Interim Report:

Robustness in Neural Networks

Submitted December 11, 2020, in partial fulfilment of the conditions for the award of the degree of
BSc Hons Computer Science with Artificial Intelligence

Can Zhou

20031818

Supervised by Dr. Amin Farjudian

School of Computer Science
University of Nottingham, Ningbo China

Abstract

Robustness is the ability of a neural network (NN) to cope with noise in input data. Because of security implication, certifying the robustness of neural networks has become an important research topic. However, one of key flaws in the research on measuring robustness in NN is that the robustness evaluation result is often intertwined with the attack algorithms and currently, there is no standard metric in this field. This report will start by investigating and summarizing state-of-art related research. Subsequently, a novel method based on interval arithmetic for evaluating the robustness of neural networks will be proposed, designed, computed, evaluated and compared to other exist robustness metrics.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Robustness is the ability of a neural network (NN) to cope with noise in input data. A neural network is robust if tiny variations in inputs do not lead to significant changes in outputs [1]. Although neural networks (NNs) have achieved exceptional performance in many machine learning tasks like image classification [2], object detection [3] and pose estimation [4], recent studies have highlighted their lack of robustness against adversarial examples [5]. Even in state-of-the-art neural networks such as Faster R-CNN [6], ResNet [7] and FaceNet [8], their robustness is vulnerable, that a slight perturbation applied to the input will confuse the neural network to produce erroneous outputs. One of the well-known examples is the one-pixel attack proposed by Su et al.(2019) [9]. Their study result shows that by modifying just one pixel, 67.97% of images in CIFAR-10 testing dataset can be disarrayed to incorrect target classes.

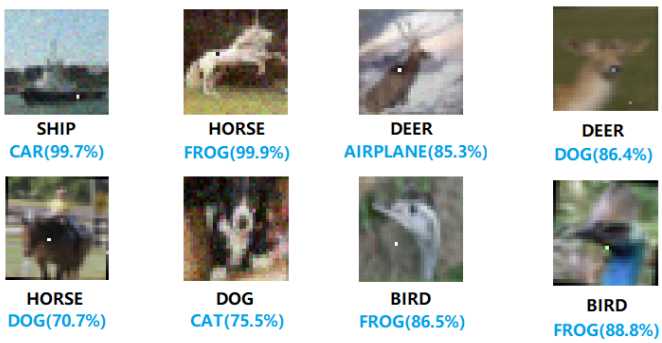

Figure 1.1: One Pixel Attack: the output of NNs can be easily altered by changing one pixel in the inputs

If these neural networks are used for safety-critical purposes, for instance, the criminal justice system, control of aeroplanes and road traffic, these may have detrimental conse-

quences. As neural networks are becoming a crucial method deployed in a wide variety of applications, including safety-critical systems, certifying neural networks robustness against adversarial perturbation has become an important research topic.

1.2 Motivation

Technically, the study of robustness in the neural network has three primary lines: studying the ways in which neural networks are vulnerable to attack; evaluating the robustness of neural networks and improving NNs robustness. In principle, these three challenges should be addressed separately. This indicates that the estimation of a neural networks robustness should not be affected by ways of attack, specifically when evaluating a new algorithm for robustness enhancement.

Unfortunately, at this stage, there is no reliable, standardized metric to evaluate the robustness of neural networks, which leads existing approaches to use different scales to measure a target neural networks robustness. In particular, many existing studies regard the attacking success rate and invalidation rate of the adversarial example based on a specific attack algorithm as robustness metrics [10]. As a result, the robustness evaluation result is intertwined with the attack algorithms. It will also cause the analysis to be restricted by the comprehensiveness of the attack. Therefore, it is worth investigating the factors that may affect the robustness of neural networks and how robustness can be analysed reliably.

1.3 Aims and Objectives

The overall structure of this project is to start by investigating and summarizing recent research, addressing two aspects: adversarial examples and robustness analysis. The project would then concentrate on evaluating robustness in MLPs. A novel method based on an arithmetic interval will be designed. Next, the MLPs that approximate predefined maps (e.g. from R^n to R^m) will be trained and the proposed method and the methods from the literature will be applied to MLPs for comparing their performance. After that, the interval method will be used to measure robustness with respect to disruptions in

model parameters, e.g. weights and biases of the network.

The key objectives of this project are:

1. Scrutinizing existing studies to clarify factors that may lead to the lack of robustness.
2. Investigating previous methodologies used to evaluate robustness.
3. Proposing a novel method for robustness analysis based on interval arithmetic.
4. Comparing the performances between the proposed method and the established methods. (e.g. CLEVER [10])
5. Applying the interval method to evaluate robustness not only with respect to input perturbations, but also with regard to model parameters, e.g., weights and biases of the network.
6. Designing and implementing a comprehensive library that can automatically measure robustness for one-hidden layer MLPs with reasonable input dimensions via the proposed method.
7. Isolating interval boxes which give the maximum norm of the local Lipschitz constant.

Chapter 2

Related Work

This chapter presents an overview containing a background of the studies in this field, spanning two specific topics: attacking neural networks by using adversarial examples and measuring robustness. In the meantime, retrospective analysis and comparison of the primary research methods in this field are present to enhance the understanding of this project. All the notations used in the remaining of the report are summarized in Table 2.1.

