural Network with Adversarial Training

by

Yajie Zhou

B.Sc., Simon Fraser University, 2016

Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in the School of Computing Science Faculty of Applied Sciences

© Yajie Zhou 2019
SIMON FRASER UNIVERSITY
Summer 2019

Copyright in this work rests with the author. Please ensure that any reproduction or re-use is done in accordance with the relevant national copyright legislation.
Approval

Name: Yajie Zhou
Degree: Master of Science (Computing Science)
Title: Improving Robustness of Neural Network with Adversarial Training
Examinining Committee: Chair: Greg Baker
Lecturer

Jian Pei
Senior Supervisor
Professor

Jiguoo Cao
Supervisor
Associate Professor
Department of Statistics and Actuarial Science

Jiannan Wang
Internal Examiner
Assistant Professor

Date Defended: May 22, 2019
Abstract

Deep neural networks have been applied in computer vision recognition and achieved great performance on many image classification tasks. However, deep neural networks are not as robust we expected all the time, and may be vulnerable to adversarial examples, which are images with some imperceptible changes to originals. In this work, we enhance the robustness of a given deep neural network using an accumulating adversarial training algorithm. We also propose an adaptive boosting method, a group sampling boosting method and a stochastic mini batch boosting method to boost the performance of the accumulating adversarial training algorithm. Moreover, we show that our proposed methods can enable a given deep neural network to protect against several adversarial attacking algorithms at the same time.

Keywords: Adversarial Training; Boosting; Deep Neural Network; Computer Visions
Dedication

To my family and friends.
Acknowledgements

First, I would like to express my deepest appreciation to my senior supervisor, Dr. Jian Pei, for his great patience, insightful guidance and endless encouragement through my Master studies.

My gratitude also goes to my supervisor, Dr. Jiguo Cao, for his constructive opinions on my research work. I also want to give my gratitude to Dr. Jianman Wang and Mr. Greg Baker for kindly serving and supports on my thesis committee and improves the quality of this thesis by providing valuable feedback.

Second, I would like to give many thanks to my dear friends for their accompany and encouragement over these years. A particular acknowledgement goes to Lingyang Chu, Chuancong Gao, Juhua Hu, Xiangbo Mao, Xiao Meng, Xiaojian Wang, Yu Yang, Zijin Zhao, Mingtao Lei, Xia Hu, Yanyan Zhang, Liang Zhao, Changbo Qu, Jinglin Peng, Pei Wang, Xiaoying Wang, Si Chen, Dan Jian, Yin Hu, Qiuyi Sun, Lirong Luo, Yao Li, Hui Gong, Meng Que and Xuezhu Song.

Finally, and most importantly, I would like to express my deepest gratitude to my parents, Heng and Ailing, for their endless support and trust during the past nine years. Also, I would like to thank my dear little sister, Yixuan Zhou, for being adorable and accompany with our dear parents. Especially, I would like to thank my husband, Xin Li, for his love, support and accompany in the past seven years and my future life. His support and encouragement is the true energy to me during these years. The sweetest acknowledgement goes to my dear son, Jayden Hongze Li, who decides to join my life in the last year of my master study and light up my life.
Table of Contents

Approval ii
Abstract iii
Dedication iv
Acknowledgements v
Table of Contents vi
List of Figures viii
1 Introduction 1
2 Related Works 4
   2.1 Adversarial Attack Algorithms on Deep Learning ......................... 4
      2.1.1 White-box Attack Algorithms ................................... 4
      2.1.2 Black-box Attack Algorithms .................................. 5
   2.2 Adversarial Training .................................................. 5
   2.3 Boosting Algorithms ............................................... 6
      2.3.1 Adaptive Boosting ........................................... 6
3 Problem Definition 7
   3.1 Preliminaries ..................................................... 7
   3.2 Problem Statement ................................................. 9
4 The Accumulating Adversarial Training Algorithm 11
   4.1 The Accumulating Adversarial Training Algorithm ....................... 11
      4.1.1 Time Complexity ............................................ 13
   4.2 Boosting Methods ................................................ 13
      4.2.1 The Accumulating Adversarial Training Algorithm with the Adaptive
             Boosting Method ........................................... 14
List of Figures

Figure 3.1 The above images are examples of (a) a clean image, (b) the corresponding adversarial perturbation, and (c) the corresponding adversarial image [20]. A well-trained neural network model can classify the clean image as an Eel confidently; however, by adding to the original image (a) with the adversarial perturbation (b), which is generated by the Fast Gradient Sign Method [9], the model may misclassify the image (c) as a Blowfish, even though according to human cognition, the image (c) looks exactly like an Eel.

Figure 4.1 The relationship among the four algorithms.

Figure 5.1 MCD values and Accuracy values at each training epoch on the MNIST dataset, when the training process of Algorithm 1 is being attacked by different adversarial attack algorithms. The caption for each subfigure declares the name of each adversarial attack algorithm.

Figure 5.2 MCD values and Accuracy values at each training epoch on the Fashion-MNIST dataset, when the training process of Algorithm 1 is being attacked by different adversarial attack algorithms. The caption for each subfigure declares the name of each adversarial attack algorithm.

Figure 5.3 MCD values and Accuracy values at each training epoch on the CIFAR-10 dataset, when the training process of Algorithm 1 is being attacked by different adversarial attack algorithms. The caption for each subfigure declares the name of each adversarial attack algorithm.
Figure 5.4  (a) the MCD values at each training epoch under attack of three adversarial attack algorithms during the training progress of Algorithm 1 on the MNIST dataset. Because the model has different defense capacities against different adversarial attack algorithms, we use coloured lines to show the MCD values of the different adversarial attack algorithms. (b)-(d) Accuracy values for each of the three adversarial attack algorithms at each training epoch on the MNIST dataset.

Figure 5.5  (a) the MCD values at each training epoch under attack of three adversarial attack algorithms during the training progress of Algorithm 1 on the Fashion-MNIST dataset. Because the model has different defense capacities against different adversarial attack algorithms, we use coloured lines to show the MCD values of the different adversarial attack algorithms. (b)-(d) Accuracy values for each of the three adversarial attack algorithms at each training epoch on the Fashion-MNIST dataset.

Figure 5.6  (a) the MCD values at each training epoch under attack of three adversarial attack algorithms during the training progress of Algorithm 1 on the CIFAR-10 dataset. Because the model has different defense capacities against different adversarial attack algorithms, we use coloured lines to show the MCD values of the different adversarial attack algorithms. (b)-(d) Accuracy values for each of the three adversarial attack algorithms at each training epoch on the CIFAR-10 dataset.

Figure 5.7  MCD values at each training epoch under the attack of one adversarial attack algorithm during the training procedure of Algorithm 2 on the MNIST dataset. In subfigures of the MCD values, we compare the result of Algorithm 1 and Algorithm 2. The “Adaboosting” line represents the MCD values of Algorithm 2, and the “noboosting” line represents the MCD values of Algorithm 1.

Figure 5.8  MCD values at each training epoch under the attack of one adversarial attack algorithm during the training procedure of Algorithm 2 on the Fashion-MNIST dataset. In subfigures of the MCD values, we compare the result of Algorithm 1 and Algorithm 2. The “Adaboosting” line represents the MCD values of Algorithm 2, and the “noboosting” line represents the MCD values of Algorithm 1.
Figure 5.9  MCD values at each training epoch under the attack of one adversarial attack algorithm during the training procedure of Algorithm 2 on the CIFAR-10 dataset. In subfigures of the MCD values, we compare the result of Algorithm 1 and Algorithm 2. The “AdaBoosting” line represents the MCD values of Algorithm 2, and the “noboosting” line represents the MCD values of Algorithm 1.  

Figure 5.10  MCD values at each training epoch of Algorithm 2 under the attack of three adversarial attack algorithms on the MNIST dataset. In the figures of the MCD values, we compare the result of Algorithm 1 and Algorithm 2. The “AdaBoosting” line represents the MCD values of Algorithm 2, and the “noboosting” line represents the MCD values of Algorithm 1. 

Figure 5.11  MCD values at each training epoch of Algorithm 2 under the attack of three adversarial attack algorithms on the Fashion-MNIST dataset. In the figures of the MCD values, we compare the result of Algorithm 1 and Algorithm 2. The “AdaBoosting” line represents the MCD values of Algorithm 2, and the “noboosting” line represents the MCD values of Algorithm 1. 

Figure 5.12  MCD values at each training epoch of Algorithm 2 under the attack of three adversarial attack algorithms on the CIFAR-10 dataset. In the figures of the MCD values, we compare the result of Algorithm 1 and Algorithm 2. The “AdaBoosting” line represents the MCD values of Algorithm 2, and the “noboosting” line represents the MCD values of Algorithm 1. 

Figure 5.13  These figures show the MCD values at each training epoch, when there is only one adversarial attack algorithm on the training procedure of Algorithm 3 on the MNIST dataset. For different adversarial attack algorithms, the name of the adversarial attack algorithm is shown in the caption under each subfigure. In each subfigure, the dashed coloured line represents the MCD values of Algorithm 1. In the legend, the number means the sample ratio. 

Figure 5.14  These figures show the MCD values at each training epoch, when there is only one adversarial attack algorithm on the training procedure of Algorithm 3 on the Fashion-MNIST dataset. For different adversarial attack algorithms, the name of the adversarial attack algorithm is shown in the caption under each subfigure. In each subfigure, the dashed coloured line represents the MCD values of Algorithm 1. In the legend, the number means the sample ratio.
Figure 5.15 These figures show the MCD values at each training epoch, when there is only one adversarial attack algorithm on the training procedure of Algorithm 3 on the CIFAR-10 dataset. For different adversarial attack algorithms, the name of the adversarial attack algorithm is shown in the caption under each subfigure. In each subfigure, the dashed coloured line represents the MCD values of Algorithm 1. In the legend, the number means the sample ratio.  

Figure 5.16 MCD value at each training epoch of each adversarial attack algorithm when we use all three adversarial attack algorithms in the training procedure of Algorithm 3 on the MNIST dataset. In each figure we compare the MCD value of Algorithm 1 (named “noboosting” in the figure) with the MCD value of Algorithm 3 with different sample ratio setting (For example, "0.05" in the legend means the sample ratio sets as 0.05).  

Figure 5.17 MCD value at each training epoch of each adversarial attack algorithm when we use all three adversarial attack algorithms in the training procedure of Algorithm 3 on the Fashion-MNIST dataset. In each figure we compare the MCD value of Algorithm 1 (named “noboosting” in the figure) with the MCD value of Algorithm 3 with different sample ratio setting (For example, "p = 0.05" in the legend means the sample ratio sets as 0.05).  

Figure 5.18 MCD value at each training epoch of each adversarial attack algorithm when we use all three adversarial attack algorithms in the training procedure of Algorithm 3 on the CIFAR-10 dataset. In each figure we compare the MCD value of Algorithm 1 (named “noboosting” in the figure) with the MCD value of Algorithm 3 with different sample ratio setting (For example, "p = 0.05" in the legend means the sample ratio sets as 0.05).  

Figure 5.19 These figures show the MCD values at each training epoch under the attack of different adversarial attack algorithms during the training procedure of Algorithm 4 on the MNIST dataset. For different adversarial attack algorithms, the name of the adversarial attack algorithm has been provided in the caption under each figure. In the figure, the legend "noboosting" refers to the MCD values of Algorithm 1.
Figure 5.20 These figures show the MCD values at each training epoch under the attack of different adversarial attack algorithms during the training procedure of Algorithm 4 on the Fashion-MNIST dataset. For different adversarial attack algorithms, the name of the adversarial attack algorithm has been provided in the caption under each figure. In the figure, the legend "noboosting" refers to the MCD values of Algorithm 1.

