Fast Certified Robust Training with Short Warmup

Anonymous Author(s)
Affiliation
Address
email

Abstract

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 per-batch training complexity similar to standard neural network training, they usually use a long warmup schedule with hundreds or thousands epochs to reach SOTA performance and are thus still costly. In this paper, we identify two important issues in existing methods, 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. To mitigate these issues and conduct certified training with shorter warmup, we propose three improvements: 1) We derive a new weight initialization method for IBP training; 2) We propose to fully add Batch Normalization (BN) to each layer in the model, since we find BN can reduce the imbalance in ReLU activation states; 3) We also design regularization to explicitly tighten certified bounds and balance ReLU activation states. In our experiments, we are able to obtain **65.03%** verified error on CIFAR-10 ($\epsilon = \frac{8}{255}$) and **82.36%** verified error on TinyImageNet ($\epsilon = \frac{1}{255}$) using very short training schedules (**160 and 80 total epochs**, respectively), outperforming literature SOTA trained with hundreds or thousands epochs under the same network architecture.

1 Introduction

While deep neural networks (DNNs) are successfully applied in various areas, its robustness problem have attracted great attention since the discovery of adversarial examples [41, 12, 6, 23, 7, 31, 40, 8], which poses concerns in DNN applications especially the safety-critical ones such as healthcare and autonomous driving. Methods for improving the empirical robustness of DNNs, such as adversarial training [31], provide no provable robustness guarantees, and thus some recent works aim to pursue *certified robustness*. Specifically, the robustness is evaluated in a certifiable manner using robustness verification methods [19, 54, 46, 38, 39, 5, 35, 44, 50, 45], which verify whether the model is provably robust under all possible input perturbations, usually by efficiently computing the output bounds.

To improve certified robustness, *certified robust training* methods (also referred to as certified defense) minimize a certified loss computed by a verifier, and the certified loss is an upper bound of the worst-case loss given specified input perturbations. So far, Interval Bound Propagation (IBP) [13, 32] and CROWN-IBP [55, 50] are the most efficient and effective methods for general models. IBP computes an interval with the output lower and upper bounds for each neuron, and CROWN-IBP further combines IBP with tighter linear relaxation-based bounds [54, 39] during warmup.

Both IBP and CROWN-IBP with loss fusion [50] have a per-batch training time complexity similar to standard DNN training. However, certified robust 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

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

regular (non-robust) loss first and then gradually increasing the perturbation radius from 0 to the target value in the robust loss. For example, generalized CROWN-IBP in Xu et al. [50] used 900 epochs for warmup and 2,000 epochs in total to train a convolutional model on CIFAR-10 [21].

In this paper, we identify two important issues in existing certified training. First, we find that the bounds obtained by certified training methods can be exploded at the training start, which is partly due to the suboptimal *weight initialization* in prior works. A good weight initialization is important for successful DNN training [11, 14], but prior works for certified training generally use weight initialization methods originally designed for regular network training, while certified training is essentially optimizing a different type of augmented network defined by robustness verification [55]. Such initializations can lead to exploded certified bounds during the training start. The long warmup with gradually increasing perturbation radii in prior works can somewhat be viewed as finding a better initialization for final IBP training with the target radius, but it is too costly. Second, we also observe that *IBP leads to imbalanced ReLU activation states*, where the model prefers inactive (dead) ReLU neurons significantly more than other states because inactive neurons tend to tighten IBP bounds. It can however hamper classification performance if too many neurons are dead.

We focus on improving IBP training since IBP is efficient per batch, with the following improvements:

- We derive a new weight initialization, *IBP initialization*, for IBP-based certified training. The new initialization can stabilize the tightness of certified bounds at initialization.
- We identify the benefit of Batch Normalization (BN) in certified training, and we find BN which normalizes pre-activation outputs can balance ReLU activation states and also stabilize variance. We propose to fully add BN to every layer, while it was partly or fully missed in prior works.
- We further propose regularizers to explicitly stabilize certified bounds and balance ReLU activation states during warmup.

We are able to efficiently train certifiably robust models that outperform previous SOTA performance in significantly shorter training epochs. We achieve a verified error of **65.03%** ($\epsilon = \frac{8}{255}$) on CIFAR-10 in **160** total training epochs, and **82.36%** on TinyImageNet ($\epsilon = \frac{1}{255}$) in **80** epochs, based on efficient IBP training. Under the same convolution-based architecture, we significantly reduce the total training cost by $20 \sim 60$ times compared to previous SOTA [55, 50] or concurrent work [30].