Table 2.1: Table of Notation

Notation	Definition	Notation	Definition
d	input vector's dimensionality	$\delta \in \mathbb{R}^d$	perturbation / distortion
K	number of output classes	$\|\delta\|_p$	ℓ_p norm of distortion, $p \geq 1$
$f : \mathbb{R}^d \to \mathbb{R}^K$	neural network classifier	θ	model parameters
$x_0 \in \mathbb{R}^d$	original input vector	$J(\theta, x, y_{true})$	cost function
$x_a \in \mathbb{R}^d$	adversarial example	$D(.,.)$	distance metric
y_{true}	true class of given x	C	constant
$c(x)$	predicted class of given x	$t(x+\delta)$	customized adversarial loss
$\rho_{adv}(c)$	robustness of the classifier c	$\mathbb{E}_x$	expectation of data distribution
$\Delta_{p,min}$	minimum ℓ_p distortion of x_0	L_q^j	Lipschitz constant
β_L	lower bound of minimum distortion	L_{q,x_0}^j	local Lipschitz constant (targeted)
$B_p(x_0, R)$	hyper-ball with centre x_0 and radius R	L_{q,x_0}	local Lipschitz constant

2.1 Attacking Neural Networks

The adversarial example is a tweaked version of the original image by inserting noise to the clean image [11] and is deliberately perturbed to fool neural networks. This section provides several major methods to generate adversarial examples thereby attacking neural networks.

2.1.1 Fast Gradient Sign Method (FGSM)

Fast gradient sign method is one of the simplest methods proposed by Goodfellow et al.(2014) [12] to generate adversarial images. It is inspired by linearising the cost function and could obtain an optimal max-norm perturbation with ℓ_∞ constraint [13]. This could be done in closed form with the expense of one back-propagation call:

$$x_a = x_0 + \epsilon \cdot sign(\nabla_{x_0} J(\theta, x_0, y_{true})) \tag{2.1}$$

Therefore, the perturbation δ could be defined as:

$$\delta = \epsilon \cdot sign(\nabla_{x_0} J(\theta, x_0, y_{true})) \tag{2.2}$$

where ϵ is a hyper-parameter.

It is essential to mention that the fast gradient sign attack is intended to be fast since it does not require an iterative procedure to produce adversarial examples. However, it may not be optimal as it is not designed to generate minimal adversarial perturbations.

2.1.2 One Pixel Attack

For attack neural networks, an extreme case is when just modifies one pixel in the image, the classifier will be misled to give wrong predicted class. Intriguingly, Su et al. [9] stated their methods could fool 70.97% of the tested images in three different models by altering one pixel per image. They suggested that the neural networks average confidence in the incorrect label was 97.47% as well.

Based on the theory of Differential Evolution [14], one-pixel attack could be implemented to produce the adversarial examples. Su et al. firstly generated a series of 400 vectors in $\mathbb{R}^5$ for a clean image x_0, such that xy-coordinates and RGB values for a random pixel are included in each vector. Then, the items of the vectors are arbitrarily changed to construct children so that children compete in the next iteration with their parents for fitness, while the fitness criterion is the probabilistic predicted label of the NNs. The chosen modify pixel is the last surviving child.

2.1.3 Carlini and Wagner Attacks (C&W Attacks)

In the wake of defensive distillation against adversarial examples, Carlini and Wagner [15] introduced a set of adversarial attacks - C&W attacks. CW0, CW2 and CW1 are three different kinds of adversarial examples generated by C&W attacks based on the norms of ℓ_0, ℓ_2, ℓ_∞. Formally, the C&W-generated adversarial images are defined as the equation below [16]:

$$\min_{\delta} \; D(x_0, x_0 + \delta) + C \cdot t(x_0 + \delta) \quad \textit{suject to } x_0 + \delta \in [0, 1] \tag{2.3}$$

where the distance metrics $D(.,.)$ only apply to ℓ_0, ℓ_2, ℓ_∞ norms, and $g(x_0 + \delta)$ satisfies $g(x_0) < 0$ if and only if NN's prediction is the attack target. Moreover, in order to ensure $(x_0 + \delta)$ produces a valid image, the variable z is used to substitute δ:

$$\delta = \frac{1}{2}[tanh(z) + 1] - x_0 \tag{2.4}$$

For the benchmark datasets, such as MNIST, CIFAR-10, and ImageNet, on ordinarily trained NNs, C&W attacks achieve a one hundred percent attack success rate. They also undermine defensive distilled models that have failed to yield adverse samples from DeepFool.

2.1.4 DeepFool

Moosavi-Dezfooli et al. [17] proposed DeepFool - a new algorithm to iteratively compute a minimum norm adversarial disruption for a given image. Given an input x_0 and a classifier f and minimal perturbation δ may be abstained by:

$$\Delta(x_0; f) := \min_{\delta} \|\delta\|_2 \; \textit{subject to } c(x_0 + \delta) \neq c(x_0) \tag{2.5}$$

where $c(x)$ is the predicted class of given x

DeepFool defines $\Delta(x_0; f)$ as the robustness of f at point x_0, hence the robustness of classifier c is defined as:

$$\rho_{adv}(f) = \mathbb{E}_{x_0} \frac{\Delta(x_0; f)}{\|x_0\|_2} \tag{2.6}$$

Experiments reveal that the DeepFool is capable of measuring perturbations that are smaller than the perturbations measured by FGSM [12], while providing similar fooling rates on multiple benchmark datasets.

2.1.5 Attacking Methods Summarized and Comparison

The following Table 2.2 summarises and contrasts the attack methods mentioned above in terms of the type of test (black box or white box), the type of target, whether it is specific or universal, whether it is learning and the strength of the methods.

Table 2.2: Attacking Methods Summarized and Comparison

Method	**Black/White box**	**Targeted/Untargeted**	**Perturbation norm**	**Learning**	**Strength**
FGSM [12]	White	Targeted	ℓ_∞	One shot	3
One-pixel [9]	Black	Untargeted	ℓ_0	Iterative	2
C&W Attacks [15]	White	Targeted	ℓ_0, ℓ_2, ℓ_∞	Iterative	5
DeepFool [17]	White	Untargeted	ℓ_2, ℓ_2	Iterative	4

2.2 Measuring Robustness

A key issue in research on measuring robustness is that the robustness evaluation result is often intertwined with the attack algorithms. However, the attack algorithms easily overestimate the amount of perturbation needed to confuse the target neural network since the attack algorithms are not optimal and comprehensive. In other words, the methodology based on the attack algorithms gives an upper bound on the size of perturbations which could fool the model, but a lower bound is needed for safety guarantees [18]. This section will emphasize measuring the lower bound of neural networks' robustness and start by the analysis of formal robustness guarantees for a classifier via lower bound.