Figure 5.21 These figures show the MCD values at each training epoch under the attack of different adversarial attack algorithms during the training procedure of Algorithm 4 on the CIFAR-10 dataset. For different adversarial attack algorithms, the name of the adversarial attack algorithm has been provided in the caption under each figure. In the figure, the legend "noboosting" refers to the MCD values of Algorithm 1.

Figure 5.22 These subfigures show the MCD values at each training epoch of each adversarial attack algorithm when we use all three adversarial attack algorithms in the training procedure of Algorithm 4 on MNIST dataset. In each subfigure we compare the MCD values of Algorithm 1 (named as "noboosting" in the subfigures) with the MCD values of Algorithm 4, when we set the batch size as different values.

Figure 5.23 These subfigures show the MCD values at each training epoch of each adversarial attack algorithm when we use all three adversarial attack algorithms in the training procedure of Algorithm 4 on Fashion-MNIST dataset. In each subfigure we compare the MCD values of Algorithm 1 (named as "noboosting" in the subfigures) with the MCD values of Algorithm 4, when we set the batch size as different values.

Figure 5.24 These subfigures show the MCD values at each training epoch of each adversarial attack algorithm when we use all three adversarial attack algorithms in the training procedure of Algorithm 4 on CIFAR-10 dataset. In each subfigure we compare the MCD values of Algorithm 1 (named as "noboosting" in the subfigures) with the MCD values of Algorithm 4, when we set the batch size as different values.
Figure 5.25 These figures show the MCD values at each training epoch under the attack of different adversarial attack algorithms during the training procedure of Algorithm 2, Algorithm 3, and Algorithm 4 in the MNIST dataset. For different adversarial attack algorithms, the name of the adversarial attack algorithm has been provided in the caption under each figure. In the figure, the legend “Adaboosting” refers to the MCD values of Algorithm 2. The legends “p = 0.01”, “p = 0.05”, “p = 0.1”, “p = 0.3” and “p = 0.01” refer to the MCD values of Algorithm 3 when we set the probability as different values. The legend “batch size = 100” refers to the MCD values of Algorithm 4 when batch size sets as 100.

Figure 5.26 These figures show the MCD values at each training epoch under the attack of different adversarial attack algorithms during the training procedure of Algorithm 2, Algorithm 3, and Algorithm 4 in the Fashion-MNIST dataset. For different adversarial attack algorithms, the name of the adversarial attack algorithm has been provided in the caption under each figure. In the figure, the legend “Adaboosting” refers to the MCD values of Algorithm 2. The legends “p = 0.01”, “p = 0.05”, “p = 0.1”, “p = 0.3” and “p = 0.01” refer to the MCD values of Algorithm 3 when we set the probability to different values. The legend “batch size = 128” refers to the MCD values of Algorithm 4 when batch size sets as 128.

Figure 5.27 These figures show the MCD values at each training epoch under the attack of different adversarial attack algorithms during the training procedure of Algorithm 2, Algorithm 3, and Algorithm 4 in the CIFAR-10 dataset. For different adversarial attack algorithms, the name of the adversarial attack algorithm has been provided in the caption under each figure. In the figure, the legend “Adaboosting” refers to the MCD values of Algorithm 2. The legends “p = 0.01”, “p = 0.05”, “p = 0.1”, “p = 0.3” and “p = 0.01” refer to the MCD values of Algorithm 3 when we set the probability to different values. The legend “batch size = 128” refers to the MCD values of Algorithm 4 when batch size sets as 128.
Figure 5.28 These figures show the MCD values at each training epoch under the attack of all three adversarial attack algorithms during the training procedure of Algorithm 2, Algorithm 3, and Algorithm 4 in the MNIST dataset and the Fashion-MNIST dataset. For different adversarial attack algorithms, the name of the adversarial attack algorithm has been provided in the caption under each figure. In the figure, the legend “Adaboosting” refers to the MCD values of Algorithm 2. The legends “p = 0.01”, “p = 0.05”, “p = 0.1”, “p = 0.3” and “p = 0.01” refer to the MCD values of Algorithm 3 when we set the probability to different values. The legend “batch size = 128” refers to the MCD values of Algorithm 4 when batch size sets as 128.

Figure 5.29 These figures show the MCD values at each training epoch under the attack of all three adversarial attack algorithms during the training procedure of Algorithm 2, Algorithm 3, and Algorithm 4 in the Fashion-MNIST dataset. For different adversarial attack algorithms, the name of the adversarial attack algorithm has been provided in the caption under each figure. In the figure, the legend “Adaboosting” refers to the MCD values of Algorithm 2. The legends “p = 0.01”, “p = 0.05”, “p = 0.1”, “p = 0.3” and “p = 0.01” refer to the MCD values of Algorithm 3 when we set the probability to different values. The legend “batch size = 128” refers to the MCD values of Algorithm 4 when batch size sets as 128.

Figure 5.30 These figures show the MCD values at each training epoch under the attack of all three adversarial attack algorithms during the training procedure of Algorithm 2, Algorithm 3, and Algorithm 4 in the CIFAR-10 dataset. For different adversarial attack algorithms, the name of the adversarial attack algorithm has been provided in the caption under each figure. In the figure, the legend “Adaboosting” refers to the MCD values of Algorithm 2. The legends “p = 0.01”, “p = 0.05”, “p = 0.1”, “p = 0.3” and “p = 0.01” refer to the MCD values of Algorithm 3 when we set the probability to different values. The legend “batch size = 128” refers to the MCD values of Algorithm 4 when batch size sets as 128.
Chapter 1

Introduction

Deep neural networks and deep learning [13] have been widely applied in different fields such as computer vision [12], natural language processing [26] and board game strategy [24]. In vision recognition, recent studies propose some state-of-the-art convolutional neural networks, such as LeNet-5 [14], Alexnet [12] and ResNets [10]. These convolutional neural networks all have great performance on some image classification tasks.

However, through some analyses and experiments on deep neural networks, Szegedy et al. [27] first surprisingly realized that even a well trained deep neural network can be vulnerable to adversarial examples, which are original images added with some imperceptible perturbations. These adversarial examples are totally unambiguous to human vision, and some of them even look exactly like the original images from a human’s perspective. However, these adversarial examples can totally fool neural network classifiers, and some neural network classifiers even make misclassification with high confidence on some adversarial examples. Later on, Goodfellow et al. [9] proposed an efficient method, Gradient Sign Method (FGSM), to generate adversarial examples by calculating the gradient of the cost function. Su et al. [25] even proposed a method to generate the adversarial examples by changing only one pixel of the image. More adversarial attack algorithms are proposed to generate adversarial examples, such as DeepFool [18], Carlini and Wagner Attacks (C&W) [5] and Universal Adversarial Perturbations [17].

Facing these challenges, there are studies trying to develop defensing methods against various adversarial attack algorithms. There are three main directions to develop the defense methods, which listed as follows [2].

- Developing or improving the training method to gain a robust model or modifying input during testing [2].
- Changing the components on a neural network, such as adding more layers in the network structures or adjusting the loss functions [2].
- Using extra models to help the main neural network in classifying unseen data [2].
In this thesis, we mainly focus on improving the robustness of neural network by improving the training method.

As the first study on adversarial examples, Szegedy et al. [27] proposed to improve the robustness of neural networks using adversarial training, which uses both adversarial examples and the original dataset to train the neural network iteratively. This method can not only improve the robustness of the target neural network [27][9] but also regularize the neural network to reduce overfitting [23]. Inspired by this method, Miyato et al. [16] proposed a virtual adversarial training method for semi-supervised classification by labeling the unlabelled data with the predicted distributions and generating adversarial examples with local distributional smoothness. Moreover, Zheng et al. [30] created a stability training method to defend against small natural distortions on the input images by training the model with those images with small natural distortions. The adversarial training method is able to make the model more robust. However, Moosavi-Dezfooli et al. [17] presented that some effective adversarial examples can still be generated to defend the neural networks that have been trained by adversarial training. Moreover, since adversarial training needs to use the original dataset and adversarial examples in each training epoch, it costs more time than the normal neural network training procedure. To overcome these shortages, in this thesis, we propose an Accumulating Adversarial Training Algorithm and three boosting methods to improve the performance of the Accumulating Adversarial Training Algorithm.

In the Accumulating Adversarial Training Algorithm, we first take a well trained or well structured neural network model. We then train the model with the dataset that mixed with the original dataset, the adversarial dataset generated in the current iteration and the adversarial datasets generated in the previous iterations iteratively. Those adversarial datasets can be generated by one or several adversarial attack algorithms. By training a model with the original dataset and the accumulated adversarial examples iteratively, we can gain a trained model, which can defend against the adversarial examples that are generated by one or several adversarial attack algorithms. In our methods, the size of the training datasets is increasing in each training epoch, and the training procedure is time-consuming. We proposed three boosting methods, namely the adaptive boosting method, the group sampling boosting method and the stochastic mini batch boosting method. In the adaptive boosting method, we assign to those wrongly classified examples heavier weights. This method forces the model to focus on learning important features that helps on classification and improves the performance of the model. In the group sampling boosting method, instead of attacking the whole original dataset with different adversarial attack algorithms in each iteration to gain the adversarial examples, we only let the attacking algorithms to attack \((p \times 100)\%\) of the original dataset. In the stochastic mini batch method, given a batch size \(s\), when we meet \(s\) adversarial examples, we train the neural network model with the training dataset that consists of original images, \(s\) adversarial examples generated in the
current iteration, and the previously generated adversarial examples. Both group sampling boosting method and stochastic mini batch method reduce the training time by decreasing the size of training data size.

We summarize our contributions as follows.

- We propose an Accumulating Adversarial Training Algorithm that can make the model more robust to several adversarial attack algorithms.
- We design three boosting methods to improve the performance of the trained model.
- Instead of using Accuracy value, we propose a new measurement named the Mean of Correctness Difference (MCD) to evaluate the robustness of the neural network.
- We present experimental results on MNIST data sets showing that only a few adversarial examples can be generated for the trained neural network model using our training method.

The rest of the thesis is organized as follows. In Chapter 2, we review related works. In Chapter 3, we give some important and useful definitions and formulate the problem of improving the robustness of neural networks against adversarial attack algorithms. In Chapter 4, We present the Accumulating Adversarial Training Algorithm and the Accumulating Adversarial Training Algorithm combined with three boosting methods separately, namely the adaptive boosting method, the group sampling boosting method and the stochastic mini batch method. In Chapter 5, we report the experimental results of the four algorithms. Last, we conclude this thesis in Chapter 6.
Chapter 2

Related Works

Improving robustness of neural networks is an important topic these days. Our study is highly related to some existing topics including adversarial training, adversarial attack algorithms and boosting algorithms. In this chapter, we provide a detailed review of the related works and discuss the connections between these topics and our work.

2.1 Adversarial Attack Algorithms on Deep Learning

The adversarial attack problem was proposed by Szegedy in 2014 [27]. Adversarial attack algorithms are the algorithms to create adversarial examples, which are original images modified by adding some adversarial perturbations to mislead classification models. Depending on the understanding of the targeted models, existing adversarial attack algorithms can be divided into two categories: white-box attack algorithms and black-box attack algorithms.

2.1.1 White-box Attack Algorithms

A white-box attack algorithm knows the complete knowledge of a targeted model, including its parameter values, its architecture, the training method of the targeted model and, in some cases, its training dataset as well [2]. Based on the knowledge of the targeted model, a white-box attack algorithm can calculate the adversarial perturbation of a certain image in order to fool the targeted model. In order to show that our algorithm can make the model more robust to white-box attack algorithms, in the experiments of this thesis (Chapter 5), we use two white-box attack algorithms to generate the adversarial examples. We give a brief introduction of the two white-box adversarial attack algorithms here.