2 Background and Related Work

Certified Robust Training Training robust neural networks can generally be viewed as solving the following min-max optimization problem:

$$\min_{\theta} \mathbb{E}_{(\mathbf{x},y)\in\mathcal{X}} \left[\max_{\delta\in\Delta(\mathbf{x})} L(f_\theta(\mathbf{x}+\delta), y) \right], \tag{1}$$

where f_θ stands for a neural network parameterized by θ, $\mathcal{X}$ is the data distribution, $\mathbf{x}$ is a data example, y is its ground-truth label, δ is a perturbation constrained by $\Delta(\mathbf{x})$, and L is the loss function. *Adversarial training* methods [12, 31] solve the inner maximization in Eq. (1) with adversarial attack and then solve the outer minimization. And for robustness guarantees, *certified robust training* methods computes a certified upper bound for the inner maximization. Raghunathan et al. [34] used semidefinite relaxations for small two-layer models, and Wong & Kolter [46], Mirman et al. [32], Dvijotham et al. [10], Wang et al. [43] used linear relaxations but are still too computationally expensive for large models. On the other hand, Mirman et al. [32] first used interval bounds to train a certifiably robust network, and Gowal et al. [13] made it more effective. This approach is often referred to as interval bound propagation (IBP). CROWN-IBP [55] further combined IBP with tighter linear relaxation bounds by CROWN [54] during warmup, and it is generalized and accelerated in Xu et al. [50]. Additionally, Balunovic & Vechev [2] combined certified training with adversarial training; Xiao et al. [48] added a ReLU stability regularizer to adversarial training, to reduce unstable neurons for faster and tighter verification when tested with mixed integer programming (MIP), but their objective is distinct from ours and this method was shown not to improve certified training [27]. In concurrent works, Lyu et al. [30] proposed a parameterized ramp function as an alternative activation function, and used a tighter linear bound propagation algorithm for verification; Zhang et al. [53] proposed to use a different architecture with "ℓ_∞ distance neurons" instead of traditional linear or convolutional layers. Yet they still need long training schedules. Moreover, there are also

randomization based works for probabilistic certified defense [9, 28, 26, 36], but they require sampling and they are usually for ℓ_2 perturbations and have fundamental limitations for ℓ_∞ ones [51, 4, 22].

Weight Initialization of Neural Networks Many prior works have studied the weight initialization for standard DNN training. Xavier or Glorot initialization [11], adopted by popular deep learning libraries such as PyTorch [33] and Tensorflow [1] as the default initialization, aim to stabilize the magnitude of forward propagation and gradient backpropagation signals measured with variance. It uses a uniform distribution or normal distribution to independently initialize each element in the weight matrix with a derived variance for the distribution. [37] proposed an orthogonal initialization which may lead to better learning dynamics. And Kaiming initialization [14] derived an initialization for ReLU networks. Some other works also derived initializations for other specific DNN structures [42, 17], or automatically learning initializations [3, 56]. However, these initializations were designed for standard training, while they can lead to exploded certified bounds for IBP training.

3 Methodology

3.1 Notations and Definitions

We focus on improving IBP training. We consider a commonly adopted ℓ_∞ perturbation setting in adversarial robustness on a K-way classification task. For a DNN $f_\theta(\mathbf{x})$ with clean input $\mathbf{x}$, there can be some perturbation δ satisfying $\|\delta\|_\infty \leq \epsilon$, and the actual perturbed input to the model is $\mathbf{x}+\delta$. In robustness verification for achieving certified robustness, we verify whether

$$[f_\theta(\mathbf{x}+\delta)]_y - [f_\theta(\mathbf{x}+\delta)]_i > 0, \ \ \forall \|\delta\|_\infty \leq \epsilon, i \neq y \tag{2}$$

holds true, where $[f_\theta(\mathbf{x}+\delta)]_i$ is the logit score for class i and y is the ground-truth. This is equivalent to verifying whether the DNN provably makes correct prediction for all input $\mathbf{x}+\delta$ ($\|\delta\|_\infty \leq \epsilon$). For $f(\theta)$, we assume that there are m hidden affine layers (either convolutional or fully-connected layers) with ReLU activation. We use $\mathbf{h}_i$ to denote the pre-activation output value of the i-th layer, and use $\mathbf{z}_i = \text{ReLU}(\mathbf{h}_i)$ to denote the post-activation value. We use $\mathbf{W}_i$ and $\mathbf{b}_i$ to denote the parameters of the convolutional or fully-connected layer, 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 [15]. This is straightforward for a fully-connected layer, and for a convolutional layer with kernel size k, c_{in} input channels and c_{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 and $\mathbf{z}_0$ is not applicable. For IBP [32, 13], it computes and propagates the lower and upper bound interval of each $\mathbf{h}_i$ layer by layer until the last year or verification objective, denoted as interval $[\underline{\mathbf{h}}_i, \overline{\mathbf{h}}_i]$ such that $\underline{\mathbf{h}}_i \leq \mathbf{h}_i \leq \overline{\mathbf{h}}_i$ ($\forall \|\delta\|_\infty \leq \epsilon$). Finally Eq. (2) can be verified by checking the lower bound of $[f_\theta(\mathbf{x}+\delta)]_y - [f_\theta(\mathbf{x}+\delta)]_i$.