2.2.1 Theoretical Robustness Guarantees for Neural Networks

In 2017, Hein & Andriushchenko [19] provided a guaranteed lower bound for the robustness of one hidden layer MLP by using a local Lipschitz continuous condition. But it is challenging for neural networks with more than one hidden layer to derive their theory. Weng et al.(2018) [10] brought an extended formal robustness guarantees for a classifier based on Hein & Andriushchenko's theory. Their extended theory is universal, since Lipschitz's continuity of the classification function is the only required assumption. The following analysis of theoretical robustness guarantees for neural networks is relied on theory provided by Weng et al. [10].

Definition 1 (adversarial example). *Let $f : \mathbb{R}^d \to \mathbb{R}^K$ be a K-class classification function, $x_0 \in \mathbb{R}^d$ be an input vector and the prediction of x_0 is $c(x_0) = \arg\max_{1 \le i \le K} f_i(x_0)$. A perturbed example of x_0 with distortion $\delta \in \mathbb{R}^d$ and ℓ_p-distortion Δ_p is denoted as x_a if $x_a = x_0 + \delta$ and $\Delta_p = \|\delta\|_p$. An adversarial example x_a is the perturbed example if $c(x_a) \neq c(x_0)$.*

Definition 2 (minimum ℓ_p adversarial distortion). *The minimum ℓ_p distortion of x_0 is denoted as $\Delta_{p,min}$ where x_0 is an input vector of a classifier f. The value $\Delta_{p,min}$ is defined as the smallest Δ_p in all the adversarial examples of x_0.*

Definition 3 (lower bound of minimum distortion). *The value β_L is denoted as a lower bound of $\Delta_{p,min}$ where $\beta_L \le \Delta_{p,min}$. Any perturbed examples of x_0 with $\|\delta\|_p \le \beta_L$ are not adversarial examples.*

Lemma 1 (Lipschitz continuity and its relationship with gradient norm (Paulavicius and Zilinskas, 2006) [20]). *Let $S \subset \mathbb{R}^d$ be a convex bounded closed set and define a continuously differentiable function on an open set containing S as $d(m) : S \to \mathbb{R}$. Then, $d(m)$ is a Lipschitz function with Lipschitz constant L_q if the following inequality holds for any $m, n \in S$:*

$$|d(m) - d(n)| \le L_q \|m - n\|_p \tag{2.7}$$

where $L_q = \max \|\nabla h(m)\|_q : m \in S$, $\nabla h(m) = (\frac{\partial h(m)}{\partial m_1}, \ldots, \frac{\partial h(m)}{\partial m_d})^\top$ is the gradient of $h(m)$, and $\frac{1}{p} + \frac{1}{q} = 1$, $1 \le p, q \le \infty$.

Based on Lemma 1, a theoretical robustness guarantee on the lower bound of minimum distortion β_L for untargeted attack is given below.

Theorem 1 (Formal guarantee on β_L for untargeted attack (Weng et al., 2018) [10]). *Denoted L^j_{q,x_0} as the local Lipschitz constant of function $f_c(x) - f_j(x)$ at x_0 over a fixed ball $B_p(x_0, R) := \{x \in \mathbb{R}_d \mid \|x - x_0\|_p \le R\}$. The formal guarantee on β_L could be defined as:*

$$\|\delta\|_p \le \min \{\min_{j \neq c} \frac{f_c(x_0) - f_j(x_0)}{L^j_{q,x_0}},\ R\} \tag{2.8}$$

2.2.2 CLEVER Score

CLEVER score [10] is a robustness metric based on estimating the local Lipschitz constant of a model via Extreme Value Theory. CLEVER score is an attack-agnostic, computationally achievable for deep neural networks and is supported by the Theorem 1 discussed above. Moreover, CLEVER is the first robustness metric independent with attack that can be applied to any neural network classifier and being widely used as the robustness metric in many adversarial robustness library like ART [21].

The key point in the CLEVER theory is the way of estimating L^j_{q,x_0}. In a total of N_b batches, CLEVER first generates N_s samples of $x^{(i)}$ over a fixed ball $B_p(x_0, R)$ uniformly and independently in each batch. The value $\|\nabla g(x^{(i)}\|_q$ is computed via back-propagation and the maximum values of each batch will be stored in set S. Then, with samples S, CLEVER calculates a maximum likelihood estimation of reverse Weibull distribution parameters, and the estimate of L^j_{q,x_0} is equal to the location estimate $\hat{a}_W$. The theory behind their estimation is Fisher-Tippett-Gnedenko Theorem [22] and reverse Weibull distribution could be defined by following:

$$\text{Reverse Weibull class: } G(y) = \begin{cases} \exp -(\frac{a_W - y}{b_W})^{c_W}, & \text{if } y < a_W \\ 1, & \text{if } y \geq a_W \end{cases} \tag{2.9}$$

where G is a non-degenerate distribution function, (a_n, b_n) is a sequence of pairs of real numbers and a_W is a finite right end-point.

Although CLEVER is supported by a rich statistic theory and performs well in practice, ***it may fail to provide a guaranteed lower bound because CLEVER measures the local Lipschitz constant according to statistic.*** Gradient masking [18] is one of strong evidence proved CLEVER overestimating the adversarial perturbation size. We will also demonstrate through interval methods that CLEVER score is not certified.