DeepFool [18]

The DeepFool method was proposed by Moosavi-Dezfooli et al. [18] to compute a minimal adversarial perturbation of an image. The DeepFool algorithm adds a small amount of vectors on a correctly classified image iteratively until the targeted model misclassifies
the changed image. A small amount of vectors can be calculated by getting the orthogonal distance between the correctly classified image and the closest affine hyper-plane. By accumulating the calculated vectors in each iteration, we can get a minimal adversarial perturbation of the image.

**Fast Gradient Sign Method (FGSM) [9]**

Fast Gradient Sign Method calculates the gradient of the cost function $J(\theta, x, y)$ based on a model parameter $\theta$ in order to get an optimal max-norm constrained perturbation. The way to calculate the perturbation can be formed as:

$$\epsilon \text{sign}(\nabla_x J(\theta, x, y)) [9]$$

where $x$ is a flattened vector of an image, $y$ is the label of $x$, $\text{sign}(.)$ is the sign function, and $\epsilon$ is a small scalar value to control the norm of the perturbation. This method gives a linear view on adversarial examples, and the authors assume that neural networks are too linear to handle linear adversarial perturbations [9].

### 2.1.2 Black-box Attack Algorithms

Different from white-box attack algorithms, a black-box attack algorithm tries to generate adversarial examples without or with limited knowledge of a targeted model [2]. Black-box attack algorithms know nothing about the parameters of the model. In order to show that our algorithm can make the model more robust to black-box attack algorithms as well, in the experiments of this thesis (Chapter 5), we use one black-box attack algorithm to generate the adversarial examples. We give a brief introduction to this black-box adversarial attack algorithm here.

**Salt-and-Pepper Noise Attack (SPNA) [22]**

Salt and Peper Noise is sparsely occurring black and white pixels in an image [29], which is also known as impulse noise. The Salt-and-Pepper Noise Attack algorithm increases the amount of salt and pepper noise by a linear search method iteratively until the image is misclassified [22].

### 2.2 Adversarial Training

Adversarial training was proposed by Szegedy [27], which is a method to feed adversarial examples in the training dataset to make a target machine learning model more robust. Adversarial training is presented to give a more robust model [27][9][15]. However, Szegedy et al. [27] only fed the adversarial examples generated on the current training epoch, and
simply omitted the adversarial examples generated on the previous training epochs. In our opinion, this simple adversarial training may make the target machine learning model lose some important features during the training procedure, and make the learning process inefficient. Thus, we propose an Accumulating Adversarial Training Algorithm in this thesis.

2.3 Boosting Algorithms

Boosting algorithms are general methods to improve the accuracy of any given learning algorithms [7]. They are able to train those weak classifiers, which are classifiers that slightly better than a random guess, into strong classifiers [31]. Moreover, boosting algorithms are proved that can primarily reduce bias and variance in supervised learning [4]. The general boosting algorithm procedure is quite simple. It trains a set of classifiers sequentially and combine them by assigning different weights based on their contributions to construct a strong classifier [31]. By reweighting this set of classifiers for several rounds and combining them linearly, we eventually get a satisfactory classifier. In this thesis, we try to combine some boosting algorithms with an accumulating adversarial training algorithm. In the following subsection, we introduce a boosting algorithm applied in our algorithm part.

2.3.1 Adaptive Boosting

Among those boosting algorithms, the most influential one is Adaptive Boosting [31]. In the machine learning area, Adaptive Boosting can be applied as follows. Given a training dataset \( \{ (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n) \} \), where \( x_i \) is in an instance space \( X \), and \( y_i \) is the label of \( x_i \) and belongs to a label set \( Y \). We denote a neural network classifier by \( f_W(x) = \sum_{j=1}^{\infty} w_j \phi_j(x) \), where \( \phi_j(x) \) is an infinite set of feature functions, and \( w_j \) is the weight of the \( j^{th} \) feature function. At the beginning of the boosting procedure, we assign the same weight to each training example in the dataset. In each round, the trained classifier may get some weak feature functions, which play an insignificant role in classification. In the adaptive boosting method, we reweight the training examples to give those incorrectly classified examples higher weights, and the adaptive boosting algorithm does coordinate descent on \( w_j \) by putting more weights on those important features. Eventually, this method leads to a strong classifier by reweighting the weight \( w_j \) of each feature function. The adaptive boosting method forces the targeted model to learn easily misclassified examples in the training dataset and focuses on learning important features of classification [7]. In our thesis, we combine the Adaptive Boosting method with an accumulating adversarial training algorithm in order to improve the performance of the trained model.
Chapter 3

Problem Definition

In this chapter, we formulate the problem of improving the robustness of neural networks. Before we define the problem, we first introduce the essential preliminaries of our problem.

3.1 Preliminaries

Definition 3.1.1 (Neural network). A neural network $f_w$ is a model for statistical pattern recognition [3]. A neural network model can be represented as a mapping $y = f_w(x)$, which learns the value of the parameters (or weights) $w$ to map an input image $x$ to a correct category $y$ [8].

From the probability distribution perspective, we interpret $p(y|x; w)$ as the conditional probability distribution of $y$ with a given image $x$ on the neural network $f_w$. Among this distribution, if a category $y$ gains the highest conditional probability, the model classifies $x$ as $y$.

Thus, given $\hat{w}$ as the estimated parameters of this neural network, the predicted output of $x$ is denoted by $\hat{y}$, where $\hat{y} = f_{\hat{w}}(x)$. With the given $\hat{w}$ and $x$, we estimate $\hat{y}$ as the class that maximizes $p(y|x; \hat{w})$.

Thus, we regard

$$\hat{y} = f_{\hat{w}}(x) = \arg\max_y (p(y|x; \hat{w}))$$

where $\hat{w}$ is the estimated parameters of the neural network.

For example, if we have a dataset which contains images of cats and dogs. Given an image $x$ and neural network $f_w$, the predicted conditional probability is $p(\text{cat}|x; w) = 0.7$ and $p(\text{dog}|x; w) = 0.3$. This means that the neural network predicts the image $x$ as a cat.

Definition 3.1.2 (Original / clean image). An original image (or clean image) is an image from the original image dataset that has never been modified by any method. It is denoted by $I_i \in I$, where $I_i$ is an original image, and $I$ is the original image dataset.
An example of an original / clean image is given in Figure 3.1 (a).

Definition 3.1.3 (Adversarial perturbation [2]). An adversarial perturbation $r_i$ is an imperceptible change on an original image $I_i$ that misleads the neural network algorithm $f_w$ to make an incorrect prediction, that is,

$$f_w(I_i) \neq f_w(I_i + r_i)$$

Equivalently,

$$\arg\max_{y_i}(p(y_i|I_i; w)) \neq \arg\max_{y_i}(p(y_i|I_i + r_i; w))$$

where $w$ is the parameters of the neural network.

Specifically, in order to make the amount of an adversarial perturbation measurable, we assume that $||r_i||_2 \in (0, \epsilon)$, which means that perturbing the image beyond $(0, \epsilon)$ will lead to a perceptible change.

An example of an adversarial perturbation is provided in Figure 3.1 (b).

Definition 3.1.4 (Adversarial example / image [18]). An adversarial example/image, $I'_i \in I'$, can be represented by

$$I'_i = I_i + r_i$$

where $I_i \in I$ is an original image, and $r_i$ is an adversarial perturbation of image $I_i$. $I_i + r_i$ means we modify the original image by adding the adversarial perturbation to generate the adversarial example/image, which leads the classification function to misclassify the image.

An example of an adversarial example / image is given in Figure 3.1 (c).

Definition 3.1.5 (Adversarial attack algorithm [18]). An adversarial attack algorithm is an algorithm $A(I, f_w)$ to calculate the adversarial perturbation $r$ with a given neural network $f_w$ and a clean image $I$:

$$\text{objective function } A(I, f_w) := \min ||r||_2$$

subject to

$$f_w(I_i) \neq f_w(I_i + r_i)$$

Definition 3.1.6 (Adversarial training [2]). Adversarial training involves a clean/original dataset $\{I_1, I_2, ..., I_k\}$ and a set of adversarial examples/images $\{I'_1, I'_2, ..., I'_k\}$ to train the neural network model.

Definition 3.1.7 (Decision regions and decision boundary [3]). In classification tasks, we divide the input space into regions, $R_1, R_2, ..., R_k$, called decision regions, one for each class,
such that all points in $R_k$ are assigned to class $C_k$. The boundaries between decision regions are called decision boundaries.

**Definition 3.1.8** (Robustness of the neural network). We use $I$ to denote an original dataset, and $I_i$ is one of the original images in the original dataset, $I_i \in I$. We use $I'$ to denote the adversarial dataset, and $I'_i$ is one of the adversarial images in the adversarial dataset, $I'_i \in I'$. Specifically, $I'_i$ is the adversarial image of $I_i$. Moreover, we use $y_i$ to denote the possible label of $I'_i$, and $y_{ti}$ is the true label of $I_i$ and $I'_i$. We evaluate the robustness of the neural network by the Mean of Correctness Difference (MCD), which is

$$MCD(I', Y; f_w) = \sum_{I'_i \in I'} \left| \max(p(y_i|I'_i; w)) - p(y_{ti}|I'_i; w) \right|.$$  

MCD measures the mean of the difference between the predicted probability of the adversarial image and the predicted probability of the correct labels in the adversarial dataset $I'$. The smaller the MCD, the more robust the model.

### 3.2 Problem Statement

Given a neural network $f_w$, an original dataset $I$, and a corresponding label set of original dataset $Y$, our task is to improve the robustness of the neural network $f_w$ to correctly classify the adversarial examples generated by one or several adversarial attack algorithms.
that is

\[
\begin{align*}
\text{minimize}_{w} \quad & MCD(I', Y; f_w) \\
\text{subject to} \quad & \text{one or several given adversarial attack algorithms } A \\
& 0 < \|r_{im}\|_2 < \epsilon
\end{align*}
\]

where \(r_{im}\) is the adversarial perturbation of an original image \(I_i\) generated by \(m^{th}\) adversarial attack algorithm, and \((0, \epsilon)\) is the range of an adversarial perturbation. We assume that perturbing the image beyond this range will lead to a perceptible change. \(I'\) is the adversarial image dataset, which is created by the adversarial attack algorithm(s) \(A\) and the original dataset \(I\).
Chapter 4

The Accumulating Adversarial Training Algorithm

In this chapter, we present the basic algorithm, the Accumulating Adversarial Training Algorithm, at first. Furthermore, to improve the efficiency and scalability of the Accumulating Adversarial Training Algorithm, we propose three different boosting methods, namely the adaptive boosting method, the group sampling boosting method and the stochastic mini batch boosting method.

4.1 The Accumulating Adversarial Training Algorithm

The core idea of adversary training is to train a neural network with both the original dataset and the adversarial dataset in every training iteration [9]. Goodfellow et al. [9] and Szegedy et al. [27] used adversarial training to regularize neural network models. Goodfellow et al. [9] also found that a model trained with the following objective function was an effective regularizer:

$$\bar{J}(w, I, y^t) = \alpha J(w, I, y^t) + (1 - \alpha) J(w, I', y')$$

where $J(w, I, y^t)$ is the loss function on data $I$ with the true label $y^t$. $I'$ is the adversarial example that was generated by an adversarial attack algorithm $A(I, f_w)$. Goodfellow et al. [9] used this approach to continually update the supply of adversarial examples to make the current version of the model resist the new adversarial examples. However, this objective function only focuses on the adversarial examples generated by the current version of the model and simply ignores the adversarial examples generated by the previous versions of models during the training procedure.