3.2 Issues in Existing Certified Robust Training

In this section, we will first analyze two issues in previous certified robust training, including exploded bounds at initialization, and also the imbalance between ReLU activation states.

3.2.1 Exploded Bounds at Initialization

For affine layer $\mathbf{h}_i = \mathbf{W}_i \mathbf{z}_{i-1} + \mathbf{b}_i$, the IBP bound computation is as follows:

$$\underline{\mathbf{h}}_i = \mathbf{W}_{i,+}\underline{\mathbf{z}}_{i-1} + \mathbf{W}_{i,-}\overline{\mathbf{z}}_{i-1} + \mathbf{b}_i, \ \ \overline{\mathbf{h}}_i = \mathbf{W}_{i,+}\overline{\mathbf{z}}_{i-1} + \mathbf{W}_{i,-}\underline{\mathbf{z}}_{i-1} + \mathbf{b}_i, \tag{3}$$

where $\mathbf{W}_{i,+}$ stands for retaining positive elements in $\mathbf{W}_i$, and vice versa for $\mathbf{W}_{i,-}$. Eq. (3) guarantees 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 check the tightness of the bounds:

$$\Delta_i = \overline{\mathbf{h}}_i - \underline{\mathbf{h}}_i = |\mathbf{W}_i|(\overline{\mathbf{z}}_{i-1} - \underline{\mathbf{z}}_{i-1}) = |\mathbf{W}_i|\delta_{i-1}, \tag{4}$$

where Δ_i denotes the difference 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. We assume each $\mathbf{W}_i$ is randomly initialized with each weight following a distribution with zero mean and variance σ_i^2. Then we view Δ_i as a random variable and use $\mathbb{E}(\Delta_i)$ to measure the expected tightness at layer i. As

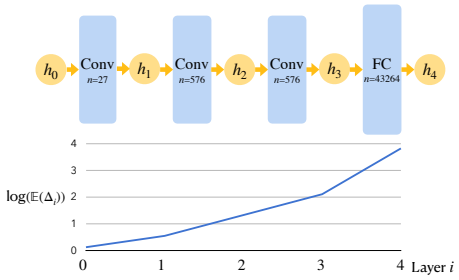

Figure 1: We show a simple untrained CNN (the classification layer is omitted) with Xavier initialization. We evaluate the mean of each layer's Δ_i as an estimation of $\mathbb{E}(\Delta_i)$ and plot $\log \mathbb{E}(\Delta_i)$. Interval bounds explode in deeper layers.

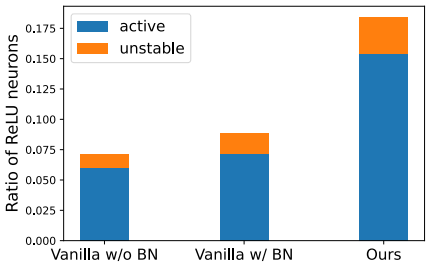

Figure 2: Ratios of active and unstable ReLU neurons for 7-layer CNN models on CIFAR-10 with different settings. The vanilla ones do not have regularization, and "vanilla (w/o BN)" does not use BN either.

$\mathbf{W}_i$ and δ_{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 D.1, we further have $\mathbb{E}(\delta_i) = \mathbb{E}(\text{ReLU}(\overline{\mathbf{h}}_i) - \text{ReLU}(\underline{\mathbf{h}}_i)) = \frac{1}{2}\mathbb{E}(\Delta_i)$, and

$$\mathbb{E}(\Delta_i) = \frac{n_i}{2}\mathbb{E}(|\mathbf{W}_i|)\mathbb{E}(\Delta_{i-1}). \tag{5}$$

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

$$\mathbb{E}(\Delta_i)/\mathbb{E}(\Delta_{i-1}) = \frac{n_i}{2}\mathbb{E}(|\mathbf{W}_i|). \tag{6}$$

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 [11], 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 are exploded at initialization, where the certified bounds become looser quickly after passing each layer. We illustrate the bound explosion in Figure 1, and in Appendix A, we list the difference gain of each initialization method in Table 5. As a result, long warmup schedules are adopted in previous works, to ease the training. Small perturbation radii are used in the early stage of the training to gradually make the model suitable for the target perturbation radius, but it is inefficient. Note that our analysis is also applicable to DNNs beyond feedforward networks, such as ResNet [16], where the input of an affine layer i can be any former layer $i'(i' < i)$ besides layer $i-1$. In the analysis above, without the loss of generality and for simplicity, we assume $i' = i - 1$.