2.3 Interval Arithmetic

2.3.1 Basic Operations for Interval Arithmetic

Interval arithmetic defines a set of operations at intervals in the same way as classical arithmetic works on real numbers. Let $[x_1, x_2]$, $[y_1, y_2]$ are denoted to interval numbers. The base interval arithmetic operations are as follows [23]:

1. **Addition**: $[x_1, x_2] + [y_1, y_2] = [x_1 + y_1, x_2 + y_2]$
2. **Subtraction**: $[x_1, x_2] - [y_1, y_2] = [x_1 - y_2, x_2 - y_1]$
3. **Multiplication**: $[x_1, x_2] * [y_1, y_2] = [\min\{x_1y_1, x_1y_2, x_2y_1, x_2y_2, \}, \max\{x_1y_1, x_1y_2, x_2y_1, x_2y_2, \}]$
4. **Division**: $\frac{[x_1,x_2]}{[y_1,y_2]} = [x_1, x_2] \cdot \frac{1}{[y_1,y_2]}$

   where

   $\frac{1}{[y_1,y_2]} = [\frac{1}{y_2}, \frac{1}{y_1}]$, if $0 \notin [y_1, y_2]$

   $\frac{1}{[y_1,y_2]} = [-\infty, \frac{1}{y_1}] \bigcup [\frac{1}{y_2}, \infty] \subseteq [-\infty, \infty]$, if $0 \in [y_1, y_2]$

2.3.2 Basic Facts

1. **Interval methods give guaranteed results.** Interval methods take into account the main sources of floating-point errors: round-off and truncation errors, and can also include modelling errors [23]. Hence, the results computed by interval methods are guaranteed.
2. **Interval methods are slow compared with machine level floating-point computations.** Interval methods are solid enough to provide rigorous mathematical proof, but there is a price for rigour [24]. Interval arithmetic can be sluggish, in particular, and often provides overly poor results for real-world computation [25].
3. **Interval computations have the wrapping-effect.**Wrapping-effect [23] is one of main flaw in interval computations which will cause overestimate. For instance, if a value $x \in [0, 1]$, then $x - x = 0$. But with interval arithmetic, $x - x = [-1, 1]$.

Chapter 3

Methodology

3.1 Interval Methodology

It is noticed that the guaranteed results of L_{q,x_0} are possible to certify by using interval method with the combination of bisection, but are not possible in classical methods. In order to get the guaranteed lower bound for local Lipschitz constant in a given radius, interval arithmetic is introduced to this project.

3.1.1 Estimating L_{q,x_0} via Interval Arithmetic

To estimate L_{q,x_0} by interval arithmetic, the input vector x_0 will be converted to an interval box set B with a given range. Then, the algorithm will keep repeating following steps until three stop conditions met. In the loop, the first step is dropping useless intervals which upper bound is lower than one of any others' lower bound. Next, find the maximum interval in the interval boxes set B and the do the bisection to the element in the B. After, the program meets one of the stop conditions. The maximum interval value in B is the maximum norm of the local Lipschitz constant L_{q,x_0}. The formal Algorithm 1 is described in Section 4.1.

3.1.2 Guaranteed Result Given by Interval Methodology

The critical step when estimating L_{q,x_0} by interval arithmetic described above is the way to compute $\|g(x_0)\|_q$ via symbolic differentiation in back-propagation. Symbolic and interval validated techniques are natural allies. In this project, symbolic differentiation is used to compute exact value of L^j_{q,x_0} in back-propagation. However, because of floating-point inaccuracies, the exact answers are not available. Thus, the symbolic environment must

resort to numerical techniques and interval methods compute bounds that contain mathematically correct results. With the combination of interval methodology and symbolic differentiation, a guaranteed result of L^j_{q,x_0} could be estimated.

3.1.3 Multi-dimensions Bisection

Bisection method plays an important role in the proposed interval algorithm which aims to get optimal L_{q,x_0}. In n dimension, the input vector K could be refined into 2^n congruent boxes $Q = \{K^1, K^2, ..., K^{2n}\}$ [26]. The intuitions behind it is shown in Figure 3.1 below. A loop bisection will be used in this project in order to obtain an optimum L_{q,x_0}. In each

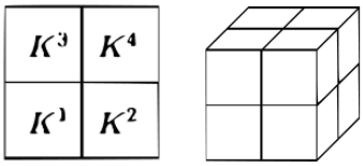

Figure 3.1: Intuitions behind bisection. Left is 4-boxes in $\mathbb{R}^2$; Right is 8-boxes in $\mathbb{R}^3$

iteration, the interval values of every boxes will be calculated, and the boxes which upper bound is less than one of rest boxes' lower bound will be dropped. The iteration will stop when one of the following conditions is met:

1. Reach the minimum width between upper and lower bounds of interval L_{q,x_0}.
2. Reach the maximum of total boxes.
3. Reach maximum iteration number.

3.2 Project Methodologies

This section presents two methodologies I used in the project: methodology for development and methodology for version control.

3.2.1 Development Methodology

The Agile-Waterfall hybrid methodology is used to structure the project. Planning, design, and requirements definition will be completed with Waterfall. However, as this

project is research-driven, most of the following tasks rely on the outcomes of the previous tasks. Therefore, each task will be developed and tested in short sprints by using Agile.

3.2.2 Version Control Methodology

This project incorporates Git as the methodology for version control. Git is the version control system that is most widely used and will trace any change made in this project. By using git, a log will be generated that the project can revert to specific past version if there has any mistake. Moreover, git could also provide evidence for revision in future.

Chapter 4

Design

The design of this project is presented in this chapter which consists of two aspects. The first is how interval arithmetic is used to measure the robustness of neural networks by estimating the local Lipschitz constant. Then, the prototype of automatic measuring system based on interval arithmetic will be introduced.

