**Title**
Towards Theoretical Analysis and Empirical Improvement of Certied Robust Training



**Author**
wang, Yihan

**Publication Date**
2022

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

# Towards Theoretical Analysis and Empirical Improvement of Certified Robust Training

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Computer Science

by

Yihan Wang

2022

ABSTRACT OF THE THESIS

Towards Theoretical Analysis
and Empirical Improvement of Certified Robust Training

by

Yihan Wang
Master of Science in Computer Science
University of California, Los Angeles, 2022
Professor Cho-Jui Hsieh, Chair

Recently, bound propagation based certified robust training methods have been proposed for training neural networks with certifiable robustness guarantees. Despite that state-of-the-art (SOTA) methods including interval bound propagation (IBP) and CROWN-IBP have succeeded in providing certified robustness with efficient per-batch training complexity, there are several challenges faced by these certified robust training methods. First, they usually use a long warmup schedule with hundreds or thousands epochs to increase the perturbation radius for SOTA performance and are thus still costly. Second, the convergence of IBP training remains unknown. In this paper, we identify two important issues related to slow warmup schedule for IBP training, namely exploded bounds at initialization, and the imbalance in ReLU activation states. These two issues make certified training difficult and unstable, and thereby long warmup schedules were needed in prior works. We proposed improvements to mitigate these issues and we are able to obtain **65.03%** verified error on CIFAR-10 ($\epsilon = \frac{8}{255}$) using very short training schedules. For the convergence problem, we show that for a randomly initialized two-layer ReLU neural network with logistic loss,

with sufficiently small perturbation radius and large network width, gradient descent for IBP training can converge to zero training robust error with a linear convergence rate with a high probability, and at this convergence state the robustness certification by IBP can accurately reflect the true robustness of the network.

The thesis of Yihan Wang is approved.

Baharan Mirzasoleimanbarzi

Quanquan Gu

Cho-Jui Hsieh, Committee Chair

University of California, Los Angeles

2022

*To my parents ...*

*for their support and love*

TABLE OF CONTENTS

**1 Introduction** . . . . . **1**

**2 Interval Bound Propagation and IBP training** . . . . . **3**

2.1 Certified Defense . . . . . 3

2.2 Interval Bound Propagation (IBP) . . . . . 4

2.3 Interval Bound Propagation (IBP) Training . . . . . 4

2.4 Slow Certified Training with Long Warm-up Schedule . . . . . 5

2.5 Difficulty in IBP Convergence . . . . . 5

**3 Fast IBP Training with Shorter Warm-up** . . . . . **7**

3.1 Two Issues in Existing IBP training . . . . . 7

3.2 Proposed Method for Fast IBP Training . . . . . 10

3.2.1 IBP initialization . . . . . 10

3.2.2 Batch Normalization . . . . . 11

3.2.3 Warmup Regularization . . . . . 11

3.2.4 Training Objectives . . . . . 13

3.3 Experiments . . . . . 14

3.3.1 Settings . . . . . 14

**4 Convergence Analysis for IBP Training** . . . . . **17**

4.1 Preliminaries . . . . . 17

4.1.1 Neural Networks . . . . . 17

4.1.2 IBP-based Certified Robust Training . . . . . 18

4.1.3 Gradient Flow and Gram Matrix . . . . . 20
4.2 Convergence Analysis for IBP Training . . . . . 22
4.3 Experiments . . . . . 22

**5 Conclusion** . . . . . **24**

**References** . . . . . **25**

LIST OF FIGURES

3.1 We show that certified bounds explode at initialization, in a simple untrained CNN (the classification layer is omitted) using Xavier initialization. We plot $\log \hat{\mathbb{E}}(\Delta_i)$ for each layer $i$. . . . . 10

3.2 Ratios of active and unstable ReLU neurons for CNN-7 on CIFAR-10 with different settings. The vanilla ones are not regularized, and "vanilla (w/o BN)" does not use BN either. . . . . 10

4.1 Experimental results. . . . . 23

LIST OF TABLES

3.1 List of several weight initialization methods and their *difference gain*. . . . . 9

3.2 Standard and verified error rates (%) of models trained with different methods respectively on MNIST ($\epsilon_{\text{target}} = 0.4$) and CIFAR-10 ($\epsilon_{\text{target}} = 8/255$). Schedule is represented as the total number of epochs and the number of epochs in each of the three phases with $\epsilon = 0$, increasing $\epsilon \in (0, \epsilon_{\text{target}})$ and final $\epsilon = \epsilon_{\text{target}}$ respectively. We report the mean and standard deviation of the results on 5 repeats for CNN-7 and 3 repeats for Wide-ResNet and ResNeXt respectively. All models include BN after every layer (see Sec. 3.2.2). We also report the best run in "Ours (best)" since main results in prior works did not have repeats. Literature results with the "†" mark are concurrent works. . . . . 15

# ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor: Cho-Jui Hsieh, who led me to the area of machine learning robustness and guided me patiently and thoughtfully in my research and study. I am also grateful to my coauthors Zhouxing Shi and Huan Zhang, who have closely worked with me on many projects with patience and friendship. Also, I must sincerely thank Jinfeng Yi, who offered me the internship opportunity at JD.com during the COVID-19 pandemic where I worked on the Fast IBP training project; and Quanquan Gu for his class on Deep Learning theory and his valuable support on the IBP convergence project. Finally, I would like to give my deepest thanks to my parents, without whose support I would not have the opportunity to study at UCLA.

# CHAPTER 1

# Introduction

As the development of machine learning algorithms, machine learning models have been successfully applied in many safety-critical areas like autonomous driving. And the safety of machine learning models has attracted lots of attention. One critical problem of machine learning is the adversarial examples [SZS13, GSS15, CW17, KGB16, CZC17, MMS18, SZC18, CZK19], which can fool machine learning models with imperceptible perturbations that do not change the latent content. The robustness can be evaluated empirically by adversarial attack. And many methods are also proposed to improve the empirical robustness of models, such as adversarial training [MMS18], which augments the training with adversarial examples. However, the empirical robustness by adversarial training is not certified, and can be attacked by stronger attacks [CW17]. Therefore, some recent works propose to evaluate the robustness by verification, which verify whether the model is safe within a given range by verifying the worst case in the range [KBD17, ZWC18, WK18, SGM18, SGP19, BTT17, RSL18, WPW18, XSZ20, WZX21]. Based on the evaluation, certified robust training is proposed to improve the certified robustness by minimizing the upper bound of a certified loss within the given perturbation range. Among the certified training methods, Interval Bound Propagation (IBP)[GDS18, MGV18] and CROWN-IBP [ZCX20, XSZ20] can produce the certified bounds the most efficiently, while CROWN [ZWC18] can produce tighter bounds with a higher computational complexity. However, there are still several challenges remaining for certified training. First, certified training needs a long warm-up schedule to gradually increase the perturbation radius $\epsilon$ from zero to the maximum radius in the beginning of the training. Otherwise, the training can be very easy to be trapped into local minima. Second,

certified training suffers significant underfitting problem instead of overfitting. Even the training certified error is difficult to decrease with a sufficiently large network width. This indicates that there might be some essential difference between certified training and clean training. As a simplified case, we focus on IBP training in our discussion, which is also potentially able to be extended to other certified robust training methods.

For the slow warm-up schedule of certified training, we identified two issues in IBP training. The first is the exploded bounds problems. The certified bounds of intermediate layers can explode when passed through the layers, which leads to very loose bounds at the last layer and can harm the training performance. To address this, we propose to use a special initialization for IBP training and an explicit regularization to stabilize the certified bounds at initialization. The second is the imbalanced ReLU status problem. The relaxation for ReLU neurons in IBP training encourages the ReLU neurons to be inactive for tight bounds, which can harm the capacity of network and harm the certified robustness. We proposed to add BatchNormalization layers after each convolutional layer to mitigate the imbalance problem and also use an explicit regularizer to control the ReLU imbalance pattern. With these improvements, we are able to achieve state-of-the-art certified robust accuracy in 160 epochs in total compared to thousands or hundreds of training epochs in previous work.

For the difficulty in IBP training, we give the first convergence analysis of IBP training. We show that for a randomly initialized two-layer ReLU neural network with logistic loss, with sufficiently small perturbation radius and large network width, gradient descent for IBP training can converge to zero training robust error with a linear convergence rate with a high probability, and at this convergence state the robustness certification by IBP can accurately reflect the true robustness of the network. On the other hand, if the perturbation radius is relatively large, the convergence is not guaranteed even with large network width, which is essentially different from standard training.