According to the decision theory [3], the existing shortages on decision boundary lead a model to misclassify some input objects. During the training procedure, input data plays
an important role in training the model to minimize the misclassification rate. The model learns features from the input data in order to classify the data correctly. If the misclassification rate is small enough, we suppose that the model has the ability to classify the majority data into the correct decision region, which means that the model is approaching the correct decision boundary [3]. Thus, we assume that each single piece of data can make a limited contribution to adjusting the decision boundary during the neural network model training procedure [27]. As a result, if we simply ignore the adversarial images generated in the previous iterations, the newly adjusted decision boundary loses some important contributions made by the ignored adversarial images. Therefore, we propose a new objective function, as follows:

\[
\tilde{J}(w_r, I, Y^t) = J(w_r, I, Y^t) + \sum_r J(w_r, I'_r, Y^t)
\]

where \( I \) is a given training dataset with the true label set \( Y^t \), and \( I'_r \) contains the adversarial examples that are generated by an adversarial attack algorithm \( A(I, f_{w_r}) \) at the \( r^{th} \) training epoch. Moreover, \( f_{w_r} \) is the neural network model with weight \( w_r \) at the \( r^{th} \) round. The intuition behind the new objective function is to train the model with the original dataset and the accumulated adversarial training dataset, which consists of adversarial examples that are generated at the previous and the current training epochs, to make the model defend against the adversarial examples that are generated by the given adversarial attack algorithms.

To implement this method, we develop Algorithm 1.

---

**Algorithm 1: The Accumulating Adversarial Training Algorithm**

**input**: original image dataset \( I = \{I_1, I_2, ..., I_k\} \), attack algorithms \( A_1, A_2, ..., A_m \)

**output**: trained neural network \( f_{w_r} \)

1. Randomly initialize neural network \( f_{w_0} \)
2. Training dataset \( D = \{I\} \)
3. repeat
4. do the training procedure of network \( f_w \) using dataset \( D \)
5. create \( I' = \{I'_1, I'_2, ..., I'_i, ..., I'_{km}\} \), where \( I'_i \) is the \( i^{th} \) image in \( I \) that attacked by \( j^{th} \) attack algorithm
6. \( D = D \cup I' \)
7. until \( MCD(I', Y; f_{w_r}) \) converges;
8. return \( f_{w_r} \)

---

In the Accumulating Adversarial Training Algorithm, we first initialize the neural network \( f_w \) by randomly setting the parameters \( w \). We also need an original image dataset \( I \) and a set of the adversarial attack algorithms \( A \). In each round of the Feed-Forward
and Back-Propagation training procedure, we train this neural network with the original dataset and the adversarial datasets that are generated in the current and the previous iterations. We then test the MCD value of the current model. If the MCD value converges, we terminate our training procedure. Through this training procedure, we keep the decision boundary that is calculated from the previous iterations; moreover, we use the newly generated adversarial images to fix the shortages in the current decision boundary.

However, this algorithm is time consuming. In each iteration, Algorithm 1 requires each adversarial attack algorithm to attack each original image in the original dataset. For example, suppose there are 1,000 images in the original training dataset, and we have 3 adversarial attack algorithms. At each training epoch, we can at most generate 3,000 adversarial examples. If each adversarial attack algorithm takes at most 2 seconds to complete an adversarial attack, it takes 100 minutes to generate all of the adversarial examples each round. If there are many adversarial attack algorithms and the size of the dataset is large, the whole training procedure is extremely time consuming.

4.1.1 Time Complexity

We assume the size of the training dataset is $k$, the longest time to make an adversarial image is $t$, the number of adversarial attack algorithms is $m$ and the total number of training epochs is $r$. Based on the Accumulating Adversarial Training Algorithm, at each training epoch, the time cost of attacking one original image is $O(tm)$. Thus, among $r$ training epochs, if the size of the training dataset is $k$, the time complexity for making adversarial images is $O(rtmk)$. At the very first training epoch, we use $k$ images to train the neural network model, and we add at most $mk$ newly created adversarial images to train the neural network in each new training epoch. If the size of the training dataset at the first training epoch is $k$, the size of the training dataset at the second training epoch is $k + mk$ and the size of the training dataset at the third training epoch is $k + 2mk$, and so on. We assume that the training time of a neural network model $f_w$ using $k$ images is $T$, and the training time is constant for every $k$ images. Thus, the time complexity of training the neural network among $r$ training epochs is $O(T + (T + mT) + (T + 2mT) + \ldots + (T + (r-1)mT)) \leq O(mTr^2)$. Thus, the time complexity of Algorithm 1 is $O(rtmk + mTr^2)$.

4.2 Boosting Methods

In order to reduce the training time of the Accumulating Adversarial Training Algorithm (Algorithm 1) and make Algorithm 1 scalable, we propose three boosting methods, namely the adaptive boosting method, the group sampling boosting method and the stochastic mini batch boosting method.
4.2.1 The Accumulating Adversarial Training Algorithm with the Adaptive Boosting Method

Inspired by the regular adaptive boosting method and based on the optimization-based perspective, Duchi [6] interpreted the adaptive boosting method as a coordinate descent method in a finite dimensional space for a neural network learning procedure. We assume that a classifier function classifies an image using different features of the image, and each feature takes a different weight in the classification. Thus, we can regard a neural network classifier as 

\[ f_w(x) = \sum_{j=1}^{\infty} w_j \phi_j(x), \]

where \( \phi_j(x) \) is the \( j \)th feature function, which maps \( x \) to a vector of feature value, and \( w_j \) is the weight of the \( j \)th feature function. At the beginning of the boosting procedure, we assign the same weight to each training example. In each round, the trained classifier may get some weak feature functions, which play an insignificant role in the classification. By taking the reweighting sample idea in the adaptive boosting method, if we put more weights on those wrongly classified examples, our adaptive boosting method does coordinate descent on \( w_j \) by putting more weights on those significant feature functions. Eventually, this method leads to a strong classifier by reweighting the weight of each feature function [6].

From the decision boundary perspective, a sample data with a higher error ratio may contribute more to learning and adjusting the decision boundary into the appropriate position. Thus, we reweight the data by multiplying the corresponding relative error ratio and use this newly weighted dataset as the input dataset. We implement the adaptive boosting method by changing our objective function as follows:

\[
\tilde{J}(w_r, I, Y^t) = J(w_r, I, Y^t) + \sum_r J(w_r, I'_r, Y^t)
\]

where 

\[
I'_r = \left\{ \frac{e_{11}}{\sum_1^m \sum_1^k e_{ij}} I'_1 1, \frac{e_{12}}{\sum_1^m \sum_1^k e_{ij}} I'_2 2, ..., \frac{e_{ij}}{\sum_1^m \sum_1^k e_{ij}} I'_i j, ..., \frac{e_{km}}{\sum_1^m \sum_1^k e_{ij}} I'_k m \right\}
\]

and 

\[
e_{ij} = |\max(p(y_i|I'_{ij}, w)) - p(y'_i|I'_{ij}, w)|
\]

In the above equation, we assume that we have \( k \) images in the original dataset, and \( m \) adversarial attacking algorithm(s). We use \( I'_r \) to denote an adversarial dataset that is generated by a set of adversarial attack algorithms \( A \) at the \( r \)th round of our training iterations. In the adversarial dataset \( I'_r \), \( \frac{e_{ij}}{\sum_1^m \sum_1^k e_{ij}} \) is the relative error ratio of \( I'_{ij} \), and \( I'_{ij} \) is the \( i \)th image that is attacked by the \( j \)th adversarial attack algorithm. \( \frac{e_{ij}}{\sum_1^m \sum_1^k e_{ij}} I'_{ij} \) represents that we reweight \( I'_{ij} \) by multiplying the corresponding relative error ratio \( \frac{e_{ij}}{\sum_1^m \sum_1^k e_{ij}} \). Using this method, an image with a higher relative error ratio takes more weight among the whole dataset, and it forces the decision boundary to be adjusted to correctly classify those misclassified images. We embed the adaptive boosting method into Algorithm 1 to boost
the performance of Algorithm 1, and we propose the Accumulating Adversarial Training Algorithm with the Adaptive Boosting Method (Algorithm 2).

In Algorithm 2, we first initialize the neural network $f_w$ by randomly setting all the weights $w$ in this neural network. With a given original dataset $I$ and the adversarial attack algorithms $A$, we start to train the neural network iteratively. In each training iteration, we train this neural network with the dataset that consists of the original dataset and the weighted adversarial datasets generated in the current iteration and the previous iterations. During the training procedure, we force the neural network to take more contributions from the misclassified images to adjust the decision boundary quickly.

**Algorithm 2: The Accumulating Adversarial Training Algorithm with the Adaptive Boosting Method**

**input**: original image dataset $I = \{I_1, I_2, ..., I_k\}$, attack algorithms $A_1, A_2, ..., A_m$

**output**: trained neural network $f_w$

1. Randomly initialize neural network $f_w$
2. Training dataset $D = \{I\}$
3. repeat
4. do the training procedure of network $f_w$ using dataset $D$
5. create $I' = \{I'_1, I'_2, ..., I'_i, ..., I'_km\}$, where $I'_i$ is the $i^{th}$ image in $I$ that attacked by the $j^{th}$ attack algorithm
6. for $i = 1$ to $k$
7. for $j = 1$ to $m$
8. $e_{ij} = |max(p(y_i|I'_i, W)) - p(y_i'|I'_i, W)|$
9. end
10. end
11. recreate $I' = \{\sum_{i}^m e_{i1} I'_1, \sum_{i}^m e_{i2} I'_2, ..., \sum_{i}^m e_{ij} I'_i, ..., \sum_{i}^m e_{km} I'_km\}$, where $I'_i$ is the $i^{th}$ image that attacked by the $j^{th}$ attack algorithm
12. $D = D \cup I'$
13. until $MCD(I', Y, f_w)$ converges;
14. return $f_w$

4.2.2 The Accumulating Adversarial Training Algorithm with the Group Sampling Boosting Method

We assume that a current training dataset is a simple random sample dataset of the true population. Based on the Simple Random Sampling method [28], the current training dataset follows the same distribution as the true population. Thus, a simple random sample dataset drawn from the current training dataset follows the same distribution as the population.
Furthermore, a simple random sample dataset has the same characteristics as the population [28]. Therefore, a model trained with simple random sample datasets of the current dataset can learn characteristics of the true population and fit the true population. Thus, we come up with the Group Sampling Boosting Method to deal with the large-scale optimization problem in the machine learning area.

In the Group Sampling Boosting Method, we draw a simple random sample dataset with size $p \times k$, where $p$ is the sample ratio in the range $(0, 1)$, and $k$ is the sample data size. Instead of attacking the whole original dataset with different adversarial attack algorithms, we only let the adversarial attack algorithms attack the $(p \times 100)\%$ original dataset. Training with simple random sample datasets, a model meets the data with the same distribution as the true population and learns the same characteristics as the true population. Thus, the model can learn the same necessary features as it can learn from the original dataset. We embed the Group Sampling Boosting method in Algorithm 1 and come up with the Accumulating Adversarial Training Algorithm with the Group Sampling Boosting method (Algorithm 3).

In Algorithm 3, given a neural network $f_w$ with randomly initialization of weights $w$, an original training dataset $I$ with size $k$, a set of adversarial attack algorithms $A$ and a uniform sampling rate $p$, we start to train the neural network iteratively. In each training iteration, we train this neural network with the dataset that includes the original training dataset and the adversarial random sample datasets generated in the current and the previous iterations. An adversarial random sample dataset is generated by attacking $p \times k$ simple random samples with each adversarial attack algorithm in each iteration. We keep training the neural network until the MCD value converges.

In Algorithm 3, we train the model with the original dataset, the newly generated and the previously generated adversarial sample datasets at each round. With the Group Sampling Boosting method, we can observe that the larger the sample ratio is, the larger a simple random sample dataset we can draw. As a result, the larger a simple random sample dataset is, the more necessary features a model can learn. Moreover, as we have mentioned before, misclassification of a model is due to the shortages in the decision boundary. Therefore, the higher sample ratio leads the larger sample dataset at different iterations. The larger sample dataset contains more images, which can contribute more different features to train the model and lower the misclassification ratio, which indicates that some shortages on the decision boundary are fixed.