4.1 An Interval Based Algorithm to Estimate L_{q,x_0}

An interval based algorithm used to estimate the maximum norm of the local Lipschitz constant $\hat{L}_{q,x_0}$ is given below. This algorithm mainly requires an interval box $b \in \mathbb{R}^d$ which converted from an instance $x \in \mathbb{R}^d$ and a neural network $f : \mathbb{R}^d \to \mathbb{R}$. All interval methodologies used in this algorithm has been described in Section 3.1.

Algorithm 1: Estimating L_{q,x_0} by interval arithmetic

Input: a neural network: $f : \mathbb{R}^d \to \mathbb{R}$, an interval box: $b \in \mathbb{R}^d$, maximum number of iteration: $MaxIter$, maximum number of boxes: $MaxBoxes$, minimum width of interval values: $MinWidth$ and the symbolic differentiate function of f : df and $\hat{df} \sqsubseteq df$

Output: the maximum norm of the local Lipschitz constant $\hat{L}_{q,x_0}$

```
1  B ← {b}, i ← 0, nB ← 1 // nB is the number of boxes in the set B
2  do
3  |  i++
4  |  dfB    ← df̂(B)
   |  // abandon useless intervals in B
5  |  B̂      ← the boxes in B which the upper bound of dfB are larger than all
   |    elements' lower bound
6  |  max_dfB ← the maximum interval in B̂
7  |  nB     ← the number of boxes in set B̂
8  |  mW     ← the max width of intervals in max_dfB
9  |  B      ← bisection(B̂)
10 while i < MaxIter && nB < MaxBoxes && mW > MinWidth;
11 L̂q, x0 ← max_dfB
```

4.2 The Prototype of Automatic Measuring System

Based on the interval based algorithm described above, an automatic robustness measuring system designed in this section. This system can automatically give the maximum norm of interval local Lipschitz constant for the input model which currently restricted to one hidden layer neural network. Figure 4.1 below shows a intuitive prototype of this system.

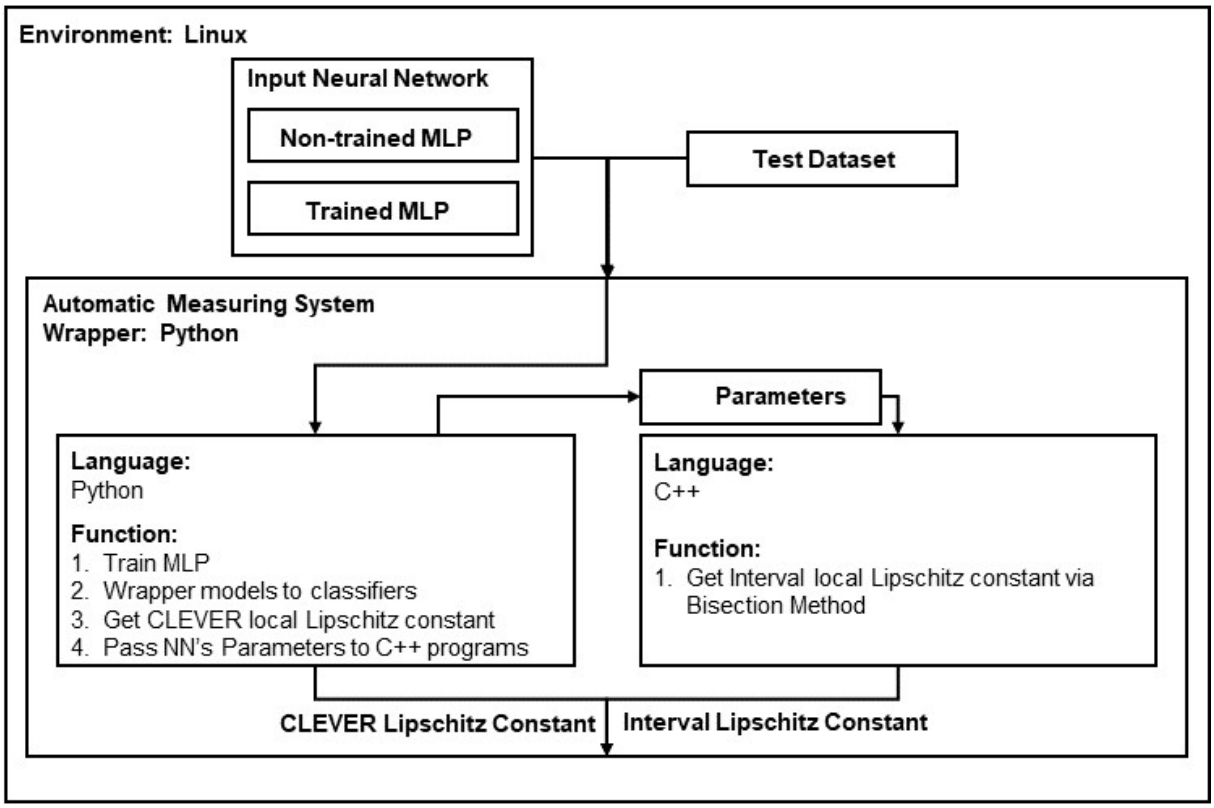

Figure 4.1: The Prototype of Automatic Measuring System

Given a neural network and a test dataset, this system can according to specific mission give required measurement results. In addition, as the reference metric, CLEVER is also introduced into this system. The implementation specifics will be addressed in the next chapter.

Chapter 5

Implementations

The implementation steps of this project are specified in this chapter. It is divided into three parts. The choices of programming languages, libraries and the running environment will be discussed first. After that an interval method for measuring the maximum norm of L_{q,x_0} will be presented. Finally, the preliminary results will be outlined and discussed.