3.2.2 Imbalanced ReLU Activation States

In this section, 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 ($\overline{\mathbf{h}}_{i,j} \leq 0$ for some neuron j in layer i), 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 6 in Appendix B, and thus the optimization tends to push the neurons to be inactive. We show this imbalance ReLU status in Figure 2 (vanilla w/o BN), and it is more severe when the warmup is shorter as shown in Appendix B. However, 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 [29] on standard training.

3.3 The Proposed Method

To address the aforementioned issues, we propose our method in three parts: 1) We derive a new weight initialization for IBP training to stabilize the tightness of bounds at initialization; 2) We

propose to fully add BN to mitigate imbalanced ReLU and stabilize the variance of bounds, while models in prior works either did not have BN for some or all the layers. 3) We further propose regularizations to stabilize the tightness and the balance of ReLU neuron states during warmup.

3.3.1 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 σ_i such that the *difference gain* defined in Eq. (6) 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.

3.3.2 Batch Normalization

Batch normalization (BN) [18] is originally designed to accelerate the convergence of standard training by reducing internal covariate shift. Specifically, BN normalizes the input of each layer to a distribution with stable mean and variance. In IBP training, BN can normalize the variance of bounds, and importantly, it can also improve the balance of ReLU activation states by shifting the center of upper and lower bounds to zero. In prior certified training works [13, 55, 50], 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 Wong et al. [47], Xu et al. [50] for certified training, where the shifting and scaling parameters are computed from unperturbed data.

3.3.3 Warmup Regularization

To further mitigate the two issues we identify for IBP training in Sec. 3.2, we add two regularizers to the warmup stage of IBP training, to explicitly stabilize the tightness of certified bounds and balance ReLU neuron states. The regularization is principled and motivated by the identified issues.

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

$$\mathcal{L}_{\text{tightness}} = \frac{1}{\tau m}\sum_{i=1}^{m} \text{ReLU}(\tau - \frac{\hat{\mathbb{E}}(\Delta_0)}{\hat{\mathbb{E}}(\Delta_i)}), \tag{7}$$

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

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 α_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 β_i to denote the ratio of contribution to $\text{Var}(\mathbf{c}_i)$. They are computed as:

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

where $\mathbf{h}_{i,j}, \underline{\mathbf{h}}_{i,j}, \overline{\mathbf{h}}_{i,j}$ stand for the value and bounds of each neuron in layer i, and in general $\alpha_i, \beta_i > 0$. We regard that balance is good if α_i and β_i are close to 1. With the same aforementioned

tolerance τ, 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:

$$\mathcal{L}_{\text{relu}} = \frac{1}{\tau m} \sum_{i=1}^{m} (\text{ReLU}(\tau - \min(\alpha_i, \frac{1}{\alpha_i})) + \text{ReLU}(\tau - \min(\beta_i, \frac{1}{\beta_i}))). \tag{8}$$

3.4 Training Objectives

Certified robust training solves the robust optimization problem as Eq. (1), and when the inner maximization is verifiably solved, the base training objective without regularization is:

$$\mathcal{L}_{\text{rob}} = \overline{L}(\mathbf{x}, y, \epsilon), \quad \text{where } \overline{L}(\mathbf{x}, y, \epsilon) \geq \max_{\delta \in \Delta(\mathbf{x})} L(f_\theta(\mathbf{x} + \delta), y), \tag{9}$$

such that $\overline{L}(\mathbf{x}, y, \epsilon)$ is an upper bound of $L(f_\theta(\mathbf{x} + \delta), y)$ given by robustness verification, e.g., IBP. In our proposed method, we firstly initialize the parameters with our IBP initialization, and then we perform a *short* warmup with gradually increasing ϵ ($0 \leq \epsilon \leq \epsilon_{\text{target}}$), where our training objective $\mathcal{L}$ combines the ordinary objective Eq. (9) and the proposed regularizers:

$$\mathcal{L} = \mathcal{L}_{\text{rob}} + \lambda(\mathcal{L}_{\text{tightness}} + \mathcal{L}_{\text{relu}}), \tag{10}$$

and λ 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 λ from λ_0 to 0 as ϵ 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 ϵ_{target}. Note that in the regularizers, the value of each $\text{ReLU}(\cdot)$ term has the same range $[0, \tau]$, and thus in Eq. (10) we directly sum up them without weighing them for simplicity. In testing, we use pure IBP bounds for robustness verification without any other tighter method.