To summarize, in principle, a high sample ratio may speed up the learning procedure. In the Accumulating Adversarial Training Algorithm with the Group Sampling Boosting Method (Algorithm 3), the small sample ratio scales down the computational time because we only need to attack $(p \times 100)\%$ percent of the original dataset at each iteration. However, there is a great chance that a small sample ratio leads to more rounds of training iterations,
because a small sample dataset may contain limited characteristics of the whole dataset that makes the model learn fewer characteristics at each iteration.

**Algorithm 3:** The Accumulating Adversarial Training Algorithm with the Group Sampling Boosting Method

**input** : original image dataset $I = \{I_1, I_2, \ldots, I_k\}$, attack algorithms $A_1, A_2, \ldots, A_m$

**output:** trained neural network $f_{w_r}$

1. Randomly initialize neural network $f_{w_0}$
2. Training dataset $D = \{I\}$
3. repeat
4. do the training procedure of network $f_W$ using dataset $D$
5. select an uniform sample $I_s$ from $I$ with sample size $k' = p \times k$
6. create $I'_s = \{I'_{11}, I'_{21}, \ldots, I'_{ij}, \ldots, I'_{k'm}\}$, where $I'_{ij}$ is the $i^{th}$ image in $I$ that attacked by the $j^{th}$ attack algorithm
7. $D = D \cup I'_s$
8. until $\text{MCD}(I', Y; f_{w_r})$ converges;
9. return $f_{w_r}$

### 4.2.3 Time Complexity

We assume the size of the training dataset is $k$, the sample ratio is $p$, the longest time to make an adversarial image is $t$, the number of adversarial attack algorithms is $m$ and the total number of training epochs is $r$. Based on the Accumulating Adversarial Training Algorithm with the Group Sampling Boosting Method, at each training epoch, the time cost of attacking one original image is $O(tm)$. Thus, among $r$ training epochs, if the size of the training dataset is $k$, the time complexity for making $p \times k$ adversarial images is $O(rtmpk)$. At the very first training epoch, we use $k$ images to train the neural network model, and we add at most $pmk$ newly created adversarial images to train the neural network in each new training epoch. If the size of the training dataset at the first training epoch is $k$, the size of the training dataset at the second training epoch is $k + mpk$ and the size of the training dataset at the third training epoch is $k + 2mpk$, and so on. We assume that the training time of a neural network model $f_w$ using $k$ images is $T$, and the training time is $p \times T$ for every $p \times k$ images. Thus, the time complexity of training the neural network among $r$ training epochs is $O(T + (T + mpT) + (CT + 2mpT) + \ldots + (CT + (r - 1)mpT)) \leq O(mpTr^2)$. Thus, the time complexity of Algorithm 3 is $O(rtmpk + mpTr^2)$. 

17
4.2.4 The Accumulating Adversarial Training Algorithm with the Stochastic Mini Batch Boosting Method

Algorithm 4: The Accumulating Adversarial Training Algorithm with the Stochastic Mini Batch Boosting Method

<table>
<thead>
<tr>
<th>input</th>
<th>original image dataset $I = {I_1, I_2, ..., I_k}$, attack algorithms $A_1, A_2, ..., A_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>output</td>
<td>trained neural network $f_{w_r}$</td>
</tr>
</tbody>
</table>

1. Randomly initialize neural network $f_{w_0}$
2. Training dataset $D = \{I\}$
3. repeat
4. do the training procedure of network $f_w$ using dataset $D$
5. for $j = 1$ to $m$ do
6. $I'_j = \{\}$
7. for $i = 1$ to $k$ do
8. if $|I'| < s$ then
9. $I' = I' \cup \{I'_{ij}\}$ where $I'_{ij}$ is the $i^{th}$ image in $I$ that attacked by the $j^{th}$ attack algorithm
10. end
11. end
12. $D = D \cup I'_j$
13. end
4. until $MCD(I', Y; f_{w_r})$ converges;
5. return $f_{w_r}$

Mini Batch Gradient Descent (MBGD) method [19] is a method to train deep neural network models efficiently. We first give a simple example to briefly explain this method. In our example, we state a simple linear model as

$$h_w(x) = \sum_{j}^{n} w_j x_j$$

where $w_j$ is the parameter of $x_j$, and $x$ is an $n$-dimensional vector, where each dimension represents a feature of the data $x$. We also use $y$ to denote the label of $x$, and $h_w(x)$ is
to estimate $y$ with a given $x$. To derive and estimate the parameters of $h_w(x)$, we use the MBGD method. We demonstrate the MBGD method as follows:

**Algorithm 5**: Mini Batch Gradient Descent (MBGD) method on simple linear model

\[19\]

**input** : Randomly initialize $w$ in neural network $h_w$, original image dataset $X = \{x^1, x^2, ..., x^k\}$, mini batch size $s$

1 repeat
   2 for each mini batch do
   3 $w_j := w_j + \alpha \sum_{i=0}^{s}(y^{(i)} - h_w(x^{(i)}))x_j^{(i)}$ (for every $j$)
   4 end
5 until convergence;

where $y^{(i)}$ is the label of $x^i$, and $x_j^{(i)}$ is the $j^{th}$ feature dimension of $x^i$. Moreover, $s$ is the size of a batch of data and $\alpha$ is the learning rate [19]. In the Mini Batch Gradient Descent method (Algorithm 5), $s$ is small. Once we calculate the total loss of one batch of data, which is denoted as $\sum_{i=0}^{s}(y^{(i)} - h_w(x^{(i)}))x_j^{(i)}$, the MBGD method (Algorithm 5) updates each parameter $w_j$ to reduce the total loss of one batch of data using the gradient descent method.

The Gradient Descent method updates the model parameters once we obtain the total loss of the whole dataset, and the Gradient Descent method can be represented by setting $s$ equal to the size of $X$ in Algorithm 5. Compared with the Gradient Descent method, the Mini Batch method may take less computational time. We assume the time for calculating the loss of one data is $t$, The Gradient Descent method takes $kt$ to obtain the total loss of the whole dataset, where $k$ is the size of $X$. However, for each iteration, the Mini Batch method takes $st$ to obtain the loss of a mini batch of data, where $s << k$. The Mini Batch method may be finished within a few iterations, which means the Mini Batch method may take less computational time than the Gradient Descent method.

The Stochastic Gradient Descent method updates the parameters of the model whenever we get the loss of a single data, and the Stochastic Gradient Descent method can be represented by setting $s$ equal to 1 in Algorithm 5. Compared with the Stochastic Gradient Descent method, the Mini Batch method takes fewer iterations to optimize the function. Since the Stochastic Gradient Descent method updates the parameters whenever the method gets the loss of a single data, the Stochastic Gradient Descent method takes at most $k$ iterations, where $k$ is the size of $X$. However, the Mini Batch method updates the parameters whenever the method gets the loss of one mini batch of data, where the size of one mini batch is $s$. The Mini Batch method takes at most $k/s$ iterations, which is less than the Stochastic Gradient Descent method.
Because of the advantage of the Mini Batch Gradient Descent method, we propose the Accumulating Adversarial Training Algorithm with the Stochastic Mini Batch Boosting method (Algorithm 4). In Algorithm 4, at the very beginning, we should provide an original dataset $I$, which contains $k$ original images and a set of adversarial attack algorithms $A$. Before we start our training procedure, we randomly initialize $w$ in $f_{w_0}$. The reason why we name it “stochastic” is that we randomly pick a batch of adversarial examples at each training epoch. We randomly shuffle the training dataset at the beginning of each training epoch to guarantee the stochastic procedure. In each training iteration, we train the neural network model with the dataset including the original dataset and the first $s$ stochastic adversarial examples we meet at each training epoch. We keep training the neural network until the MCD value converges.

Compared with Algorithm 1, Algorithm 4 updates the neural network model faster. Algorithm 1 updates the neural network once we attack the whole dataset with each adversarial attack algorithm. For large-scale data, it is very time consuming to attack the whole training dataset at each training epoch. However, Algorithm 4 updates the neural network model once it finds $s$ adversarial examples at each training epoch, which takes less training time than Algorithm 1. For different datasets, the values of the batch sizes are different. We will discuss the values of batch sizes in Chapter 5. In general, the larger the batch size, the smaller the MCD values converge. Furthermore, for the large-scale data, Algorithm 1 takes many I/O times to access the whole training dataset at each iteration, while Algorithm 4 takes fewer I/O times because Algorithm 4 attacks a mini batch of the large-scale dataset at each iteration. For the large-scale data, Algorithm 1 needs to attack the entire data, which need to have many interactions with the disks to fetch the data. However, Algorithm 4 only needs to attack $s$ data, and we usually set $s << k$. As a result, Algorithm 4 needs fewer iterations with the disks and takes fewer I/O times than Algorithm 1.

From the decision boundary perspective, at each training epoch, Algorithm 4 leads the model to refine parts of the decision boundary using the stochastic mini batch of adversarial examples and learn some features from the stochastic mini batch of adversarial examples. By accumulating each mini batch of adversarial examples, Algorithm 4 gradually fixes the shortages in the decision boundary.

### 4.3 Relationship Among the Four Algorithms

The main purpose of the four algorithms is to improve the robustness of a neural network model. Among the four algorithms, Algorithm 1 is the main algorithm, which uses the accumulating adversarial training method to improve the robustness of a neural network model. However, Algorithm 1 is slow. In order to improve the performance of Algorithm 1, we propose three more algorithms.
In The Accumulating Adversarial Training Algorithm with the Adaptive Boosting Method (Algorithm 2), we try to improve the performance of Algorithm 1 by reweighting the adversarial datasets. In the Accumulating Adversarial Training Algorithm with the Group Sampling Boosting Method (Algorithm 3), we try to overcome the weakness of Algorithm 1 by sampling a portion of the original dataset to generate the adversarial examples. In the Accumulating Adversarial Training Algorithm with the Stochastic Mini Batch Boosting Method (Algorithm 4), we try to boost the performance of Algorithm 1 by updating the target model whenever we collect the first $s$ stochastic adversarial examples we meet with each adversarial attack algorithm.
Chapter 5

Experiments

In this chapter, we report, discuss and analyze the experiment results of Accumulating Adversarial Training Algorithm (Algorithm 1), the Accumulating Adversarial Training Algorithm with Adaptive Boosting Method (Algorithm 2), the Accumulating Adversarial Training Algorithm with Group Sampling Boosting Method (Algorithm 3), and the Accumulating Adversarial Training Algorithm with the Stochastic Mini Batch Boosting Method (Algorithm 4).

5.1 Environment and Datasets

The four proposed algorithms are all implemented in Python 3.7. All experiments are run on PC computers with Intel Core i7-3770 (3.40 GHz) CPUs and 16GB main memory running Microsoft Windows 7.