5.1 Languages, Libraries and Environment

This section will discuss the programming languages, libraries and system environment used in this project in detail. These can also be seen in Figure 4.1 on the previous page.

Due to the distinct mission requirements, two programming languages will be used in this project: C++ and Python. C++ is a well structured and written language with high running speed. This project is mostly based on the interval method, but this method has a significant flaw: it is very difficult for calculation, which leads to a slow running speed. Compared to other languages like Python, the programming coded in C++ has a higher running speed which is more suitable for the system based on interval methods. In addition, C++ has several powerful libraries that are also well written and can accommodate high-precision calculations. Python will be used as a wrapper language in this project to string key components together in the system. In addition, Python is also used to calculate CLEVER score and train neural networks because it has several strong frameworks like Pytorch to support machine learning tasks.

The main library used to support interval calculation is MPFI [27] (Multiple Precision Floating-point Interval library). At the same time, Boost Library offers a user-friendly wrapper for MPFI in C++ that allows the easy use of MPFI in this project. For the

machine learning tasks, the Pytorch framework has been chosen because it has powerful functions which is easy to program and can be customised for each model's parameters. Finally, Linux was chosen as the operating environment of this project in order to be compatible with the tools and libraries required by each task.

5.2 Implementation of Automatic Measuring System

The main procedure was described in Section 3.1.1 and measuring Algorithm 1 was given in Section 4.1. This section will give some implementation details. First one is to combine Python and C++. As Python is the wrapper language, the C++ code will be compiled into external Python library via pybind11 Library. Thus, the interval L_{q,x_0} which should be calculated in C++ could be directly called in Python. Another detail of the automatic measuring system is user can customize their requirements. Users can choose the testing dataset and whether the interval L_{q,x_0} will be compared with CLEVER L_{q,x_0}.

5.3 Preliminary Results

Preliminary experiments are performed on the IRIS dataset, which has four input attributes and three output classifications. In the experiments, 40 input vectors are randomly picked from the IRIS dataset and used to calculate the local Lipschitz constant L_{q,x_0} in both CLEVER and interval methods. The results shown in Table 5.1, clearly demonstrate that the CLEVER approach fails to certify a guaranteed lower bound. If the maximum norm of the local Lipschitz constant is correctly determined by CLEVER, the value L_{q,x_0} estimated by CLEVER should in the range of $\hat{L}_{q,x_0}$ which computed by interval method. However, in the experiments result, after less than five times bisection, the value L_{q,x_0} estimated by CLEVER is lower than the lower bound of interval-based L_{q,x_0} in all selected samples.

Preliminary results provide a clear evidence that the direction of this project is right. More relevant works will be conducted in the rest of the project.

Table 5.1: Preliminary Results

Index	Input Vector x_0	CLEVER L_{q,x_0}	Interval L_{q,x_0}	Bisection Number
B01	[6.7, 3.3, 5.7, 2.1]	12.0288	[12.0381, 12.0679]	3
B02	[4.4, 2.9, 1.4, 0.2]	3.733917	[3.73652, 3.74262]	3
B03	[7.2, 3.6, 6.1, 2.5]	9.195505	[9.19735, 9.21089]	5
B04	[7. , 3.2, 4.7, 1.4]	13.6207	[13.6243, 13.6646]	4
B05	[5.4, 3.4, 1.7, 0.2]	2.095303	[2.09721, 2.10014]	3
B06	[5.5, 4.2, 1.4, 0.2]	1.020723	[1.02118, 1.02203]	4
B07	[5. , 3.5, 1.3, 0.3]	1.463961	[1.46441, 1.46645]	3
B08	[6.4, 2.9, 4.3, 1.3]	13.64134	[13.6418, 13.6952]	3
B09	[5.4, 3.7, 1.5, 0.2]	1.058541	[1.05931, 1.06071]	3
B10	[4.6, 3.4, 1.4, 0.3]	2.541125	[2.54136, 2.54774]	2
B11	[6.8, 3. , 5.5, 2.1]	13.47618	[13.4803, 13.5135]	3
B12	[6.5, 3. , 5.2, 2.]	13.68451	[13.686, 13.7295]	3
B13	[7.6, 3. , 6.6, 2.1]	7.334187	[7.33738, 7.35474]	3
B14	[5.5, 2.4, 3.7, 1.]	12.20307	[12.2046, 12.2519]	3
B15	[4.9, 3.1, 1.5, 0.1]	2.916889	[2.91838, 2.92295]	3
B16	[6.4, 2.8, 5.6, 2.1]	6.35967	[6.36138, 6.373]	4
B17	[6.5, 3. , 5.2, 2.]	11.82055	[11.8251, 11.8568]	4
B18	[6.8, 3.2, 5.9, 2.3]	6.678896	[6.6842, 6.69774]	3
B19	[6.3, 2.5, 5. , 1.9]	11.33109	[11.3324, 11.3606]	3
B20	[5.1, 3.5, 1.4, 0.3]	2.901821	[2.90234, 2.9065]	3
B21	[6.3, 3.4, 5.6, 2.4]	8.122278	[8.12711, 8.14505]	3
B22	[6.3, 2.9, 5.6, 1.8]	11.29898	[11.3013, 11.33]	3
B23	[5.6, 2.9, 3.6, 1.3]	13.5021	[13.5072, 13.5552]	3
B24	[6. , 2.2, 5. , 1.5]	11.36601	[11.3696, 11.3962]	4
B25	[7.4, 2.8, 6.1, 1.9]	12.77108	[12.776, 12.8081]	3
B26	[6.3, 2.9, 5.6, 1.8]	12.73948	[12.7432, 12.7691]	4
B27	[5.8, 2.6, 4. , 1.2]	14.10678	[14.1073, 14.1571]	4
B28	[6.6, 3. , 4.4, 1.4]	13.26788	[13.269, 13.3096]	4
B29	[5.1, 2.5, 3. , 1.1]	10.99253	[10.9969, 11.0326]	4
B30	[4.9, 3.1, 1.5, 0.2]	2.747983	[2.74862, 2.75543]	2
B31	[5.7, 2.6, 3.5, 1.]	9.803847	[9.81069, 9.84816]	3
B32	[5. , 3.3, 1.4, 0.2]	3.10479	[3.105, 3.11164]	2
B33	[5.8, 2.8, 5.1, 2.4]	5.865237	[5.86926, 5.88481]	2
B34	[5.5, 4.2, 1.4, 0.2]	1.072406	[1.07241, 1.0747]	2
B35	[5.4, 3.4, 1.5, 0.4]	2.755713	[2.75716, 2.76312]	3
B36	[5. , 2.3, 3.3, 1.]	12.04789	[12.0508, 12.0969]	3
B37	[5.8, 2.6, 4. , 1.2]	11.39727	[11.4002, 11.443]	3
B38	[7.6, 3. , 6.6, 2.1]	5.606251	[5.60715, 5.61429]	4
B39	[4.4, 3. , 1.3, 0.2]	3.659902	[3.66149, 3.66531]	4
B40	[4.7, 3.2, 1.6, 0.2]	2.404177	[2.40482, 2.41004]	3