# CHAPTER 2

# Interval Bound Propagation and IBP training

## 2.1 Certified Defense

Robust training can be formulated as solving a min-max optimization problem:

$$\min_{\theta} \mathbb{E}_{(\mathbf{x},y)\in\mathcal{X}} \left[ \max_{\delta\in\Delta(\mathbf{x})} L(f_\theta(\mathbf{x}+\delta), y) \right], \tag{2.1}$$

where $f_\theta$ stands for a neural network parameterized by $\theta$, $\mathcal{X}$ is the data distribution, $\mathbf{x}$ is a data example, $y$ is its ground-truth label, $\delta$ is a perturbation constrained by specification $\Delta(\mathbf{x})$, and $L$ is the loss function. Empirical robust training like adversarial training [GSS15, MMS18] solves the inner-max of 2.1 by adversarial attack. While adversarial training can obtain effective empirical robustness, it is not certifiable robust and can be attacked by stronger attack methods [CW17]. In contrast, certified robust training methods compute a certified upper bound for for the inner maximization, which can give a provable guarantee on the robustness after training.

For certified robust training, some works applies semidefinite relaxation for small networks, and another line of works use linear relaxation based bound propagation [GDS18, MGV18, ZCX20, XSZ20, ZWC18]. Among the linear relaxation based methods, Interval Bound Propagation (IBP) [GDS18, MGV18] is one of the most efficient methods that propagate the upper and lower bounds in the forward pass.

## 2.2 Interval Bound Propagation (IBP)

We consider a network $f_\theta$ with $L$ layers, indexed by $i = 1, 2, ..., L$. We use $\mathbf{h}_i$ to denote the pre-activation output value of the $i$-th layer, and $\mathbf{h}_{i,j}$ denotes the $j$-th neuron in the $i$-th layer. We also use $\mathbf{z}_i = \text{ReLU}(\mathbf{h}_i)$ to denote the post-activation value. For a convolutional or fully-connected layer, we use $\mathbf{W}_i$ and $\mathbf{b}_i$ to denote its parameters, where $\mathbf{W}_i \in \mathbb{R}^{r_i \times n_i}, \mathbf{b} \in \mathbb{R}^{r_i}$, and $r_i$ and $n_i$ are called the "fan-out" and "fan-in" number of the layer respectively [HZR15]. This is straightforward for a fully-connected layer, and for a convolutional layer with kernel size $k$, $c_{\text{in}}$ input channels and $c_{\text{out}}$ output channels, we can still view the convolution as an affine transformation with $n_i = k^2 c_{\text{in}}$ and $r_i = c_{\text{out}}$. In particular, we use $\mathbf{h}_0 = \mathbf{x} + \delta$ to denote the input layer perturbed by $\delta$ ($\mathbf{z}_0$ is not applicable).

In IBP [MGV18, GDS18], it computes and propagates lower and upper bound intervals layer by layer until the last layer.

For pre-activation $\mathbf{h}_i$, its interval bounds can be denoted as $[\underline{\mathbf{h}}_i, \overline{\mathbf{h}}_i]$, where $\underline{\mathbf{h}}_i \leq \mathbf{h}_i \leq \overline{\mathbf{h}}_i$ ($\forall \|\delta\|_\infty \leq \epsilon$). Similarly, there are also post-activation interval bounds $[\underline{\mathbf{z}}_i, \overline{\mathbf{z}}_i]$. For an affine layer with weight $\mathbf{W}_i$, we can formulate the bound propagation as

$$\underline{\mathbf{h}}_i = \mathbf{W}_{i,+}\underline{\mathbf{z}}_{i-1} + \mathbf{W}_{i,-}\overline{\mathbf{z}}_{i-1} + \mathbf{b}_i, \;\; \overline{\mathbf{h}}_i = \mathbf{W}_{i,+}\overline{\mathbf{z}}_{i-1} + \mathbf{W}_{i,-}\underline{\mathbf{z}}_{i-1} + \mathbf{b}_i, \tag{2.2}$$

## 2.3 Interval Bound Propagation (IBP) Training

In IBP training, we can extend the network with a loss operation, for which we can also get its interval bounds by propagating from its inputs. And for Loss $L$, we can get its upper and lower bounds within the perturbation set $\delta \in \Delta(x)$ as $\underline{L} \leq L \leq \overline{L}$. Then in certified training, we can relax 2.1 as

$$\min_\theta \mathbb{E}_{(\mathbf{x},y)\in\mathcal{X}} \left[\overline{L}(f_\theta(\mathbf{x}), y)\right], \tag{2.3}$$

## 2.4 Slow Certified Training with Long Warm-up Schedule

Certified robust training including IBP and CROWN-IBP training remains costly and challenging, mainly due to their unstable training behavior – they could easily diverge or stuck at a degenerate solution without a long "warmup" schedule. The warmup schedule here refers to training the model with a regular (non-robust) loss first and then gradually increasing the perturbation radius from 0 to the target value in the robust loss (some previous works also refer to it as "ramp-up"). For example, generalized CROWN-IBP in [XSZ20] used 900 epochs for warmup and 2,000 epochs in total to train a convolutional model on CIFAR-10 [KH09].

## 2.5 Difficulty in IBP Convergence

Despite being one of the most successful certified defense methods, the convergence properties of IBP training remained unknown. For natural neural network training (training without considering adversarial perturbation, aka standard training), it has been shown that gradient descent for overparameterized networks can provably converge to a global minimizer with random initialization [LL18, DZP18, DLL18, JGH18, ALS18, ZCZ18]. Compared to natural neural network training, IBP-based robust training has a very different training scheme, and thus requires a different convergence analysis. First, in the robust training problem, input data can contain adversarial perturbations, and the training objective is to minimize a robust loss rather than a natural loss. Second, IBP training essentially optimizes an augmented network which contains IBP bound computation rather than standard neural networks, as illustrated in [ZCX20]. Third, in IBP training, the activation state of each neuron depends on the certified bounds rather than the value in natural neural network computation, and this introduces additional perturbation-related terms in the convergence analysis for IBP.

Also, some empirical experiments show that convergence of IBP-based robust training is much more difficult than standard training. And in some settings, this difficulty can not be

addressed by simply enlarging width like standard training.

# CHAPTER 3

# Fast IBP Training with Shorter Warm-up

## 3.1 Two Issues in Existing IBP training

In this section, we analyze the issues in existing IBP training. In particular, we identify two issues, including exploded bounds at initialization, and also the imbalance between ReLU activation states.

#### 3.1.0.1 Exploded Bounds at Initialization

For simplicity, we assume the network has a feedforward architecture in this analysis, but the analysis can also be easily extended to other architectures. For affine layer $\mathbf{h}_i = \mathbf{W}_i\mathbf{z}_{i-1} + \mathbf{b}_i$, the IBP bound computation is as follows:

$$\underline{\mathbf{h}}_i = \mathbf{W}_{i,+}\underline{\mathbf{z}}_{i-1} + \mathbf{W}_{i,-}\overline{\mathbf{z}}_{i-1} + \mathbf{b}_i, \;\; \overline{\mathbf{h}}_i = \mathbf{W}_{i,+}\overline{\mathbf{z}}_{i-1} + \mathbf{W}_{i,-}\underline{\mathbf{z}}_{i-1} + \mathbf{b}_i, \tag{3.1}$$

where $\mathbf{W}_{i,+}$ stands for retaining positive elements in $\mathbf{W}_i$ only while setting other elements to zero, and vice versa for $\mathbf{W}_{i,-}$. $\mathbf{h}_i$ can be viewed as a function with the post-activation value of the previous layer $\mathbf{z}_i$ as input, denoted as $\mathbf{h}_i(\mathbf{z}_i)$. In Eq. (3.1), the IBP bounds guarantee that $\underline{\mathbf{h}}_i \leq \mathbf{h}_i(\mathbf{z}_i) \leq \overline{\mathbf{h}}_i$ $(\forall \underline{\mathbf{z}}_i \leq \mathbf{z}_i \leq \overline{\mathbf{z}}_i)$ for element-wise "$\leq$". We then check the tightness of the interval bounds:

$$\Delta_i = \overline{\mathbf{h}}_i - \underline{\mathbf{h}}_i = |\mathbf{W}_i|(\overline{\mathbf{z}}_{i-1} - \underline{\mathbf{z}}_{i-1}) = |\mathbf{W}_i|\delta_{i-1}, \tag{3.2}$$