4 Experimental Results

4.1 Settings

We adopt three datasets, MNIST [25], CIFAR-10 [21] and TinyImageNet [24]. Following Xu et al. [50], we consider three model architectures: a 7-layer feedforward convolutional network (CNN-7), Wide-ResNet [52] and ResNeXt [49]. According our discussion in Sec. 3.3.2, we also modify the models to fully add a BN after each convolutional and 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, and we provide results on other perturbation radii in Appendix B. We provide more implementation details in Appendix C. 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):

- Vanilla IBP [13] with existing initialization and no warmup regularizer. We use the default Xavier initialization in PyTorch to represent existing initializations for regular DNN training, and we find that orthogonal initialization originally used by [13] does not improve the performance.
- CROWN-IBP [55] with linear relaxation bounds by CROWN [54] during warmup. We use the generalized and accelerated version with loss fusion by Xu et al. [50], while the original version is $O(K)$ (K is the number of classes) more costly. During the warmup, it combines bounds by IBP and linear relaxation with weight $\frac{\epsilon}{\epsilon_{\text{target}}}$ and $(1 - \frac{\epsilon}{\epsilon_{\text{target}}})$ respectively.

4.2 Certified Robust Training with Short Warmup

We conduct certified robust training using relatively short warmup schedules to demonstrate the effectiveness of our proposed techniques for fast training, and we show our results in Table 1 for MNIST, CIFAR-10 and Table 2 for TinyImageNet. Compared to Vanilla IBP and CROWN-IBP, our improved IBP training with IBP initialization and warmup regularization 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 [50] tends to require a larger number of epochs to obtain good results and it sometimes underperform Vanilla IBP

Table 1: 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 phase (in the parentheses), $\epsilon = 0$, 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. We also report the result of our best run in "Ours (best)", since main results in prior works did not have repeats, and we include literature results for reference. Literatures with the "†" mark are concurrent preprint works.

Dataset	Schedule (epochs)	Method	CNN-7		Wide-ResNet		ResNeXt	
			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	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	$\mathbf{2.33 \pm 0.08}$	$\mathbf{11.03 \pm 0.13}$	$\mathbf{2.77 \pm 0.02}$	$\mathbf{11.76 \pm 0.07}$	$\mathbf{3.22 \pm 0.08}$	$\mathbf{13.43 \pm 0.17}$
		Ours (best)	**2.20**	**10.82**	2.75	11.69	3.17	13.20
	Literature results			Warmup		Total (epochs)	Standard	Verified
	Gowal et al. [13]			(2K+10K) steps		100	1.66	15.01 [a]
	Zhang et al. [55]			$(9 + 51)$ epochs		200	2.17	12.06
	†IBP+ParamRamp [30] [d]			$(9 + 51)$ epochs		200	2.16	10.88
	†CROWN-IBP+ParamRamp [30] [d]			$(9 + 51)$ 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	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	$\mathbf{56.64 \pm 0.48}$	$\mathbf{68.81 \pm 0.24}$	$\mathbf{56.74 \pm 0.40}$	$\mathbf{68.71 \pm 0.29}$	$\mathbf{59.33 \pm 0.86}$	$\mathbf{70.62 \pm 0.59}$
	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	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	$\mathbf{51.72 \pm 0.40}$	$\mathbf{65.58 \pm 0.32}$	$\mathbf{51.95 \pm 0.27}$	$\mathbf{65.91 \pm 0.14}$	$\mathbf{53.68 \pm 0.33}$	$\mathbf{66.91 \pm 0.40}$
		Ours (best)	**51.06**	**65.03**	51.63	65.72	53.38	66.41
	Literature results			Warmup		Total (epochs)	Standard	Verified
	Gowal et al. [13]			(5K+50K) steps		3,200	50.51	68.44 [b]
	Zhang et al. [55]			$(320 + 1600)$ epochs		3,200	54.02	66.94
	Balunovic & Vechev [2]			N/A [c]		800	48.3	72.5
	Xu et al. [50]			$(100 + 800)$ epochs		2,000	53.71	66.62
	†IBP+ParamRamp [30] [d]			$(320 + 1600)$ epochs		3,200	55.28	67.09
	†CROWN-IBP+ParamRamp [30] [d]			$(320 + 1600)$ epochs		3,200	51.94	65.08
	†ℓ_∞-dist net (other architecture) [53] [e]			N/A [e]		800	48.32	64.90

[a] Some test results in Gowal et al. [13] are obtained with costly mixed integer programming (MIP) and linear programming (LP); we take IBP verified errors for fair comparison following Zhang et al. [55].