Chapter 6

Progress

This chapter discusses the progress made so far in the project as well as a critical assessment of that progress.

6.1 Project Management

This project is to start by investigating and summarizing recent research, addressing two aspects: adversarial examples and robustness analysis. The project would then concentrate on evaluating robustness in MLPs. A novel method based on interval arithmetic is designed. Next, the MLPs that approximate predefined maps (e.g. from R^n to R^m) will be trained and the proposed method and the methods from the literature will be applied to MLPs for comparing their performance. All the tasks described above have been completed ahead of time, even though some of them are planned for the next semester. In addition, I have enhanced the requirements of some tasks. For example, according to the original plan, the training MLPs is approximate predefined maps, but now they can be obtained by training based on the chosen datasets.

The Gannt chart below (Figure 6.1) describes the original project plan and Figure 6.2 is the updated timeline. The main reason for plan updated is this project began in summer vacation and lots of related literature has been reviewed and analysed before this semester. With the support of those methodologies, this project is progressing smoothly. Thus, this project will add two objects: one is (L.) Using estimated interval local Lipschitz constant to evaluate the robustness of classifiers and another one is (M.) Isolating interval boxes which give the maximum norm of the local Lipschitz constant. which is presented as blue block in the Figure 6.2.

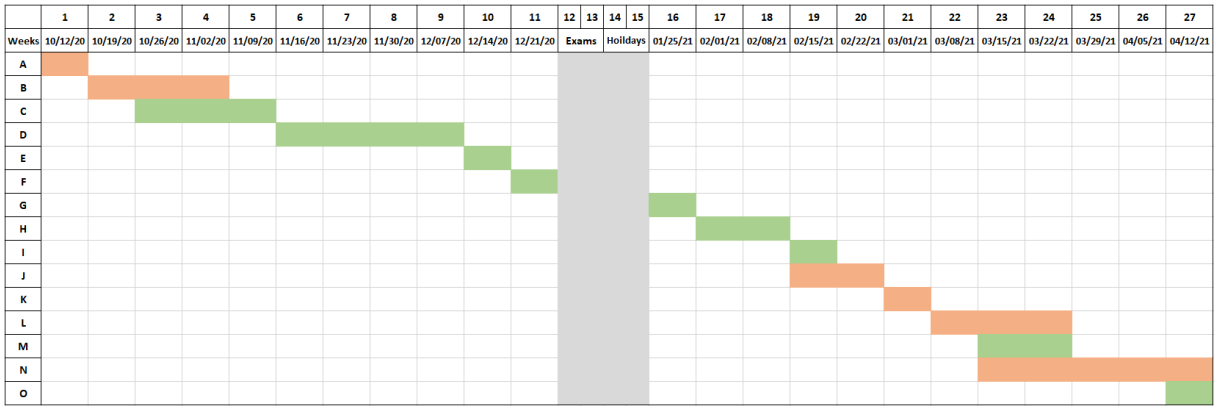

Figure 6.1: Original Timeline

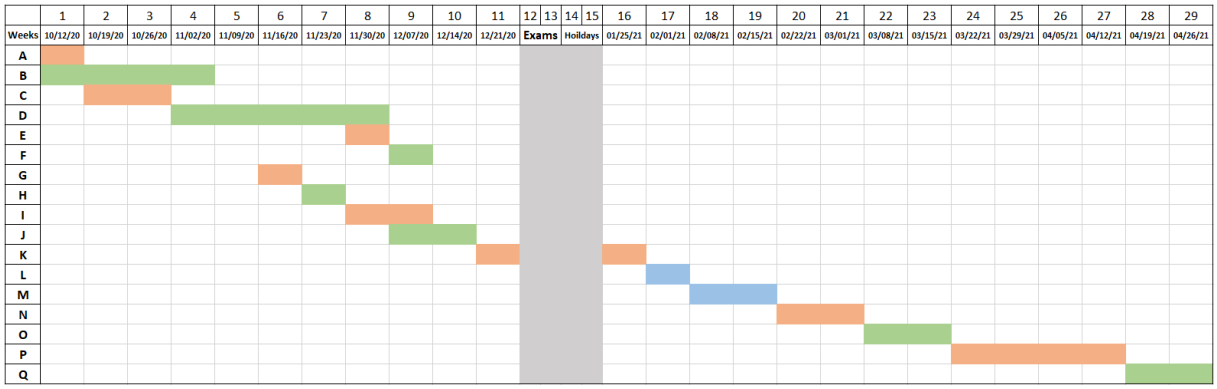

Figure 6.2: Updated Timeline

The updated tasks of this project are:

A. Preparing and submitting the proposal and ethics form.

B. Gathering necessary literature.

C. Implementing the established methods and summarizing their drawbacks and advantages.

D. Proposing a novel method for robustness analysis based on interval arithmetic.

E. Writing the interim report.

F. Modifying and submitting the interim report.

G. Training MLPs (Multilayer Perceptrons)

H. Applying proposed the method and the methods from literature.

I. Evaluating the performance of the proposed method.

J. Comparing the performances between the proposed method and the established methods.

K. Applying the interval method for analysing robustness not just with regard to the perturbations of the input, but also the model parameters, e.g., weights and biases of the network.