where $\Delta_i$ denotes the gap between the upper and lower bounds, which can reflect the tightness of the bounds, and $|\mathbf{W}_i|$ stands for taking the absolute value element-wise. At initialization, we assume that each $\mathbf{W}_i$ independently follows a distribution with zero mean and variance $\sigma_i^2$, and the distribution is symmetric about 0. For a vector or matrix with independent elements following the same distribution, we use $\mathbb{E}(\cdot)$ to denote the expectation of this distribution. We can view each element in vector $\Delta_i$ as a random variable that follows the same distribution, and we denote its expectation as $\mathbb{E}(\Delta_i)$, to measure the expected tightness at layer $i$. As $\mathbf{W}_i$ and $\delta_{i-1}$ are independent, we have $\mathbb{E}(\Delta_i) = n_i\mathbb{E}(|\mathbf{W}_i|)\mathbb{E}(\delta_{i-1})$. Detailed in Appendix **??**, we further have $\mathbb{E}(\delta_i) = \mathbb{E}(\mathrm{ReLU}(\overline{\mathbf{h}}_i) - \mathrm{ReLU}(\underline{\mathbf{h}}_i)) = \frac{1}{2}\mathbb{E}(\Delta_i)$, and

$$\mathbb{E}(\Delta_i) = \frac{n_i}{2}\mathbb{E}(|\mathbf{W}_i|)\mathbb{E}(\Delta_{i-1}). \tag{3.3}$$

Empirically, we can estimate $\mathbb{E}(\Delta_i)$ given a batch of concrete data, by taking the mean, and we use $\hat{\mathbb{E}}(\Delta_i)$ to denote the result of the empirical estimation.

We define a metric to characterize to what extent the certified bounds become looser, after propagating bounds from layer $i-1$ to layer $i$:

**Definition 1.** *We define the difference gain when bounds are propagated from layer $i-1$ to layer $i$:*

$$\mathbb{E}(\Delta_i)/\mathbb{E}(\Delta_{i-1}) = \frac{n_i}{2}\mathbb{E}(|\mathbf{W}_i|). \tag{3.4}$$

*Bounds are considered to be stable if the difference gain $\mathbb{E}(\Delta_i)/\mathbb{E}(\Delta_{i-1})$ is close to 1.*

A large difference gain indicates exploded bounds, but it cannot be much smaller than 1 either to avoid signal vanishing in the model. We find that weight initialization in prior works have large difference gain values especially for layers with larger $n_i$. For example, for the widely used Xavier initialization [GB10], the difference gain is $\frac{1}{4}\sqrt{n_i}$, and it can be as large as 45.25 when $n_i = 32768$ for a fully-connected layer in experiments.

This indicates that certified bounds explode at initialization. We illustrate the bound

explosion in Figure 3.1. And in 3.1, we show difference gains for different initializations. We show both closed form result and empirical values for a 7-layer CNN model with $n_i \in \{27, 576, 1152, 32768\}$ (without BN). The concrete values are obtained by computing the mean of 100 random trials respectively. For orthogonal initialization, obtaining a closed form of difference gain is non-trivial so we omit its closed-form result, but it has large difference gains under empirical measurements.

As a result, long warmup schedules are important in previous works, to gradually tighten certified bounds and ease training, but this is inefficient.

Table 3.1: List of several weight initialization methods and their *difference gain*.

| Method | Adopted by | Difference Gain | | | | |
|---|---|---|---|---|---|---|
| | | Closed form | $n_i = 27$ | $n_i = 576$ | $n_i = 1152$ | $n_i = 32768$ |
| Xavier (uniform) [GB10] | [ZCX20, XSZ20] | $\frac{1}{4}\sqrt{n_i}$ | 1.30 | 6.00 | 8.48 | 45.25 |
| Xavier (Gaussian) [GB10] | - | $\sqrt{\frac{1}{2\pi}}\sqrt{n_i}$ | 2.07 | 9.57 | 13.54 | 72.2 |
| Kaiming (uniform) [HZR15] | - | $\frac{\sqrt{3}}{4}\sqrt{n_i}$ | 3.20 | 14.70 | 20.77 | 110.85 |
| Kaiming (Gaussian) [HZR15] | - | $\sqrt{\frac{1}{\pi}}\sqrt{n_i}$ | 2.93 | 13.54 | 19.15 | 102.13 |
| Orthogonal [SMG13] | [GDS18] | - | 2.09 | 9.58 | 13.54 | 72.22 |
| IBP Initialization | This work | 1 | 1.01 | 1.00 | 1.00 | 1.00 |

#### 3.1.0.2 Imbalanced ReLU Activation States

We show another issue in existing certified training, where the models have a bias towards *inactive ReLU neurons*. Here "inactive ReLU neurons" are defined as neurons with non-positive pre-activation upper bounds ($\overline{\mathbf{h}}_{i,j} \leq 0$), i.e., they are always inactive regardless of input perturbations. Similarly, *active ReLU neurons* have non-negative pre-activation lower bounds ($\underline{\mathbf{h}}_{i,j} \geq 0$). There are also *unstable ReLU neurons* with uncertain activation states given different input perturbations ($\underline{\mathbf{h}}_{i,j} \leq 0 \leq \overline{\mathbf{h}}_{i,j}$). In IBP training, inactive neurons have tighter bounds than active and unstable ones as shown in Figure **??** in Appendix **??**, and thus the optimization tends to push the neurons to be inactive. We show this imbalance ReLU status in Figure 3.2 (vanilla w/o BN), and it is more severe when the warmup is

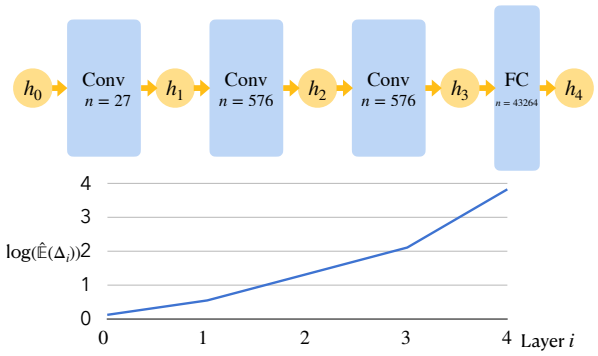



Figure 3.1: We show that certified bounds explode at initialization, in a simple untrained CNN (the classification layer is omitted) using Xavier initialization. We plot $\log \hat{\mathbb{E}}(\Delta_i)$ for each layer $i$.

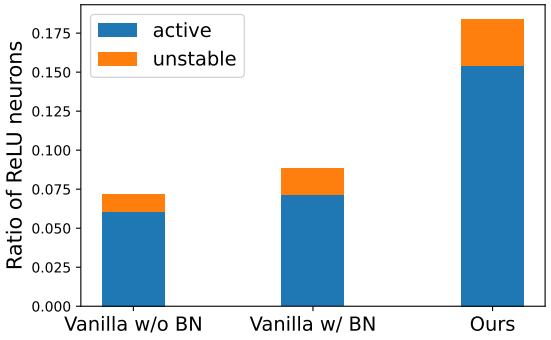



Figure 3.2: Ratios of active and unstable ReLU neurons for CNN-7 on CIFAR-10 with different settings. The vanilla ones are not regularized, and "vanilla (w/o BN)" does not use BN either.

shorter. Too many inactive neurons indicates that many neurons are essentially unused or dead, which will harm the model's capacity and block gradients as discussed by [LSS19] on standard training.

## 3.2 Proposed Method for Fast IBP Training

### 3.2.1 IBP initialization

We propose a new *IBP initialization* for IBP training. Specifically, we independently initialize each element in $\mathbf{W}_i$ following a normal distribution $\mathcal{N}(0, \sigma_i^2)$, and we aim to choose a value for $\sigma_i$ such that the *difference gain* defined in Eq. (3.4) is exactly 1. When elements in $\mathbf{W}_i$ follow the normal distribution, we have $\mathbb{E}(|\mathbf{W}_i|) = \sqrt{2/\pi}\sigma_i$, and thereby we take $\sigma_i = \frac{\sqrt{2\pi}}{n_i}$, which makes the difference gain $\frac{n_i}{2}\mathbb{E}(|\mathbf{W}_i|)$ exactly 1. This initialization can further be calibrated for non-feedforward networks such as ResNet, please refer to [] for details.