In this thesis, we conduct the experiments on the MNIST datasets [14] of handwritten digits, the Fashion-MNIST datasets [21] of fashion images and the CIFAR-10 datasets [11] of colour images in 10 classes. Both the MNIST datasets and the Fashion-MNIST datasets consist of black and white images, and the size of each image is $28 \times 28$ pixels. In MNIST datasets, each image records the handwriting of one number between 0 and 9. In Fashion-MNIST datasets, each image records one of the fashion items among T-shirt/top, trouser/pants, pullover shirt, dress, coat, sandal, shirt, sneaker, bag and ankle boot. The CIFAR-10 datasets consists of $32 \times 32$ pixels images in 10 classes, each image records one of the items among airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. In all of the three datasets, there are training dataset and test dataset. For MNIST and Fashion-MNIST datasets, the training dataset contains 60,000 images, and the training dataset of CIFAR-10 contains 50,000 images. We use the training dataset during the training procedure. For MNIST, Fashion-MNIST and CIFAR-10 datasets, the test dataset in each of these datasets contains 10,000 images, and we use the test dataset during the validation procedure. The test dataset is an unseen dataset during the training procedures.
5.2 Evaluation Metrics

5.2.1 The Mean of Correctness Difference (MCD)

The MCD value is described in Definition 1.1.8, and the MCD value measures the mean of the difference between the predicted probability of the adversarial image and the predicted probability of the correct labels on the adversarial dataset $I'$, that is,

$$
MCD(I', Y; f_w) = \frac{\sum_{I_i' \in I'} |\max(p(y_i | I_i' ; w)) - p(y_{t_i} | I_i' ; w)|}{|I'|}.
$$

where $I_i'$ is an adversarial image in the adversarial dataset, $I_i' \in I'$, $y_i$ is the possible label of $I_i'$, $y_{t_i}$ is the true label of $I_i'$, $w$ is the parameters of neural network $f_w$. During the training procedure, if the model becomes robust, the MCD value is decreasing.

5.2.2 Accuracy

Accuracy measures the prediction capability of a model. We adopt the following definition

$$
\text{Accuracy} = \frac{\text{number of correct predictions}}{\text{total number of predictions}}
$$

Specifically, in the machine learning area, we place more emphasis on the accuracy of the unseen data, which represents the generalization capability of a model.

5.3 Experimental Results and Analysis

In this section, we report the experimental results of four algorithms and provide a brief analysis of the experiments’ results for each algorithm. In our experiments, for MNIST and Fashion-MNIST datasets, we use a convolutional neural network LeNet-5 [14] as the structure of the neural network model in each algorithm. For CIFAR-10 datasets, we use a ResNet [10] as the structure of the neural network model in each algorithm. In graphs of the experimental results, the MCD value is the y-axis because we tried to measure the robustness of the model at each epoch. The number of the epoch is the x-axis value because we have to train the model until the MCD values converge. The number of the epoch is a consistent and obvious measurement to observe the performance of each algorithm at each training epoch.

For each algorithm, at the beginning of each training epoch, we train the model with the accumulated training dataset that is mixed with the original dataset and the accumulated adversarial images. In our experiments, we generate adversarial images using three adversarial attack algorithms, which are DeepFool [18], Fast Gradient Sign Method (FGSM) [9]
and Salt and Pepper Noise Attack (SPNA) [22]. For each algorithm, the first part of the experiments involves attacking the model with one adversarial attack algorithm, and the second part of the experiments involves attacking the model with all three adversarial attack algorithms simultaneously, which means that we use three adversarial attack algorithms to attack the model during the training procedure.

Among these experimental results, the MCD value at the 0th epoch is small. The reason is that a model is only trained with the original dataset at the 0th epoch, and an adversarial attack algorithm can easily attack the model by changing a little bit features on an original image. These slight changes on an original image will lead the difference between the predicted probability of the adversarial image and the predicted probability of the correct label of the adversarial image to be small. Since the MCD value measures the mean of the difference between the predicted probability of the adversarial image and the predicted probability of the correct labels on the adversarial dataset, the MCD value is small at the 0th epoch. At the first epoch, the MCD value increases. The reason is that the model at the first epoch is trained by the original images and the adversarial images that are generated based on the previous epoch. The decision boundary at the 1st epoch is more clear and appropriate than the decision boundary at the 0th epoch. An adversarial attack algorithm needs to change more features than the previous epoch on each image to push the image from the correct decision boundary to the other incorrect decision boundaries. The more changes on an original image lead the difference between the predicted probability of the adversarial image and the predicted probability of the correct label of the adversarial image to be larger. Eventually, the MCD value will decrease and converge to a value. It is because the decision boundary is approaching to an appropriate position after several training epochs, and the trained model is becoming robust and well-trained. Adversarial attack algorithms are hard to attack the well-trained model.

5.3.1 Experimental Results and Analysis of Accumulating Adversarial Training Algorithm (Algorithm 1)

In this section, we present the performance of the Accumulating Adversarial Training Algorithm (Algorithm 1) on the MNIST dataset, the Fashion-MNIST dataset and the CIFAR-10 dataset. From the experimental results, we can observe that after 30 to 50 training epochs the MCD values tend to become smaller and smaller until they converge. Meanwhile, the Accuracy of the adversarial test dataset keeps growing until it converges. Both pieces of evidence show that a model trained by Algorithm 1 is robust, and a model trained by Algorithm 1 can defend one or several adversarial algorithms.

One Adversarial Attack Algorithm. In this part, we use one adversarial attack algorithm to attack the model during the training procedure of Algorithm 1. Figure 5.1, Figure 5.2 and Figure 5.3 show the MCD values and the Accuracy values at each training epoch.
with the MNIST dataset, the Fashion-MNIST dataset and the CIFAR-10 dataset respectively, when the training process of Algorithm 1 is under attack from different adversarial attack algorithms.

We can observe that under different adversarial attack algorithms the Accuracy values on the adversarial test dataset increase dramatically during the first 10 epochs and remain stable during the last 10 epochs. Facing different adversarial attack algorithms, the MCD values of the model decrease sharply at the first 10 epochs and converge into a small value during the last 10 training epochs; this represents that the model is robust to an adversarial attack algorithm.

**Three Adversarial Attack Algorithms.** In this part, we use three adversarial attack algorithms to attack the model at the same time during the training progress of Algorithm 1.

Figure 5.4 (a), 5.5 (a) and 5.6 (a) shows the MCD values at each training epoch under the attack of three adversarial attack algorithms during the training process of Algorithm 1 on the MNIST dataset, the Fashion-MNIST dataset and the CIFAR-10 dataset respectively. Because the model has different defense capacity for different adversarial attack algorithms, we use coloured lines to show the MCD values of the different adversarial attack algorithms. According to Figure 5.4 (a), Figure 5.5 (a) and Figure 5.6 (a), we can observe that the MCD values of different adversarial attack algorithms drop significantly at the first 10 training epochs and converge to small values eventually, and this indicates that the model trained by Algorithm 1 is robust to three adversarial attack algorithms. Figure 5.4 (b) – (d), Figure 5.5 (b) – (d) and Figure 5.6 (b) – (d) show the Accuracy values at each training epoch of the model to different adversarial test datasets generated by different adversarial attack algorithms on the MNIST dataset, the Fashion-MNIST dataset and the CIFAR-10 dataset respectively.

To be specific, we measure the Accuracy of the adversarial test dataset generated by the specific adversarial attack algorithm. For example, Figure 5.4 (b) represents the Accuracy of the model for the adversarial test dataset, which is generated by attacking the MNIST test dataset with DeepFool on the MNIST dataset. Figure 5.4 (b) – (d), Figure 5.5 (b) – (d) and Figure 5.6 (b) – (d) provide evidence that the Accuracy values of different adversarial test datasets generated by different adversarial attack algorithms grow fast during the first 10 epochs, and then the Accuracy values grow slowly until they converge to a high accuracy value (around 80% to 95%), which means the model can classify the majority of these unseen adversarial examples. Since we accumulate the adversarial examples that are generated by three different adversarial attack algorithms, the trained model has different ability on defending against different adversarial attack algorithms during the training procedure. For example, in Figure 5.4 (b) and Figure 5.4 (d), there is a zigzag shape change of the Accuracy values between epoch 2 and epoch 3. However, between epoch 2 and epoch 3, the Accuracy in Figure 5.4 (c) increased slightly. It is because the model has the stronger
Figure 5.1: MCD values and Accuracy values at each training epoch on the MNIST dataset, when the training process of Algorithm 1 is being attacked by different adversarial attack algorithms. The caption for each subfigure declares the name of each adversarial attack algorithm.
Figure 5.2: MCD values and Accuracy values at each training epoch on the Fashion-MNIST dataset, when the training process of Algorithm 1 is being attacked by different adversarial attack algorithms. The caption for each subfigure declares the name of each adversarial attack algorithm.
Figure 5.3: MCD values and Accuracy values at each training epoch on the CIFAR-10 dataset, when the training process of Algorithm 1 is being attacked by different adversarial attack algorithms. The caption for each subfigure declares the name of each adversarial attack algorithm.
(a) MCD values at each training epoch under the attacking of DeepFool, FGSM and SPNA simultaneously

(b) DeepFool: Accuracy at each epoch
(c) FGSM: Accuracy at each epoch
(d) SPNA: Accuracy at each epoch

Figure 5.4: (a) the MCD values at each training epoch under attack of three adversarial attack algorithms during the training progress of Algorithm 1 on the MNIST dataset. Because the model has different defense capacities against different adversarial attack algorithms, we use coloured lines to show the MCD values of the different adversarial attack algorithms. (b)-(d) Accuracy values for each of the three adversarial attack algorithms at each training epoch on the MNIST dataset.

ability to defend the adversarial images generated by FGSM than the ability to defend the adversarial images generated by DeepFool and SPNA, which can be observed from Figure 5.4 (a) as the MCD values of DeepFool and SPNA is higher than FGSM. Moreover, at the very first few training epochs, the model is not well-trained, and the model may put more effort into fitting the adversarial images generated by one adversarial attack algorithm than the adversarial images generated by the other adversarial attack algorithms. This causes a fluctuation on the accuracy values at the first few training epochs, when the decision boundary does not been adjusted appropriately.

Analysis. These experimental results show that Algorithm 1 can make the model robust to one or several adversarial attack algorithms. Meanwhile, Algorithm 1 leads the
Figure 5.5: (a) the MCD values at each training epoch under attack of three adversarial attack algorithms during the training progress of Algorithm 1 on the Fashion-MNIST dataset. Because the model has different defense capacities against different adversarial attack algorithms, we use coloured lines to show the MCD values of the different adversarial attack algorithms. (b)-(d) Accuracy values for each of the three adversarial attack algorithms at each training epoch on the Fashion-MNIST dataset.
Figure 5.6: (a) the MCD values at each training epoch under attack of three adversarial attack algorithms during the training progress of Algorithm 1 on the CIFAR-10 dataset. Because the model has different defense capacities against different adversarial attack algorithms, we use coloured lines to show the MCD values of the different adversarial attack algorithms. (b)-(d) Accuracy values for each of the three adversarial attack algorithms at each training epoch on the CIFAR-10 dataset.
model to be an accurate classifier to those adversarial examples. We analyze the reasons as follows:

- Algorithm 1 accumulate the adversarial examples created in the current and the previous training epoch. At each training epoch, Algorithm 1 trained the model with the original dataset and the adversarial training datasets that are generated on the current and the previous training epochs. The Accumulating Adversarial Training forces the model to learn the key features in images to prevent the model from being misled by adversarial perturbation in the adversarial examples.

- By accumulating the adversarial examples that are generated at each training epoch, we accumulate the contributions made by each single adversarial example to adjust the decision boundary, which leads the decision boundary to be closer to the correct position. A model with an appropriate decision boundary is an accurate classifier to a specific dataset.

5.3.2 Experimental Results and Analysis of Accumulating Adversarial Training Algorithm with Adaptive Boosting Method (Algorithm 2)

In this section, we present the performance of the Accumulating Adversarial Training Algorithm with Adaptive Boosting Method (Algorithm 2) on the MNIST dataset, Fashion-MNIST dataset and CIFAR-10 dataset. In each subfigure we compare the MCD values of Algorithm 1 (named as “noboosting” in the subfigures) with the MCD values of Algorithm 2 (named as “Adaboosting” in the subfigures). Generally, for each adversarial attack algorithm, the MCD value at each epoch of Algorithm 2 is smaller than the MCD value at each epoch of Algorithm 1. These results show that the Adaptive Boosting Method boosts the performance of the Accumulating Adversarial Training Algorithm.