[b] Additional PGD adversarial training was involved for this result, according to Zhang et al. [55].

[c] Balunovic & Vechev [2] used a different training scheme and train the network layer by layer.

[d] Concurrent Lyu et al. [30] 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.

[e] Concurrent Zhang et al. [53] use a very different model architecture with ℓ_∞ distance neurons rather than traditional DNNs, but still need a long schedule on both ϵ and ℓ_p norm where p is gradually increased until ∞.

Table 2: Standard and verified error rates (%) of models trained on TinyImageNet ($\epsilon_t = 1/255$). The best result in literature [50] is standard error 72.18% and verified error 84.14% using 800 epochs.

Model	Schedule (epochs)	Vanilla IBP w/o BN		Vanilla IBP		CROWN-IBP		Ours	
		Standard	Verified	Standard	Verified	Standard	Verified	Standard	Verified
CNN-7	80 (1+10+69)	80.28	86.59	75.50	82.92	76.00	82.81	75.20	**82.45**
	80 (1+20+59)	79.35	86.06	74.68	82.84	76.27	83.35	74.29	**82.36**
Wide-Resnet [a]	80 (1+10+69)	79.26	85.40	75.89	83.00	75.85	83.65	74.90	**82.49**
	80 (1+20+59)	78.45	85.19	75.65	83.17	75.95	83.08	74.59	**82.75**
ResNext	80(1+10+69)	83.27	88.14	82.39	87.15	85.47	89.11	80.20	**85.77**
	80 (1+20+59)	82.04	87.88	81.72	87.10	80.81	86.43	78.91	**85.78**

[a] The Wide-ResNet model used here is 5 times smaller than the one used in [50] to save cost.

under short schedules, but disabling loss fusion can make it much more costly and unscalable [50]. In terms of the best results, we achieve verified error 10.82% on MNIST $\epsilon = 0.4$, 65.03% on CIFAR-10 $\epsilon = 8/255$, and 82.36% on TinyImageNet $\epsilon_{\text{target}} = 1/255$, which significantly outperform literature SOTA [13, 50]. Compared to concurrent preprint works [30, 53] which use different improvement techniques, we have comparable verified errors, but they still need long training schedules, and thus our method is much faster (see Sec. 4.3). For a reference, we tried the code of concurrent Zhang et al. [53] which used a different network architecture with "ℓ_∞ distance neurons" rather than standard 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.

4.3 Comparison on Training Cost

Table 3: Comparison of estimated time cost (seconds), for CNN-7 on CIFAR-10. For short warmup, we use report the per-epoch time during training phases with different ϵ ranges, and the total time under the $1 + 80$ warmup schedule with 160 total epochs. We also include the total training cost of literature works using long schedules, where literatures with the "†" mark are concurrent works.

Method		Epochs	Per-epoch for ϵ			Total
			0	$(0, \epsilon_{\text{target}})$	ϵ_{target}	
Literatures	IBP [13]	3200	-			40496×4 [a]
	CROWN-IBP (w/o loss fusion) [55]	3200				91288×4 [a]
	CROWN-IBP [50]	2000				52362×4 [a]
	†IBP+ParamRamp [30]	3200	-			$40496 \times 4 \times 1.09$ [b]
	†CROWN-IBP+ParamRamp [30]	3200				$91288 \times 4 \times 1.51$ [b]
Short Warmup	Vanilla IBP	160	30.0	54.8	54.8	8747.9
	CROWN-IBP	160	30.0	78.5	54.8	10641.3
	Ours	160	64.0	64.0	54.8	9512.3

[a] 4 GPUs were used in [55, 50]. Their models did not have BN or missed some BNs.
[b] 1.09 and 1.51 are from the overhead over IBP or CROWN-IBP reported in concurrent Lyu et al. [30].