### 3.2.2 Batch Normalization

Batch normalization (BN) [IS15] normalizes the input of each layer to a distribution with stable mean and variance. It can improve the optimization for DNN as shown in prior works for standard DNN training [IS15, Van17, STI18]. In addition, for IBP training, BN can normalize the variance of bounds, and it can also improve the balance of ReLU activation states by shifting the center of upper and lower bounds to zero (before the additional linear transformation which comes after the normalization). In prior certified training works [GDS18, ZCX20, XSZ20], they only used BN for some layers in some models but not all layers, and they did not identify the benefit of BN in certified training. We empirically demonstrate that fully adding BN to each affine layer can significantly mitigate the imbalance ReLU issue and improve IBP training. We follow the BN implementation by [WSM18, XSZ20] for certified training, where the shifting and scaling parameters are computed from unperturbed data.

Note that our previous analysis on IBP initialization considers a network without BN. BN which rescales the output of each layer can still affect the tightness of IBP bounds, and the effect of IBP initialization may be weakened. This is a limitation of the proposed initialization which could possibly be improved by considering the effect of BN in future work.

### 3.2.3 Warmup Regularization

To further address the aforementioned two issues in Sec. 3.1, and to explicitly stabilize the tightness of certified bounds and balance ReLU neuron states, we add two regularizers in the warmup stage of IBP training, The regularizers are principled and motivated by the two issues we discover.

#### 3.2.3.1 Bound Tightness Regularizer

Similar to the goal of stabilizing certified bounds at initialization, we also expect to keep the mean value of $\Delta_i$ in the current batch, $\hat{\mathbb{E}}(\Delta_i)$, stable along the warmup. Note that $\hat{\mathbb{E}}(\Delta_i)$ is empirically computed from a concrete batch and different from the expectation $\mathbb{E}(\Delta_i)$ at initialization In the initialization, we aim to make $\mathbb{E}(\Delta_i) \approx \mathbb{E}(\Delta_{i-1})$. Here, we relax the goal to making $\tau\hat{\mathbb{E}}(\Delta_i) \leq \hat{\mathbb{E}}(\Delta_0)$ with a configurable tolerance value $\tau$ $(0<\tau\leq 1)$, to balance the regularization power and the model capacity. We add the following regularization term:

$$\mathcal{L}_{\text{tightness}} = \frac{1}{\tau m}\sum_{i=1}^{m}\text{ReLU}(\tau - \frac{\hat{\mathbb{E}}(\Delta_0)}{\hat{\mathbb{E}}(\Delta_i)}), \tag{3.5}$$

where the training is penalized only when $\tau\hat{\mathbb{E}}(\Delta_i) > \hat{\mathbb{E}}(\Delta_0)$ due to the clipping effect by $\text{ReLU}(\cdot)$.

#### 3.2.3.2 ReLU activation states balancing regularizer

To balance ReLU activation states, we expect to balance the impact of active ReLU neurons and inactive neurons respectively. Here, we consider the center of the interval bound, $\mathbf{c}_i = (\underline{\mathbf{h}}_i + \overline{\mathbf{h}}_i)/2$, and we model the impact as the contribution of each type of neurons to the mean and variance of the whole layer, i.e., $\hat{\mathbb{E}}(\mathbf{c}_i)$ and $\text{Var}(\mathbf{c}_i)$ respectively. Note that in the beginning almost all neurons are unstable, and gradually most neurons become either active or inactive. Therefore, we add this regularizer only when there is at least one active neuron and one inactive neuron, which generally holds true unless at the training start. We use $\alpha_i$ to denote the ratio between the contribution of the active neurons and inactive neurons respectively to $\hat{\mathbb{E}}(\mathbf{c}_i)$, and similarly we use $\beta_i$ to denote the ratio of contribution to $\text{Var}(\mathbf{c}_i)$. They are computed as:

$$\alpha_i = \frac{\sum_j \mathbb{I}(\underline{\mathbf{h}}_{i,j} > 0)\mathbf{c}_{i,j}}{-\sum_j \mathbb{I}(\overline{\mathbf{h}}_{i,j} < 0)\mathbf{c}_{i,j}}, \qquad \beta_i = \frac{\sum_j \mathbb{I}(\underline{\mathbf{h}}_{i,j} > 0)(\mathbf{c}_{i,j} - \hat{\mathbb{E}}(\mathbf{c}_i))^2}{\sum_j \mathbb{I}(\overline{\mathbf{h}}_{i,j} < 0)(\mathbf{c}_{i,j} - \hat{\mathbb{E}}(\mathbf{c}_i))^2},$$

and in general $\alpha_i, \beta_i > 0$. We regard that the activation states are roughly balanced if $\alpha_i$ and $\beta_i$ are close to 1. With the same aforementioned tolerance $\tau$, we expect to make $\tau \leq \alpha_i, \beta_i \leq 1/\tau$, which is equivalent to making $\min(\alpha_i, 1/\alpha_i) \geq \tau$, $\min(\beta_i, 1/\beta_i) \geq \tau$. Thereby we design the following regularization term:

$$\mathcal{L}_{\text{relu}} = \frac{1}{\tau m} \sum_{i=1}^{m} \left( \text{ReLU}(\tau - \min(\alpha_i, \frac{1}{\alpha_i})) + \text{ReLU}(\tau - \min(\beta_i, \frac{1}{\beta_i})) \right). \tag{3.6}$$

### 3.2.4 Training Objectives

Certified robust training solves the robust optimization problem as Eq. (2.1), and when the inner maximization is verifiably solved, the base training objective without regularization is:

$$\mathcal{L}_{\text{rob}} = \overline{L}(f_\theta, \mathbf{x}, y, \epsilon), \quad \text{where } \overline{L}(f_\theta, \mathbf{x}, y, \epsilon) \geq \max_{\|\delta\|_\infty \leq \epsilon} L(f_\theta(\mathbf{x}+\delta), y), \tag{3.7}$$

such that $\overline{L}(f_\theta, \mathbf{x}, y, \epsilon)$ is an upper bound of $L(f_\theta(\mathbf{x}+\delta), y)$ given by a robustness verifier, e.g., IBP. In our proposed method, we first initialize the parameters with our IBP initialization, and then we perform a *short* warmup with gradually increasing $\epsilon$ ($0 \leq \epsilon \leq \epsilon_{\text{target}}$), where $\epsilon_{\text{target}}$ stands for the target perturbation radius that is usually equal to or slightly larger than the maximum perturbation radius used for test.

Our training objective $\mathcal{L}$ combines the ordinary objective Eq. (3.7) and the proposed regularizers:

$$\mathcal{L} = \mathcal{L}_{\text{rob}} + \lambda(\mathcal{L}_{\text{tightness}} + \mathcal{L}_{\text{relu}}), \tag{3.8}$$

where $\lambda$ is for balancing the regularizers and the original $\mathcal{L}_{\text{rob}}$ loss. For simplicity and efficiency, we use IBP to compute the bounds in $\mathcal{L}_{\text{rob}}$ and the regularizers. During warmup, we also gradually decrease $\lambda$ from $\lambda_0$ to 0 as $\epsilon$ grows, where $\lambda = \lambda_0(1 - \epsilon/\epsilon_{\text{target}})$. After warmup, we only use $\mathcal{L} = \mathcal{L}_{\text{rob}}$ for final training with $\epsilon_{\text{target}}$. Note that in the regularizers, the value of each ReLU$(\cdot)$ term has the same range $[0, \tau]$, and thus in Eq. (3.8) we directly sum up them without weighing them for simplicity. In test, we still only use pure IBP bounds

without any other tighter method.

## 3.3 Experiments

In the experiments, we demonstrate the effectiveness of our proposed method for training certifiably robust neural networks more efficiently while achieving better or comparable verified errors.