L. Using estimated interval local Lipschitz constant to evaluate the robustness of classifiers

M. Isolating interval boxes which give the maximum norm of the local Lipschitz constant.

N. Allowing time for any major changes that may need to take place.

O. Concluding the project and provide outcome analysis.

P. Writing the final dissertation.

Q. Modifying and submitting the final report.

6.2 Reflections

The speed of the project development is satisfactory and deadlines are strictly maintained by all deliverable submitted on time. All the tasks planned for this semester have been achieved and the requirements of some tasks have enhanced. Due to good progress in this semester, additional aims and objectives will be planned for next semester.

6.2.1 Risk Management

However, there still exist some risks in this project that may lead issues. In this section, four main risks are outlined in the following table and the mitigated methods are provided to decrease the risks.

Table 6.1: Risks and Mitigated Methods

Risks	Mitigated and Managed Methods
Made some mistakes that cannot be seen easily and clearly while programming codes	Taking Unit Test
There is still some uncertainty while applying interval arithmetics	1. Reviewing more related field research literature.
	2. Discussing the puzzle with supervisor.
Some difficult tasks take more time than I planned	1. Maintaining the some extra time for buffering.
	2. Be more cautious when designing plan.
The time left for writing the report is too short	Arrange the time and start writing the report early

6.2.2 Future Plan

In the next semester, maintaining the same speed of progress is expected while paying more attention to avoiding the risks mentioned above. In addition to completing the originally planned tasks in the next semester, I hope I can also complete the new added tasks with high quality, especially the following two:

1. Evaluating the robustness of classification via interval arithmetic.

   Currently, the value L_{q,x_0} could be estimated with the combination of interval arithmetic and bisection method. However, could the robustness of classification be computed by the same equation provided by Weng et.al. (2018) [10] needs more discussion.

2. Isolating interval boxes which give the maximum norm of the local Lipschitz constant.

   Interval boxes that have the maximum norm of the local Lipschitz constant can be the most influential points for the robustness of the neural network.the Isolating those point could help us better understand the factors which could trigger the lack of robustness.

References

[1] D. Heaven, "Why deep-learning ais are so easy to fool," *Nature*, vol. 574, no. 7777, pp. 163–166, 2019.

[2] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, "Pcanet: A simple deep learning baseline for image classification?" *IEEE transactions on image processing*, vol. 24, no. 12, pp. 5017–5032, 2015.

[3] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[4] A. Mathis, P. Mamidanna, K. M. Cury, T. Abe, V. N. Murthy, M. W. Mathis, and M. Bethge, "Deeplabcut: markerless pose estimation of user-defined body parts with deep learning," *Nature neuroscience*, vol. 21, no. 9, pp. 1281–1289, 2018.

[5] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 427–436.

[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

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

[8] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[9] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019.

[10] T.-W. Weng, H. Zhang, P.-Y. Chen, J. Yi, D. Su, Y. Gao, C.-J. Hsieh, and L. Daniel, "Evaluating the robustness of neural networks: An extreme value theory approach," *arXiv preprint arXiv:1801.10578*, 2018.

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

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

[13] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[14] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE transactions on evolutionary computation*, vol. 15, no. 1, pp. 4–31, 2010.

[15] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*. IEEE, 2017, pp. 39–57.

[16] K. Ren, T. Zheng, Z. Qin, and X. Liu, "Adversarial attacks and defenses in deep learning," *Engineering*, 2020.

[17] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2574–2582.

[18] I. Goodfellow, "Gradient masking causes clever to overestimate adversarial perturbation size," *arXiv preprint arXiv:1804.07870*, 2018.

[19] M. Hein and M. Andriushchenko, "Formal guarantees on the robustness of a classifier against adversarial manipulation," in *Advances in Neural Information Processing Systems*, 2017, pp. 2266–2276.

[20] R. Paulavičius and J. Žilinskas, "Analysis of different norms and corresponding lipschitz constants for global optimization in multidimensional case," *Information Technology and Control*, vol. 36, no. 4, 2007.

[21] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, I. Molloy, and B. Edwards, "Adversarial robustness toolbox v1.2.0," *CoRR*, vol. 1807.01069, 2018. [Online]. Available: https://arxiv.org/pdf/1807.01069

[22] M. L. Bianchi, S. V. Stoyanov, G. L. Tassinari, F. J. Fabozzi, S. M. Focardi *et al.*, "Extreme value theory," *World Scientific Book Chapters*, pp. 367–430, 2019.

[23] R. E. Moore, R. B. Kearfott, and M. J. Cloud, *Introduction to interval analysis*. SIAM, 2009.

[24] M. W. Gutowski, "Power and beauty of interval methods," *arXiv preprint physics/0302034*, 2003.

[25] W. Tucker, *Validated numerics: a short introduction to rigorous computations*. Princeton University Press, 2011.

[26] M. L. Galván, "The multivariate bisection algorithm," *arXiv preprint arXiv:1702.05542*, 2017.

[27] N. Revol and F. Rouillier, "Mpfi: a library for arbitrary precision interval arithmetic," 2002.