We compare the training cost of different methods. We use a single Nvidia RTX 2080 Ti GPU, and for Vanilla, CROWN-IBP and our method using short schedules, we measure the per-epoch time during three phases, namely $\epsilon = 0$, $0 < \epsilon < \epsilon_{\text{target}}$, and $\epsilon = \epsilon_{\text{target}}$, and we then estimate the total training time according to the schedule. We use gradient accumulation wherever needed to fit each method into the memory of a single GPU. We also compare with total time cost with literature methods using long schedules. We show results of CNN-7 for CIFAR-10 in Table 3, and other settings in Appendix B. For $\epsilon = 0$, Vanilla IBP and CROWN-IBP use regular training while we compute IBP bounds for regularization and have a small overhead, but this phase is extremely short (no more than 1 epoch here) and thus negligible. For $0 < \epsilon < \epsilon_{\text{target}}$, our method has a smaller overhead on regularizers compared to Vanilla IBP, while CROWN-IBP using linear relaxation can be more costly. In $\epsilon = \epsilon_{\text{target}}$, all the three methods use the same pure IBP.

For total time on CIFAR-10 with the same 160-epoch schedule, we have a small overhead of around $9\% \sim 13\%$ compared to Vanilla IBP but the cost is still around $12\% \sim 23\%$ lower than CROWN-IBP, while we achieve lower verified errors than the baselines under such short warmup schedules (see Table 1). And importantly, compared to literatures using long training schedules, we significantly reduce the number of training epochs and the total training time (e.g., Xu et al. [50] is around $20\times$ more costly than ours in total).

4.4 Ablation Study and Discussions

Table 4: Ablation study results. We use the CNN-7 model on CIFAR-10. "BN-Conv" stands for BN layers after each convolutional layer, and "BN-FC" stands for BN layers after the hidden fully-connected layer. "✓" means that the component is enabled, and "×" means that the component is disabled. We repeat each setting for 5 times and report the mean and standard deviation.

BN-Conv	BN-FC	IBP Initialization	$\mathcal{L}_{\text{tightness}}$	$\mathcal{L}_{\text{relu}}$	70 (1+20+49)		160 (1+80+79)	
					Standard (%)	Verified (%)	Standard (%)	Verified (%)
×	×	×	×	×	59.33±0.70	70.18±0.18	57.08±0.29	69.43±0.28
✓	×	×	×	×	61.95±0.80	71.12±0.42	57.21±0.65	69.21±0.30
✓	✓	×	×	×	58.72±0.27	69.88±0.10	53.80±0.71	67.01±0.29
✓	✓	✓	×	×	58.93±0.29	69.60±0.35	54.59±0.64	67.63±0.34
✓	✓	✓	✓	×	56.76±0.38	68.96±0.49	53.08±0.26	66.74±0.20
✓	✓	✓	×	✓	58.49±0.42	69.38±0.23	53.29±0.76	66.46±0.44
✓	✓	×	✓	✓	58.79±0.40	69.29±0.28	52.45±0.34	66.34±0.38
✓	✓	✓	✓	✓	**56.64±0.48**	**68.81±0.24**	**51.72±0.40**	**65.58±0.32**

In this section, we empirically verify whether each part of our modification contribute to the improvement and whether they behave as we expect. We first conduct an ablation study and we also plot the curve of the regularization terms to reflect the bound tightness and ReLU balance in different settings.

In the ablation study, we use CIFAR-10 with the currently best CNN-7 model under the "$1 + 20$" and "$1 + 80$" warmup schedules as used in Table 1. We start from a vanilla setting, and we add BN, IBP initialization, and the warmup regularizers to the model or training. We report the results in Table 4.

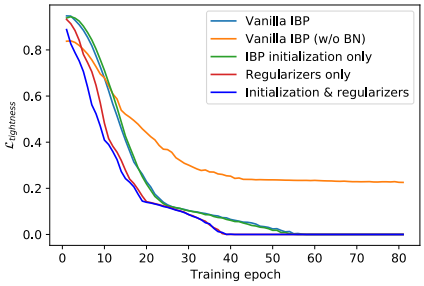

Figure 3: $\mathcal{L}_{\text{tightness}}$ during warmup. $\mathcal{L}_{\text{tightness}}$ is optimized only for "regularizers only" and "initialization & regularizers" setting, and BN is fully added except for "Vanilla IBP (w/o BN)".

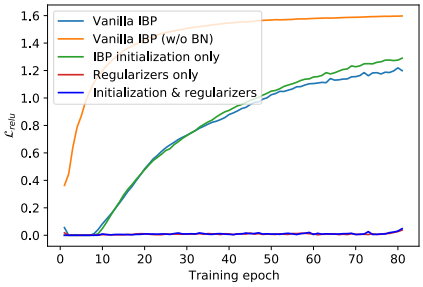

Figure 4: $\mathcal{L}_{\text{relu}}$ during warmup, with same settings as in Figure 3.