### 3.3.1 Settings

We adopt two datasets, MNIST [LCB10], CIFAR-10 [KH09]. Following [XSZ20], we consider three model architectures: a 7-layer feedforward convolutional network (CNN-7), Wide-ResNet [ZK16] and ResNeXt [XGD17]. According our discussion in Sec. 3.2.2, we also modify the models to fully add a BN after every convolutional or fully-connected layer. For target perturbation radii, we mainly use $\epsilon_{\text{target}} = 0.4$ for MNIST, $\epsilon_{\text{target}} = 8/255$ for CIFAR-10, and $\epsilon_{\text{target}} = 1/255$ for TinyImageNet, following prior works. We mainly compare with the following SOTA baselines on all the settings (note that in our main results, we also make these baselines use models with full BNs unless otherwise indicated):

Table 3.2: Standard and verified error rates (%) of models trained with different methods respectively on MNIST ($\epsilon_{\text{target}} = 0.4$) and CIFAR-10 ($\epsilon_{\text{target}} = 8/255$). Schedule is represented as the total number of epochs and the number of epochs in each of the three phases with $\epsilon = 0$, increasing $\epsilon \in (0, \epsilon_{\text{target}})$ and final $\epsilon = \epsilon_{\text{target}}$ respectively. We report the mean and standard deviation of the results on 5 repeats for CNN-7 and 3 repeats for Wide-ResNet and ResNeXt respectively. All models include BN after every layer (see Sec. 3.2.2). We also report the best run in "Ours (best)" since main results in prior works did not have repeats. Literature results with the "†" mark are concurrent works.

| Dataset | Schedule (epochs) | Method | CNN-7 (with full BN) | | Wide-ResNet (with full BN) | | ResNeXt (with full BN) | |
|---|---|---|---|---|---|---|---|---|
| | | | Standard | Verified | Standard | Verified | Standard | Verified |
| MNIST | 70 (0+20+50) | Vanilla IBP | 2.59 ± 0.06 | 12.03 ± 0.09 | 3.18 ± 0.05 | 12.93 ± 0.17 | 4.09 ± 0.46 | 15.36 ± 0.94 |
| | | CROWN-IBP [a] | 2.75 ± 0.12 | 12.04 ± 0.22 | 3.39 ± 0.05 | 13.10 ± 0.15 | 4.22 ± 0.53 | 15.24 ± 0.78 |
| | | Ours | **2.33 ± 0.08** | **11.03 ± 0.13** | **2.77 ± 0.02** | **11.76 ± 0.07** | **3.22 ± 0.08** | **13.43 ± 0.17** |
| | | Ours (best) | **2.20** | **10.82** | 2.75 | 11.69 | 3.17 | 13.20 |
| | Literature results | | | Warmup | | Total (epochs) | Standard | Verified |
| | [GDS18] | | | (2K+10K) steps | | 100 | 1.66 | 15.01 [b] |
| | [ZCX20] | | | (9 + 51) epochs | | 200 | 2.17 | 12.06 |
| | †IBP+ParamRamp [LGW21] [e] | | | (9 + 51) epochs | | 200 | 2.16 | 10.88 |
| | †CROWN-IBP+ParamRamp [LGW21] [e] | | | (9 + 51) epochs | | 200 | 2.36 | 10.61 |
| CIFAR-10 | 70 (1+20+49) | Vanilla IBP | 58.72 ± 0.27 | 69.88 ± 0.10 | 58.85 ± 0.22 | 69.77 ± 0.32 | 60.10 ± 0.27 | 71.19 ± 0.21 |
| | | CROWN-IBP [a] | 63.19 ± 0.36 | 71.29 ± 0.19 | 62.76 ± 0.23 | 71.82 ± 0.30 | 64.75 ± 0.50 | 72.50 ± 0.20 |
| | | Ours | **56.64 ± 0.48** | **68.81 ± 0.24** | **56.74 ± 0.40** | **68.71 ± 0.29** | **59.33 ± 0.86** | **70.62 ± 0.59** |
| | 160 (1+80+79) | Vanilla IBP | 53.80 ± 0.71 | 67.01 ± 0.29 | 54.31 ± 0.46 | 67.45 ± 0.21 | 55.23 ± 0.12 | 68.28 ± 0.15 |
| | | CROWN-IBP [a] | 58.76 ± 0.76 | 69.67 ± 0.38 | 60.39 ± 0.33 | 70.07 ± 0.42 | 61.08 ± 0.35 | 71.26 ± 0.11 |
| | | Ours | **51.72 ± 0.40** | **65.58 ± 0.32** | **51.95 ± 0.27** | **65.91 ± 0.14** | **53.68 ± 0.33** | **66.91 ± 0.40** |
| | | Ours (best) | **51.06** | **65.03** | 51.63 | 65.72 | 53.38 | 66.41 |
| | Literature results | | | Warmup | | Total (epochs) | Standard | Verified |
| | [GDS18] | | | (5K+50K) steps | | 3,200 | 50.51 | 68.44 [c] |
| | [ZCX20] | | | (320 + 1600) epochs | | 3,200 | 54.02 | 66.94 |
| | [BV20] | | | N/A [d] | | 800 | 48.3 | 72.5 |
| | [XSZ20] | | | (100 + 800) epochs | | 2,000 | 53.71 | 66.62 |
| | †IBP+ParamRamp [LGW21] [e] | | | (320 + 1600) epochs | | 3,200 | 55.28 | 67.09 |
| | †CROWN-IBP+ParamRamp [LGW21] [e] | | | (320 + 1600) epochs | | 3,200 | 51.94 | 65.08 |
| | †$\ell_\infty$-dist net (other architecture) [ZCL21] [f] | | | N/A [f] | | 800 | 48.32 | 64.90 |

[a] CROWN-IBP here follows [XSZ20] with loss fusion for efficiency, but we found it does not perform well with a short training schedule under our settings and usually requires a longer schedule to achieve good results.

[b] Some test results in [GDS18] are obtained with costly mixed integer programming (MIP) and linear programming (LP); we take IBP verified errors for fair comparison following [ZCX20].

[c] Additional PGD adversarial training was involved for this result, according to [ZCX20].

[d] [BV20] used a different training scheme and train the network layer by layer.

[e] [LGW21] use IBP-based and CROWN-IBP-based training respectively with their parameterized activation, and they use a tighter linear bound propagation method for testing instead of IBP.

[f] [ZCL21] use a very different model architecture with $\ell_\infty$ distance neurons rather than traditional DNNs, but still need a long schedule on both $\epsilon$ and $\ell_p$ norm where $p$ is gradually increased until $\infty$.

We conduct certified robust training using relatively short warmup schedules to demonstrate the effectiveness of our proposed techniques for fast training. We show the results in Table 3.2 for MNIST, CIFAR-10. Compared to Vanilla IBP and CROWN-IBP, our improved IBP training consistently achieves lower standard errors and verified errors under same schedules respectively, where BN is added to the models for all these three training methods. We find that CROWN-IBP with loss fusion [XSZ20] tends to require a larger number of epochs to obtain good results and it sometimes underperform Vanilla IBP under short schedules, but disabling loss fusion can make it much more costly and unscalable. In terms of the best results, we achieve verified error 10.82% on MNIST $\epsilon_{\text{target}} = 0.4$, 65.03% on CIFAR-10 $\epsilon_{\text{target}} = 8/255$, and 82.36% on TinyImageNet $\epsilon_{\text{target}} = 1/255$, which makes a notable improvement over literature SOTA [GDS18, XSZ20] that used long training schedules. Compared to concurrent works [LGW21, ZCL21] which use different improvement techniques, we have comparable verified errors, but they still need long training schedules. For reference, we tried [ZCL21] which used a different architecture with "$\ell_\infty$ distance neurons" rather than convolution-based DNNs. On CIFAR-10 using 160 total epochs by reducing their training schedule proportionally, their verified error is 68.44% which is much higher than ours. Overall, the results demonstrate that our improved IBP training is effective for more efficient certified robust training with a shorter warmup. For more experiments, please refer to [SWZ21].

# CHAPTER 4

# Convergence Analysis for IBP Training

## 4.1 Preliminaries

### 4.1.1 Neural Networks

We consider a similar network architecture as used in [DZP18] – a two-layer ReLU network. Unlike [DZP18] which considered a regression task with a square loss, here we consider a classification task where IBP is usually used, and we consider binary classification for simplicity. On a training dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, for every $i \in [n]$, $(\mathbf{x}_i, y_i)$ is a training example with $d$-dimensional input $\mathbf{x}_i (\mathbf{x}_i \in \mathbb{R}^d)$ and label $y_i (y_i \in \{\pm 1\})$, and the network output is:

$$f(\mathbf{W}, \mathbf{a}, \mathbf{x}_i) = \frac{1}{\sqrt{m}} \sum_{r=1}^{m} a_r \sigma(\mathbf{w}_r^\top \mathbf{x}_i), \tag{4.1}$$

where $m$ is the width of hidden layer (the first layer) in the network, $\mathbf{W} \in \mathbb{R}^{m \times d}$ is the weight matrix of the hidden layer, $\mathbf{w}_r (r \in [m])$ is the $r$-th row of $\mathbf{W}$, $\mathbf{a} \in \mathbb{R}^m$ is the weight vector of the second layer (output layer) with elements $a_1, \cdots, a_m$, and $\sigma(\cdot)$ is the activation function. We assume the activation is ReLU as IBP is typically used with ReLU. For initialization, we set $a_r \sim \text{unif}[\{1, -1\}]$ and $\mathbf{w}_r \sim \mathbf{N}(0, \mathbf{I})$. Following [DZP18], we assume the second layer is fixed after initialization, and we only train the first layer. Since we consider binary classification, we use a logistic loss. For training example $(\mathbf{x}_i, y_i)$, we define $u_i(\mathbf{W}, \mathbf{a}, \mathbf{x}_i) := y_i f(\mathbf{W}, \mathbf{a}, \mathbf{x}_i)$, and then the loss for this example is computed as $l(u_i(\mathbf{W}, \mathbf{a}, \mathbf{x}_i)) = \log(1 +$