One Adversarial Attack Algorithm. Figure 5.7, Figure 5.8 and Figure 5.9 show the MCD values at each training epoch under the attack of an adversarial attack algorithm during the training procedure of Algorithm 2 on the MNIST dataset, the Fashion-MNIST dataset and the CIFAR-10 dataset respectively. For different adversarial attack algorithms, compared with Algorithm 1, the MCD values of Algorithm 2 at each epoch are lower. Moreover, in general, the MCD values at the training epoch of Algorithm 2 decrease rapidly at the first 6 training epochs and converge to a small value eventually. The results of the MCD values in Figure 5.7, Figure 5.8 and Figure 5.9 show that Algorithm 2 can make a model robust to a specific adversarial attack algorithm. Moreover, the model trained by Algorithm 2 is more robust than the model trained by Algorithm 1, because the MCD values of the model trained by Algorithm 2 are smaller than the MCD values of the model trained by Algorithm 1.
Three Adversarial Attack Algorithms. Figure 5.10, Figure 5.11 and Figure 5.12 show the MCD values at each training epoch of each adversarial attack algorithm when we use all three adversarial attack algorithms to generate the adversarial images during the training procedure of Algorithm 2 on the MNIST dataset, the Fashion-MNIST dataset and the CIFAR-10 respectively. The MCD values of Algorithm 2 decline sharply at the first 5 training epochs and converge to a small value eventually, which represents that Algorithm 2 can make a model robust to multiple adversarial attack algorithms. Obviously, for each adversarial attack algorithm, compared with Algorithm 1, the model trained by Algorithm 2 gives a smaller MCD value at each epoch. This means that with the attacking of three adversarial attack algorithms simultaneously, the model trained by Algorithm 2 exhibits
Figure 5.9: MCD values at each training epoch under the attack of one adversarial attack algorithm during the training procedure of Algorithm 2 on the CIFAR-10 dataset. In subfigures of the MCD values, we compare the result of Algorithm 1 and Algorithm 2. The “AdaBoosting” line represents the MCD values of Algorithm 2, and the “noboosting” line represents the MCD values of Algorithm 1.

the better performance in terms of defending itself against the adversarial images generated by these three adversarial attack algorithms than the model trained by Algorithm 1.

**Analysis.** The above experimental results show that Algorithm 2 can make the model robust to one or several adversarial attack algorithms. Moreover, the model trained by Algorithm 2 is more robust than the model trained by Algorithm 1. We analyze the reasons as follows:

- Compared with Algorithm 1, the MCD values of Algorithm 2 converge fast, which experimentally shows that Algorithm 2 quickly adjusted the decision boundary into an appropriate position. The reason for this is that we put more weights on those misclassified images, and those misclassified images with heavier loss contribute more to the boundary adjustment, which leads the decision boundary to be adjusted into an appropriate position quickly.

- Compared with Algorithm 1, Algorithm 2 puts more weights on those misclassified images, which pushes the model to put more effort into learning those important and missed features on the reweighted misclassified images and make a better classifier. As a result, Algorithm 2 can reach a smaller MCD value than Algorithm 1 at each training epoch.

### 5.3.3 Experimental Results and Analysis of Accumulating Adversarial Training Algorithm with Group Sampling Boosting method (Algorithm 3)

In this section, we present the performance of the Accumulating Adversarial Training Algorithm with Group Sampling Boosting Method (Algorithm 3) on the MNIST dataset, the
Figure 5.10: MCD values at each training epoch of Algorithm 2 under the attack of three adversarial attack algorithms on the MNIST dataset. In the figures of the MCD values, we compare the result of Algorithm 1 and Algorithm 2. The “Adaboosting” line represents the MCD values of Algorithm 2, and the “noboosting” line represents the MCD values of Algorithm 1.

Figure 5.11: MCD values at each training epoch of Algorithm 2 under the attack of three adversarial attack algorithms on the Fashion-MNIST dataset. In the figures of the MCD values, we compare the result of Algorithm 1 and Algorithm 2. The “AdaBoosting” line represents the MCD values of Algorithm 2, and the “noboosting” line represents the MCD values of Algorithm 1.
Figure 5.12: MCD values at each training epoch of Algorithm 2 under the attack of three adversarial attack algorithms on the CIFAR-10 dataset. In the figures of the MCD values, we compare the result of Algorithm 1 and Algorithm 2. The “AdaBoosting” line represents the MCD values of Algorithm 2, and the “noboosting” line represents the MCD values of Algorithm 1.

Fashion-MNIST dataset and the CIFAR-10 dataset. Generally, the MCD values decrease monotonically no matter what sample ratios are set, which indicates that Algorithm 3 makes the model robust. In general, the higher the sample ratio, the smaller the MCD value of each epoch. This evidence experimentally shows that the Group Sampling Boosting Method can boost the performance of the Accumulating Adversarial Training Algorithm.

One Adversarial Attack Algorithm. Figure 5.13, Figure 5.14 and Figure 5.15 show the MCD value at each training epoch, when we use one adversarial attack algorithm to attack the model during the training iterations of Algorithm 3 on the MNIST dataset, the Fashion-MNIST dataset and the CIFAR-10 dataset respectively. For different adversarial attack algorithms, the larger the sample ratio, the smaller the MCD value at each epoch. In general, among these three adversarial attack algorithms, if the sample ratio is larger than 0.1, the model trained by Algorithm 3 is more robust than the model trained by Algorithm 1. Thus, we can tell that if the sample ratio has been set appropriately, the group sampling boosting method improves the performance of the model.

Three Adversarial Attack Algorithms. Figure 5.16, Figure 5.17 and Figure 5.18 show the MCD values at each training epoch of each adversarial attack algorithm when we use all three adversarial attack algorithms to generate adversarial images in the training procedure of Algorithm 3 on the MNIST dataset, the Fashion-MNIST dataset and the CIFAR-10 dataset respectively. In general, the higher the sample ratio, the smaller the MCD value converges to. If the sample ratio is set to a value larger than or equal to 0.1, Algorithm 3 can gain a more robust model than Algorithm 1, because the MCD value of Algorithm 3 converges to a smaller value than Algorithm 1 converged to.
Figure 5.13: These figures show the MCD values at each training epoch, when there is only one adversarial attack algorithm on the training procedure of Algorithm 3 on the MNIST dataset. For different adversarial attack algorithms, the name of the adversarial attack algorithm is shown in the caption under each subfigure. In each subfigure, the dashed coloured line represents the MCD values of Algorithm 1. In the legend, the number means the sample ratio.

Figure 5.14: These figures show the MCD values at each training epoch, when there is only one adversarial attack algorithm on the training procedure of Algorithm 3 on the Fashion-MNIST dataset. For different adversarial attack algorithms, the name of the adversarial attack algorithm is shown in the caption under each subfigure. In each subfigure, the dashed coloured line represents the MCD values of Algorithm 1. In the legend, the number means the sample ratio.
Figure 5.15: These figures show the MCD values at each training epoch, when there is only one adversarial attack algorithm on the training procedure of Algorithm 3 on the CIFAR-10 dataset. For different adversarial attack algorithms, the name of the adversarial attack algorithm is shown in the caption under each subfigure. In each subfigure, the dashed coloured line represents the MCD values of Algorithm 1. In the legend, the number means the sample ratio.

Figure 5.16: MCD value at each training epoch of each adversarial attack algorithm when we use all three adversarial attack algorithms in the training procedure of Algorithm 3 on the MNIST dataset. In each figure we compare the MCD value of Algorithm 1 (named “noboosting” in the figure) with the MCD value of Algorithm 3 with different sample ratio setting (For example, "0.05" in the legend means the sample ratio sets as 0.05).
(a) DeepFool: MCD values at each epoch
(b) FGSM: MCD values at each epoch
(c) SPNA: MCD values at each epoch

Figure 5.17: MCD value at each training epoch of each adversarial attack algorithm when we use all three adversarial attack algorithms in the training procedure of Algorithm 3 on the Fashion-MNIST dataset. In each figure we compare the MCD value of Algorithm 1 (named “noboosting” in the figure) with the MCD value of Algorithm 3 with different sample ratio setting (For example, “p = 0.05” in the legend means the sample ratio sets as 0.05).

(a) DeepFool: MCD values at each epoch
(b) FGSM: MCD values at each epoch
(c) SPNA: MCD values at each epoch

Figure 5.18: MCD value at each training epoch of each adversarial attack algorithm when we use all three adversarial attack algorithms in the training procedure of Algorithm 3 on the CIFAR-10 dataset. In each figure we compare the MCD value of Algorithm 1 (named “noboosting” in the figure) with the MCD value of Algorithm 3 with different sample ratio setting (For example, ”p = 0.05” in the legend means the sample ratio sets as 0.05).
Analysis. The above experimental results show that Algorithm 3 can make the model robust to one or several adversarial attack algorithms. We also observe that the higher the sample ratio, the smaller the MCD value converges to. Moreover, in Algorithm 3, if we set the sample ratio value larger than 0.1, the model trained by Algorithm 3 is more robust than the model trained by Algorithm 1. We analyze the reasons as follows:

- Because the sample dataset is drawn from the same true population as the training dataset using the simple random sampling method, the model trained with the sample dataset can learn the same features as the model trained with the training dataset. Thus, the trained neural network model fits the true population dataset.

- The larger the sample size we have, the more information the neural network learns. Thus, the larger the sampling ratio we pick, the more contributions we get for adjusting the decision boundary and the faster the convergence happens.

- If the sample ratio has been set appropriately, the model can put effort into learning partially important features at each training epoch. After several training epochs, the model can gain more important features on classification. As a result, the Group Sampling Boosting Method can lead to a more robust model than Algorithm 1.

5.3.4 Experimental Results and Analysis of Accumulating Adversarial Training Algorithm with the Stochastic Mini Batch Boosting Method (Algorithm 4)

In this section, we present the performance of the Accumulating Adversarial Training Algorithm with the Stochastic Mini Batch Boosting Method (Algorithm 4) on the MNIST dataset, the Fashion-MNIST dataset and the CIFAR-10 dataset. Generally, the MCD values decrease monotonically when we set the batch size to different values, which represents that Algorithm 4 makes the model robust, and the running time of Algorithm 4 with different batch sizes is smaller than the running time of Algorithm 1. We can also observe that the higher the batch size, the smaller the MCD value of each epoch.

One Adversarial Attack Algorithm. Figure 5.19, Figure 5.20 and Figure 5.21 shows the MCD values at each training epoch under attack of different adversarial attack algorithms during the training procedure of Algorithm 4 on the MNIST dataset, the Fashion-MNIST dataset and the CIFAR-10 dataset respectively. For different adversarial attack algorithms, the MCD values decrease monotonically. In Figure 5.19, Figure 5.20 and Figure 5.21, the downward trend in the MCD values shows that Algorithm 4 can make a model be robust to a specific adversarial attack algorithm. For DeepFool and FGSM, if the batch size is larger than 256, the model trained by Algorithm 4 is more robust than the model trained by Algorithm 1. Thus, we can tell that if the batch size has been set appropriately, Algorithm 4 improves the performance of the model.
Figure 5.19: These figures show the MCD values at each training epoch under the attack of different adversarial attack algorithms during the training procedure of Algorithm 4 on the MNIST dataset. For different adversarial attack algorithms, the name of the adversarial attack algorithm has been provided in the caption under each figure. In the figure, the legend "noboosting" refers to the MCD values of Algorithm 1.