$\exp(-u_i(\mathbf{W}, \mathbf{a}, \mathbf{x}_i)))$, and the standard training loss on the whole training set is

$$L = \sum_{i=1}^{n} l(u_i(\mathbf{W}, \mathbf{a}, \mathbf{x}_i)) = \sum_{i=1}^{n} \log\Big(1 + \exp(-u_i(\mathbf{W}, \mathbf{a}, \mathbf{x}_i))\Big). \tag{4.2}$$

### 4.1.2 IBP-based Certified Robust Training

In the robust training setting, for some original input $\mathbf{x}_i$ ($\forall i \in [n]$), we consider that the actual input to the model may be perturbed into $\mathbf{x}_i + \Delta_i$ by perturbation $\Delta_i$. For a widely adopted setting in the adversarial robustness area, we consider $\ell_\infty$ perturbations, where the perturbation is bounded by an $\ell_\infty$ ball with radius $\epsilon(0 \leq \epsilon \leq 1)$, i.e., $\|\Delta_i\|_\infty \leq \epsilon$. For the convenience of subsequent analysis and without loss of generality, we set the following assumption on each $\mathbf{x}_i$:

**Assumption 1.** *$\forall i \in [n]$, we assume there exists some $\xi > 0$, such that $\mathbf{x}_i \in [\epsilon, 1]^d$, $\|\mathbf{x}_i\|_2 \geq \xi$.*

This assumption can be easily satisfied by normalizing the training data. Through out the remaining part of this paper, we assume this assumption holds. In [DZP18], they also assume there are no parallel data points, and in our case we assume this holds under any possible perturbation, formulated as:

**Assumption 2.** *For perturbation radius $\epsilon$, we assume that*

$$\forall i, j \in [n], i \neq j, \forall \mathbf{x}_i' \in B_\infty(\mathbf{x}_i, \epsilon), \forall \mathbf{x}_j' \in B_\infty(\mathbf{x}_j, \epsilon), \quad \mathbf{x}_i' \nparallel \mathbf{x}_j',$$

*where $B_\infty(\mathbf{x}_i, \epsilon)$ stands for the $\ell_\infty$ ball with radius $\epsilon$ centered at $\mathbf{x}_i$.*

IBP training computes and optimizes a robust loss $\overline{L}$, which is an upper bound of the standard loss for any possible perturbation $\Delta_i(\forall i \in [n])$:

$$\overline{L} \geq \sum_{i=1}^{n} \max_{\Delta_i} \Big\{ \log\Big(1 + \exp(-y_i f(\mathbf{W}, \mathbf{a}, \mathbf{x}_i + \Delta_i))\Big) \;\mid\; \|\Delta_i\|_\infty \leq \epsilon \Big\}.$$

To compute $\overline{L}$, since $\log(\cdot)$ and $\exp(\cdot)$ are both monotonic, for every $i \in [n]$, IBP first computes the lower bound of $u_i(\mathbf{W}, \mathbf{a}, \mathbf{x}_i + \Delta_i)$ for $\|\Delta_i\|_\infty \leq \epsilon$, denoted as $\underline{u_i}$. Then the IBP robust loss is:

$$\overline{L} = \sum_{i=1}^{n} \log(1 + \exp(-\underline{u_i})), \quad \text{where } \underline{u_i} \leq \min_{\Delta_i} u_i(\mathbf{W}, \mathbf{a}, \mathbf{x}_i + \Delta_i) \ (i = 1, 2, \cdots, n). \tag{4.3}$$

For all $i \in [n]$, IBP computes and propagates an interval lower and upper bound for each neuron in the network, and then $\underline{u_i}$ is equivalent to the lower bound of the final output neuron. Initially, the interval bound of the input is $[\mathbf{x}_i - \epsilon \cdot \mathbf{1}, \mathbf{x} + \epsilon \cdot \mathbf{1}]$. Given constraints of $\Delta_i$, we have the interval bound of each neuron in the first layer:

$$\forall r \in [m], \ \ \sigma\big(\mathbf{w}_r^\top \mathbf{x}_i - \epsilon \|\mathbf{w}_r\|_1\big) \leq \sigma\big(\mathbf{w}_r^\top (\mathbf{x}_i + \Delta_i)\big) \leq \sigma\big(\mathbf{w}_r^\top \mathbf{x}_i + \epsilon \|\mathbf{w}_r\|_1\big). \tag{4.4}$$

Then these interval bounds are propagated to the second layer. We focus on the lower bound of $u_i$, which can be computed from the bounds of the first layer by considering the sign of multiplier $y_i a_r$:

$$u_i = y_i \frac{1}{\sqrt{m}} \sum_{r=1}^{m} a_r \sigma(\mathbf{w}_r^\top (\mathbf{x}_i + \Delta_i)) \tag{4.5}$$

$$\geq \frac{1}{\sqrt{m}} \sum_{r=1}^{m} \mathbb{1}(y_i a_r = 1) \sigma\big(\mathbf{w}_r^\top \mathbf{x}_i - \epsilon \|\mathbf{w}_r\|_1\big) + \mathbb{1}(y_i a_r = -1) \sigma\big(\mathbf{w}_r^\top \mathbf{x}_i + \epsilon \|\mathbf{w}_r\|_1\big) \coloneqq \underline{u_i}. \tag{4.6}$$

Then the IBP robust loss can be obtained as Eq. (4.3). And we define $\underline{\mathbf{u}} \coloneqq (\underline{u_1}, \underline{u_2}, \cdots, \underline{u_n})$.

For certified robust training, we can define a **certified robust accuracy** in IBP training, which is the percentage of examples that IBP bounds can successfully certify that the prediction is correct for any possible perturbation. For every example $i (i \in [n])$, it is considered as classified correctly and robustly under IBP verification, if and only if $\underline{u_i} > 0$ where $\underline{u_i}$ is computed from IBP. Let $\tilde{u}_i$ be the *exact* solution of the minimization in Eq. (4.3) rather than relaxed IBP bounds, we can also define a **true robust accuracy**, where the

robustness of prediction requires $\tilde{u}_i > 0$. The certified robust accuracy by IBP is a provable lower bound of the true robust accuracy.

### 4.1.3 Gradient Flow and Gram Matrix

To analyze the convergence of IBP training, we adopt a continuous time analysis with gradient flow, which is gradient descent with infinitesimal step size and is also used in prior works for standard training [ACH18, DLL18, DZP18]. In the gradient flow for IBP training,

$$\forall r \in [m], \quad \frac{d\mathbf{w}_r(t)}{dt} = -\frac{\partial \overline{L}(t)}{\partial \mathbf{w}_r(t)}, \tag{4.7}$$

where $\mathbf{w}_1(t), \mathbf{w}_2(t), \cdots, \mathbf{w}_m(t)$ are rows of the weight matrix at time $t$, and $\overline{L}(t)$ is the IBP robust loss defined as Eq. (4.3) using weights at time $t$.

Under the gradient flow setting as Eq. (4.7), for all $i \in [n]$, we analyze the dynamics of $\underline{u}_i$ during IBP training, and we use $\underline{u}_i(t)$ to denote the value of $\underline{u}_i$ at time $t$:

$$\frac{d}{dt}\underline{u}_i(t) = \sum_{r=1}^{m} \left\langle \frac{\partial \underline{u}_i(t)}{\partial \mathbf{w}_r(t)}, \frac{d\mathbf{w}_r(t)}{dt} \right\rangle = \sum_{j=1}^{n} -l'(\underline{u}_j)\mathbf{H}_{ij}(t), \tag{4.8}$$

where $l'(\underline{u}_j)$ is the derivative of the loss, $\mathbf{H}(t)$ is defined as $\mathbf{H}_{ij}(t) = \sum_{r=1}^{m} \left\langle \frac{\partial \underline{u}_i(t)}{\partial \mathbf{w}_r(t)}, \frac{\partial \underline{u}_j(t)}{\partial \mathbf{w}_r(t)} \right\rangle$ ($\forall 1 \le i, j \le n$), and we provide a detailed derivation in Appendix **??**. With the definition of $\mathbf{H}$, we can describe the dynamic of $\underline{u}_i$ under the gradient flow using $\mathbf{H}$.

From Eq. (4.6), $\forall i \in [n]$, $r \in [m]$, derivative $\frac{\partial \underline{u}_i(t)}{\partial \mathbf{w}_r(t)}$ can be computed as follows:

$$\frac{\partial \underline{u}_i(t)}{\partial \mathbf{w}_r(t)} = \frac{1}{\sqrt{m}} y_i a_r \Big( A_{ri}^{+}(t)(\mathbf{x}_i - \epsilon \operatorname{sign}(\mathbf{w}_r(t))) + A_{ri}^{-}(t)(\mathbf{x}_i + \epsilon \operatorname{sign}(\mathbf{w}_r(t))) \Big),$$

where $\text{sign}(\mathbf{w}_r(t))$ is element-wise for $\mathbf{w}_r(t)$, and we define *indicators*

$$A_{ri}^{+}(t) \coloneqq \mathbb{1}(y_i a_r = 1, \mathbf{w}_r(t)^\top \mathbf{x}_i - \epsilon \|\mathbf{w}_r(t)\|_1 > 0), \tag{4.9}$$

$$A_{ri}^{-}(t) \coloneqq \mathbb{1}(y_i a_r = -1, \mathbf{w}_r(t)^\top \mathbf{x}_i + \epsilon \|\mathbf{w}_r(t)\|_1 > 0), \tag{4.10}$$

where terms $(\mathbf{w}_r(t)^\top \mathbf{x}_i - \epsilon \|\mathbf{w}_r(t)\|_1 > 0)$ and $(\mathbf{w}_r(t)\mathbf{x}_i + \epsilon \|\mathbf{w}_r(t)\|_1 > 0)$ stand for the active status of the activation. Then elements in $\mathbf{H}$ can be written as $(\forall 1 \le i, j \le n)$:

$$\begin{aligned}\mathbf{H}_{ij}(t) = \frac{1}{m} y_i y_j \sum_{r=1}^{m} a_r^2 \Big( A_{ri}^{+}(t)(\mathbf{x}_i - \epsilon \,\text{sign}(\mathbf{w}_r(t))) + A_{ri}^{-}(t)(\mathbf{x}_i + \epsilon \,\text{sign}(\mathbf{w}_r(t))) \Big)^\top \\ \Big( A_{rj}^{+}(t)(\mathbf{x}_j - \epsilon \,\text{sign}(\mathbf{w}_r(t))) + A_{rj}^{-}(t)(\mathbf{x}_j + \epsilon \,\text{sign}(\mathbf{w}_r(t))) \Big)\end{aligned} \tag{4.11}$$

$$= \frac{1}{m} y_i y_j \Big( \mathbf{x}_i^\top \mathbf{x}_j \sum_{r=1}^{m} \alpha_{rij}(t) - \epsilon \big( \sum_{r=1}^{m} (\beta_{rij}(t)\mathbf{x}_i + \beta_{rji}(t)\mathbf{x}_j)^\top \text{sign}(\mathbf{w}_r(t)) \big) + \epsilon^2 d \sum_{r=1}^{m} \gamma_{rij}(t) \Big), \tag{4.12}$$

$$\text{where} \qquad \alpha_{rij}(t) = (A_{ri}^{+}(t) + A_{ri}^{-}(t))(A_{rj}^{+}(t) + A_{rj}^{-}(t)), \tag{4.13}$$

$$\beta_{rij}(t) = (A_{ri}^{+}(t) + A_{ri}^{-}(t))(A_{rj}^{+}(t) - A_{rj}^{-}(t)), \tag{4.14}$$

$$\gamma_{rij}(t) = (A_{ri}^{+}(t) - A_{ri}^{-}(t))(A_{rj}^{+}(t) - A_{rj}^{-}(t)). \tag{4.15}$$

Further, we define Gram matrix $\mathbf{H}^\infty$ which is the elementwise expectation of $\mathbf{H}(0)$, to characterize $\mathbf{H}(0)$ on the random initialization basis:

$$\forall 1 \le i, j \le n, \;\; \mathbf{H}_{ij}^\infty \coloneqq \mathbb{E}_{\forall 1 \le r \le m, \mathbf{w}_r \sim \mathbf{N}(0, \mathbf{I}), a_r \sim \text{unif}[\{-1,1\}]} \mathbf{H}_{ij}(0),$$

where $\mathbf{H}_{ij}(0)$ depends on the initialization of weights $\mathbf{w}_r$ and $a_r$. We also define $\lambda_0 \coloneqq \lambda_{\min}(\mathbf{H}^\infty)$ as the smallest eigenvalue of $\mathbf{H}^\infty$.

## 4.2 Convergence Analysis for IBP Training

We present the following main theorem which shows the convergence of IBP training under certain conditions on perturbation radius and network width:

**Theorem 1** (Convergence of IBP Training)**.** *Suppose Assumption 1 and 2 hold for the training data, and the $\ell_\infty$ perturbation radius satisfies $\epsilon \leq O(\min(\frac{\delta^2\lambda_0^2}{d^{2.5}n^3}, \frac{\sqrt{2d}R}{\log(\sqrt{\frac{2\pi d}{R}}\xi)}))$, where $R = \frac{c\delta\lambda_0}{d^{1.5}n^2}$, $c = \frac{\sqrt{2\pi}\xi}{384}$. For a two-layer ReLU network (Eq. (4.1)), suppose its width for the first hidden layer satisfies $m \geq \Omega\bigg(\Big(\frac{d^{1.5}n^4\delta\lambda_0}{\delta^2\lambda_0^2 - \epsilon d^{2.5}n^4}\Big)^2\bigg)$, and the network is randomly initialized as $a_r \sim unif[\{1,-1\}], \mathbf{w}_r \sim \mathbf{N}(0,\mathbf{I})$, with the second layer fixed during training. Then for any confidence $\delta(0<\delta<1)$, with probability at least $1-\delta$, IBP training with gradient flow can converge to zero training error.*

**Theorem 2** (Convergence Rate of IBP Training)**.** *With the same assumptions and notations in 1 and $t \leq \frac{c\delta\lambda_0\sqrt{m}}{d^{1.5}n^3}$, we have*

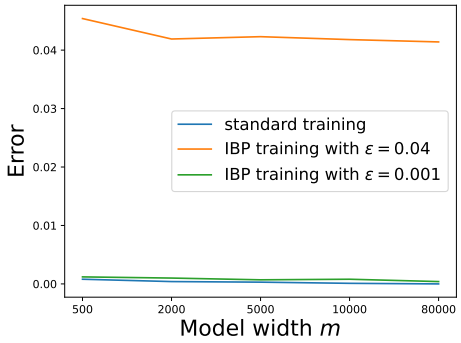
$$\overline{L}(t) \leq \exp\bigg(2\overline{L}(0)\bigg)\overline{L}(0)\exp\bigg(-\frac{\lambda_0 t}{2}\bigg),$$

*where $\overline{L}(0)$ is the IBP loss at initialization $t=0$, and $\mathbf{w}_r(t)$ is the weight $\mathbf{w}_r$ at time $t$.*

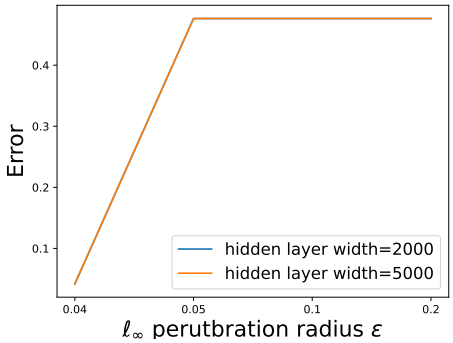
The proof is inspired by the convergence analysis of standard training based on Neural Tangent Kernel [DZP18]. Please refer to [WSG21] for detailed proof.

## 4.3 Experiments

We further conduct experiments to compare the convergence of networks with different width $m$ for natural training and IBP training respectively. We use the MNIST [LCB10] dataset and choose digit images with label 2 and 5 to construct a binary classification dataset. And we use a two-layer fully-connected ReLU network with a variable width. In each experiment, we train the model for 70 epochs. For IBP training, we keep $\epsilon$ fixed throughout the whole




(a) Final training error of standard training and IBP (with $\epsilon \in \{0.001, 0.04\}$) respectively, when the width $m$ of the model is varied.




(b) Final training error of IBP training (on models with width 2000 and 5000 respectively), when the perturbation radius $\epsilon$ is varied.