Figure 5.20: These figures show the MCD values at each training epoch under the attack of different adversarial attack algorithms during the training procedure of Algorithm 4 on the Fashion-MNIST dataset. For different adversarial attack algorithms, the name of the adversarial attack algorithm has been provided in the caption under each figure. In the figure, the legend "noboosting" refers to the MCD values of Algorithm 1.
Figure 5.21: These figures show the MCD values at each training epoch under the attack of different adversarial attack algorithms during the training procedure of Algorithm 4 on the CIFAR-10 dataset. For different adversarial attack algorithms, the name of the adversarial attack algorithm has been provided in the caption under each figure. In the figure, the legend "noboosting" refers to the MCD values of Algorithm 1.

Figure 5.22: These subfigures show the MCD values at each training epoch of each adversarial attack algorithm when we use all three adversarial attack algorithms in the training procedure of Algorithm 4 on MNIST dataset. In each subfigure we compare the MCD values of Algorithm 1 (named as “noboosting” in the subfigures) with the MCD values of Algorithm 4, when we set the batch size as different values.
Figure 5.23: These subfigures show the MCD values at each training epoch of each adversarial attack algorithm when we use all three adversarial attack algorithms in the training procedure of Algorithm 4 on Fashion-MNIST dataset. In each subfigure we compare the MCD values of Algorithm 1 (named as “noboosting” in the subfigures) with the MCD values of Algorithm 4, when we set the batch size as different values.

Figure 5.24: These subfigures show the MCD values at each training epoch of each adversarial attack algorithm when we use all three adversarial attack algorithms in the training procedure of Algorithm 4 on CIFAR-10 dataset. In each subfigure we compare the MCD values of Algorithm 1 (named as “noboosting” in the subfigures) with the MCD values of Algorithm 4, when we set the batch size as different values.
Three Adversarial Attack Algorithms. Figure 5.22, Figure 5.23 and Figure 5.24 show the MCD values at each training epoch under attack of the three adversarial attack algorithms during the training procedure of Algorithm 4 on the MNIST dataset, the Fashion-MNIST and the CIFAR-10 dataset respectively. For different adversarial attack algorithms, the MCD values decrease with fluctuation. These results of the MCD values show that Algorithm 4 can make a model be robust to several adversarial attack algorithms.

Analysis. The above experimental results show that Algorithm 4 can make the model robust to one or several adversarial attack algorithms. We analyze the reasons as follows:

- Compared with Algorithm 1, which attacks all images in an original training dataset by each adversarial attacking algorithm to generate adversarial examples, Algorithm 4 only generates a batch size of adversarial examples. As a result, Algorithm 4 spends less time at each training epoch as long as the batch size is less than the size of the original training dataset.

- We adjusted the decision boundary whenever we get a batch of adversarial examples, which makes the model learn a partial of features of the training dataset at each iteration. By accumulating the batch of adversarial examples that are generated at each training epoch, the model learns more and more features of the dataset and updates the neural network to approach the true decision boundary.

5.3.5 Comparison Among Boosting Algorithms

In this section, we compare the performance of Algorithm 2, Algorithm 3 and Algorithm 4. In general, the performance of Algorithm 2 is better than Algorithm 3 and Algorithm 4, because the MCD value of Algorithm 2 is generally smaller than the MCD values of other algorithms at each epoch.

One Adversarial Attack Algorithm. Figure 5.25, Figure 5.26 and Figure 5.27 show the MCD values at each training epoch under the attack of different adversarial attack algorithms during the training procedure of Algorithm 2, Algorithm 3 and Algorithm 4 on the MNIST dataset, the Fashion-MNIST dataset and the CIFAR-10 dataset respectively. For different adversarial attack algorithms, the MCD values decrease monotonically with some fluctuation. In general, the MCD value of Algorithm 2 is smaller than the MCD values of other algorithms at each epoch. Thus, among the three boosting algorithms, Algorithm 2 works the best in terms of improving the performance of the model. For Algorithm 3, if we set the sample ratio as 0.5, it is the second-best method for boosting the performance of the model. For Algorithm 4, if we set the batch size as 512, it is the third-best method for making the model robust. However, Algorithm 4 with batch size as 512 takes less time than Algorithm 3 with a sample ratio of 0.5 at each training epoch. According to the size
of our datasets, Algorithm 3 with a sample ratio of 0.5 has to attack 30000 images at each training epoch, whereas Algorithm 4 only needs to attack 512 images.

Three Adversarial Attack Algorithms. Figure 5.28, Figure 5.29 and Figure 5.30 show the MCD values at each training epoch under the attack of different adversarial attack algorithms during the training procedure of Algorithm 2, Algorithm 3, and Algorithm 4 on the MNIST dataset, the Fashion-MNIST dataset and the CIFAR-10 dataset. For different adversarial attack algorithms, the MCD values decrease monotonically with some fluctuation. In general, the MCD value of Algorithm 2 is smaller than the MCD values of other algorithms at each epoch. Thus, among the three boosting algorithms, Algorithm 2 works the best in terms of improving the performance of the model. For Algorithm 3, if we set the sample ratio larger or equal to 0.3, Algorithm 3 is the second-best method for improving the performance of the model.

Analysis. The above experimental results show that Algorithm 2 is the boosting algorithm that best improves the performance of the model. For Algorithm 3 and Algorithm 4, the larger the size of the generated adversarial images at each training epoch, the better the performance in terms of making the model robust. We analyze the reasons for this as follows:

- Compared with Algorithm 3 and Algorithm 4, the MCD values of Algorithm 2 converge fast, which experimentally shows that Algorithm 2 quickly adjusted the decision boundary into an appropriate position. The reason for this is that we put more weights on those misclassified images, and those misclassified images with heavier loss con-
Figure 5.26: These figures show the MCD values at each training epoch under the attack of different adversarial attack algorithms during the training procedure of Algorithm 2, Algorithm 3, and Algorithm 4 in the Fashion-MNIST dataset. For different adversarial attack algorithms, the name of the adversarial attack algorithm has been provided in the caption under each figure. In the figure, the legend “Adaboosting” refers to the MCD values of Algorithm 2. The legends “p = 0.01”, “p = 0.05”, “p = 0.1”, “p = 0.3” and “p = 0.01” refer to the MCD values of Algorithm 3 when we set the probability to different values. The legend “batch size = 128” refers to the MCD values of Algorithm 4 when batch size sets as 128.

Figure 5.27: These figures show the MCD values at each training epoch under the attack of different adversarial attack algorithms during the training procedure of Algorithm 2, Algorithm 3, and Algorithm 4 in the CIFAR-10 dataset. For different adversarial attack algorithms, the name of the adversarial attack algorithm has been provided in the caption under each figure. In the figure, the legend “Adaboosting” refers to the MCD values of Algorithm 2. The legends “p = 0.01”, “p = 0.05”, “p = 0.1”, “p = 0.3” and “p = 0.01” refer to the MCD values of Algorithm 3 when we set the probability to different values. The legend “batch size = 128” refers to the MCD values of Algorithm 4 when batch size sets as 128.
Figure 5.28: These figures show the MCD values at each training epoch under the attack of all three adversarial attack algorithms during the training procedure of Algorithm 2, Algorithm 3, and Algorithm 4 in the MNIST dataset and the Fashion-MNIST dataset. For different adversarial attack algorithms, the name of the adversarial attack algorithm has been provided in the caption under each figure. In the figure, the legend “Adaboosting” refers to the MCD values of Algorithm 2. The legends “p = 0.01”, “p = 0.05”, “p = 0.1”, “p = 0.3” and “p = 0.01” refer to the MCD values of Algorithm 3 when we set the probability to different values. The legend “batch size = 128” refers to the MCD values of Algorithm 4 when batch size sets as 128.

Figure 5.29: These figures show the MCD values at each training epoch under the attack of all three adversarial attack algorithms during the training procedure of Algorithm 2, Algorithm 3, and Algorithm 4 in the Fashion-MNIST dataset. For different adversarial attack algorithms, the name of the adversarial attack algorithm has been provided in the caption under each figure. In the figure, the legend “Adaboosting” refers to the MCD values of Algorithm 2. The legends “p = 0.01”, “p = 0.05”, “p = 0.1”, “p = 0.3” and “p = 0.01” refer to the MCD values of Algorithm 3 when we set the probability to different values. The legend “batch size = 128” refers to the MCD values of Algorithm 4 when batch size sets as 128.
Figure 5.30: These figures show the MCD values at each training epoch under the attack of all three adversarial attack algorithms during the training procedure of Algorithm 2, Algorithm 3, and Algorithm 4 in the CIFAR-10 dataset. For different adversarial attack algorithms, the name of the adversarial attack algorithm has been provided in the caption under each figure. In the figure, the legend “Adaboosting” refers to the MCD values of Algorithm 2. The legends “p = 0.01”, “p = 0.05”, “p = 0.1”, “p = 0.3” and “p = 0.01” refer to the MCD values of Algorithm 3 when we set the probability to different values. The legend “batch size = 128” refers to the MCD values of Algorithm 4 when batch size sets as 128.

- At each training epoch, Algorithm 2 attacks the whole original dataset to generate the adversarial images, whereas Algorithm 3 and Algorithm 4 only attack a portion of the original dataset to generate the adversarial images. As a result, a model trained by Algorithm 2 can learn more features at each training epoch than a model trained by the other algorithms.

- The larger the sample size we have in Algorithm 3 and the larger the batch size we set in Algorithm 4, the more information the neural network learns. Thus, the larger the sampling ratio we pick and the larger the batch size we pick, the more contributions we get for adjusting the decision boundary and the faster the convergence happens.
Chapter 6

Conclusions

In this thesis, we formulate the problem of improving the robustness of neural networks against adversarial attack algorithms. In order to tackle this problem, we propose the Accumulating Adversarial Training Algorithm. In our algorithm, we train a model iteratively using a dataset that includes the original dataset, the adversarial dataset generated in the current iteration, and the adversarial datasets generated in the previous iterations. To measure and evaluate the robustness of the neural network, we propose a new measurement index named the Mean of Correctness Difference (MCD). To improve the performance of the Accumulating Adversarial Training Algorithm, we propose three boosting methods: the Adaptive Boosting method, the Group Sampling Boosting method and the Stochastic Mini Batch Boosting method.

We evaluate the Accumulating Adversarial Training Algorithm and the Accumulating Adversarial Training Algorithm with the three different boosting methods using the MNIST dataset, the Fashion-MNIST dataset and the CIFAR-10 dataset. The experimental results indicate that a trained model can defend against one or several adversarial attack algorithms, and the boosting methods improve the performance of a trained model.

As for future work, we can also consider the following directions.

- **There might be hierarchy or inclusion relationship between adversarial attack algorithms.** Suppose there is an adversarial example $I'_1$ that is generated by an adversarial attack algorithm $A_1$, and an adversarial example $I'_2$ that is generated by an adversarial attack algorithm $A_2$, if the relationship between any $I'_1$ and $I'_2$ is $I'_1 = I'_2 + r$, we think there is a hierarchy or inclusion relationship between $A_1$ and $A_2$. Can we design a method to figure out the hierarchy or inclusion relationship between the adversarial attack algorithms in order to make the training procedure of the Accumulating Adversarial Training Algorithm more efficient? If we can discover the hierarchy or inclusion relationship between the adversarial attack algorithms, we do not need to include many adversarial attack algorithms in the training procedure of the Accumulating
Adversarial Training Algorithm. This will save time otherwise spent on the steps of generating the adversarial examples and the training procedure.

- **Can we find more ways to boost the Accumulating Adversarial Training Algorithm?**
  In our thesis, we use three methods to boost the Accumulating Adversarial Training Algorithm, but perhaps we could develop more methods to boost the Accumulating Adversarial Training Algorithm. For example, we can try to tackle the main features of images, and extra those features to train a model.

- **Can we extend our method for interpreting machine learning models?** By using our method, we can gain a robust model and observe the changing of adversarial examples in each training iteration. Analyzing these data and summarizing the changes in the model parameters may inspire us to solve the machine learning interpretation problem.
Bibliography