Figure 4.1: Experimental results.

training process. For all the experiments, we use the SGD optimizer. We present results in Figure 4.1. First, compared with standard training, for the same network width $m$, IBP has higher training errors (Figure 4.1a). Second, for relatively large $\epsilon$ ($\epsilon = 0.04$), even if we enlarge $m$ up to 80,000 which is limited by the memory of a single GeForce RTX 2080 GPU, IBP certified robust error is far away from 0 (Figure 4.1a). This is consistent to our main theorem that when $\epsilon$ is too large, simply enlarging $m$ can not guarantee the convergence. Moreover, when $\epsilon$ grows even larger, it can be difficult to even start the training, although standard training is possible. IBP training stucks in a local minima of random guess (with errors close to 50%) at the beginning of training (Figure 4.1b). Therefore in IBP-based training, existing works typically use a scheduling on $\epsilon$ to gradually increase $\epsilon$ from 0 until the target value. Theoretically, we believe that this is partly because $\lambda_0$ can be very small with a large perturbation, and then the training can be much more difficult, while this difficulty cannot be alleviated by simply enlarging the network width $m$. Overall, the empirical observations match our theoretical results.

# CHAPTER 5

# Conclusion

This paper theoretically analyzes the challenges and difficulty in IBP-based certified robust training, and also provides empirical improvement for fast and better IBP training. However, there is still a significant performance gap between certified robust training and empirical defense methods like adversarial training, and this demands a deeper understanding of the neural networks and more essential improvements in the future.

## REFERENCES


[ACH18] Sanjeev Arora, Nadav Cohen, and Elad Hazan. "On the Optimization of Deep Networks: Implicit Acceleration by Overparameterization." In *International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 244–253, 2018.

[ALS18] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. "A convergence theory for deep learning via over-parameterization." *arXiv preprint arXiv:1811.03962*, 2018.

[BTT17] Rudy Bunel, Ilker Turkaslan, Philip H. S. Torr, Pushmeet Kohli, and M. Pawan Kumar. "Piecewise Linear Neural Network verification: A comparative study." *CoRR*, **abs/1711.00455**, 2017.

[BV20] Mislav Balunovic and Martin Vechev. "Adversarial training and provable defenses: Bridging the gap." In *International Conference on Learning Representations*, 2020.

[CW17] Nicholas Carlini and David Wagner. "Adversarial examples are not easily detected: Bypassing ten detection methods." In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 3–14. ACM, 2017.

[CZC17] Hongge Chen, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, and Cho-Jui Hsieh. "Attacking visual language grounding with adversarial examples: A case study on neural image captioning." *arXiv preprint arXiv:1712.02051*, 2017.

[CZK19] Jun-Ho Choi, Huan Zhang, Jun-Hyuk Kim, Cho-Jui Hsieh, and Jong-Seok Lee. "Evaluating robustness of deep image super-resolution against adversarial attacks." In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 303–311, 2019.

[DLL18] Simon S Du, Jason D Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. "Gradient descent finds global minima of deep neural networks." *arXiv preprint arXiv:1811.03804*, 2018.

[DZP18] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. "Gradient descent provably optimizes over-parameterized neural networks." *arXiv preprint arXiv:1810.02054*, 2018.

[GB10] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. JMLR Workshop and Conference Proceedings.

[GDS18] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. "On the effectiveness of interval bound propagation for training verifiably robust models." *arXiv preprint arXiv:1810.12715*, 2018.

[GSS15] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." In *ICLR*, 2015.

[HZR15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification." In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[IS15] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

[JGH18] Arthur Jacot, Franck Gabriel, and Clément Hongler. "Neural tangent kernel: Convergence and generalization in neural networks." In *NIPS*, 2018.

[KBD17] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. "Reluplex: An efficient SMT solver for verifying deep neural networks." In *International Conference on Computer Aided Verification*, pp. 97–117. Springer, 2017.

[KGB16] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. "Adversarial examples in the physical world." *arXiv preprint arXiv:1607.02533*, 2016.

[KH09] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images." *Technical Report TR-2009*, 2009.

[LCB10] Yann LeCun, Corinna Cortes, and CJ Burges. "MNIST handwritten digit database." *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, **2**, 2010.

[LGW21] Zhaoyang Lyu, Minghao Guo, Tong Wu, Guodong Xu, Kehuan Zhang, and Dahua Lin. "Towards Evaluating and Training Verifiably Robust Neural Networks.", 2021.

[LL18] Yuanzhi Li and Yingyu Liang. "Learning Overparameterized Neural Networks via Stochastic Gradient Descent on Structured Data." In *Advances in Neural Information Processing Systems*, pp. 8168–8177, 2018.

[LSS19] Lu Lu, Yeonjong Shin, Yanhui Su, and George Em Karniadakis. "Dying relu and initialization: Theory and numerical examples." *arXiv preprint arXiv:1903.06733*, 2019.

[MGV18] Matthew Mirman, Timon Gehr, and Martin Vechev. "Differentiable abstract interpretation for provably robust neural networks." In *International Conference on Machine Learning*, pp. 3575–3583, 2018.

[MMS18] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. "Towards deep learning models resistant to adversarial attacks." In *ICLR*, 2018.

[RSL18] Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang. "Semidefinite relaxations for certifying robustness to adversarial examples." In *Advances in Neural Information Processing Systems*, pp. 10877–10887, 2018.

[SGM18] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. "Fast and Effective Robustness Certification." In *Advances in Neural Information Processing Systems*, pp. 10825–10836, 2018.

[SGP19] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. "An abstract domain for certifying neural networks." *Proceedings of the ACM on Programming Languages*, **3**(POPL):41, 2019.

[SMG13] Andrew M Saxe, James L McClelland, and Surya Ganguli. "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks." *arXiv preprint arXiv:1312.6120*, 2013.

[STI18] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. "How does batch normalization help optimization?" In *Proceedings of the 32nd international conference on neural information processing systems*, pp. 2488–2498, 2018.

[SWZ21] Zhouxing Shi, Yihan Wang, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. "Fast Certified Robust Training via Better Initialization and Shorter Warmup." *arXiv preprint arXiv:2103.17268*, 2021.

[SZC18] Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen, and Yupeng Gao. "Is Robustness the Cost of Accuracy?–A Comprehensive Study on the Robustness of 18 Deep Image Classification Models." In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 631–648, 2018.

[SZS13] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. "Intriguing properties of neural networks." In *ICLR*, 2013.

[Van17] Twan Van Laarhoven. "L2 regularization versus batch and weight normalization." *arXiv preprint arXiv:1706.05350*, 2017.

[WK18] Eric Wong and Zico Kolter. "Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope." In *International Conference on Machine Learning*, pp. 5283–5292, 2018.

[WPW18] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. "Efficient formal safety analysis of neural networks." In *Advances in Neural Information Processing Systems*, pp. 6367–6377, 2018.

[WSG21] Yihan Wang, Zhouxing Shi, Quanquan Gu, and Cho-Jui Hsieh. "On the Convergence of Certified Robust Training with Interval Bound Propagation." In *International Conference on Learning Representations*, 2021.

[WSM18] Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J Zico Kolter. "Scaling provable adversarial defenses." In *NIPS*, 2018.

[WZX21] Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. "Beta-CROWN: Efficient Bound Propagation with Per-neuron Split Constraints for Complete and Incomplete Neural Network Verification." *arXiv preprint arXiv:2103.06624*, 2021.

[XGD17] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. "Aggregated residual transformations for deep neural networks." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

[XSZ20] Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. "Automatic perturbation analysis for scalable certified robustness and beyond." *Advances in Neural Information Processing Systems*, **33**, 2020.

[ZCL21] Bohang Zhang, Tianle Cai, Zhou Lu, Di He, and Liwei Wang. "Towards Certifying $\ell_\infty$ Robustness using Neural Networks with $\ell_\infty$-dist Neurons." *arXiv preprint arXiv:2102.05363*, 2021.

[ZCX20] Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Duane Boning, and Cho-Jui Hsieh. "Towards stable and efficient training of verifiably robust neural networks." In *International Conference on Learning Representations*, 2020.

[ZCZ18] Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu. "Stochastic gradient descent optimizes over-parameterized deep relu networks." *arXiv preprint arXiv:1811.08888*, 2018.

[ZK16] Sergey Zagoruyko and Nikos Komodakis. "Wide residual networks." *arXiv preprint arXiv:1605.07146*, 2016.

[ZWC18] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. "Efficient neural network robustness certification with general activation functions." In *Advances in neural information processing systems*, pp. 4939–4948, 2018.