The first three rows show that fully adding BN improves the training when vanilla IBP is used, and it is important to add BN for the fully-connected layer, which was missed in prior works. Based on the improved model structure, adding both IBP initialization and warmup regularization further improves the performance, and removing either of these parts leads to a degraded performance.

We notice that adding IBP initialization alone may not necessarily bring improvement to the verified error. A factor is that IBP initialization can reduce the variance of the outputs, as discussed in Appendix D.2, and it may harm the training during the early warmup when ϵ is small and certified training is close to standard training. Also, the effect of initialization can be weakened during the warmup when ϵ is much smaller than ϵ_{target}. But when we combine it with the regularizers, the regularization can continue to tighten the bounds, and the IBP initialization can benefit the optimization for the tightness regularizer. Nevertheless, IBP initialization is more beneficial for deep models where the exploded bound issue is more severe. In Figure 5, we show that for a ResNeXt on TinyImageNet, IBP initialization is helpful for reaching lower verified errors especially at early epochs.

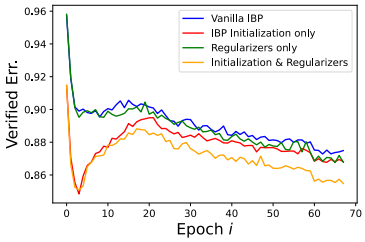

Figure 5: Curve of training verified error of a ResNeXt on TinyImageNet. Note that the verified errors can increase during the warmup as ϵ increases.

In Figure 3, we plot the $\mathcal{L}_{\text{tightness}}$ during training for different settings. Note that for the settings without regularizers, we only plot the loss terms but not optimize them during training. By using the regularization in training, $\mathcal{L}_{\text{tightness}}$ descends faster, and further adding the IBP initialization leads to even faster descent during the early epochs. In Figure 4, we show that the $\mathcal{L}_{\text{relu}}$ term is indeed under control with our regularizer added in training, which indicates the ReLU activation states is more balanced during training, while $\mathcal{L}_{\text{relu}}$ could gradually grow larger when the regularization is not added in training. Notably, when BN is removed and the regularization term is not optimized (Vanilla IBP (w/o BN)), $\mathcal{L}_{\text{relu}}$ becomes extremely large in later epochs, and $\mathcal{L}_{\text{tightness}}$ is also large in the end, which suggests the training is hampered.

5 Conclusion

In this paper, we identify two issues in existing certified robust training methods regarding exploded bounds and imbalanced ReLU neuron states. To address these issues based on IBP training, we propose an IBP initialization and warmup regularization, and we also identify the benefit of fully adding BN. With our improvements, we demonstrate that we are able to achieve better verified errors using much shorter warmup and training schedules compared to literatures under the same convolution-based network architecture, for fast certified robust training.

References

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mane, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viegas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. Tensorflow: Large-scale machine learning on heterogeneous distributed systems, 2016.

[2] Balunovic, M. and Vechev, M. Adversarial training and provable defenses: Bridging the gap. In *International Conference on Learning Representations*, 2020.

[3] Bhattacharya, A. Learnable weight initialization in neural networks. 2020.

[4] Blum, A., Dick, T., Manoj, N., and Zhang, H. Random smoothing might be unable to certify ℓ_∞ robustness for high-dimensional images. *Journal of Machine Learning Research*, 21:1–21, 2020.

[5] Bunel, R., Turkaslan, I., Torr, P. H. S., Kohli, P., and Kumar, M. P. Piecewise linear neural network verification: A comparative study. *CoRR*, abs/1711.00455, 2017. URL http://arxiv.org/abs/1711.00455.

[6] Carlini, N. and Wagner, D. 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.

[7] Chen, H., Zhang, H., Chen, P.-Y., Yi, J., and Hsieh, C.-J. Attacking visual language grounding with adversarial examples: A case study on neural image captioning. *arXiv preprint arXiv:1712.02051*, 2017.

[8] Choi, J.-H., Zhang, H., Kim, J.-H., Hsieh, C.-J., and Lee, J.-S. 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.

[9] Cohen, J. M., Rosenfeld, E., and Kolter, J. Z. Certified adversarial robustness via randomized smoothing. In *ICML*, 2019.

[10] Dvijotham, K., Gowal, S., Stanforth, R., Arandjelovic, R., O'Donoghue, B., Uesato, J., and Kohli, P. Training verified learners with learned verifiers. *arXiv preprint arXiv:1805.10265*, 2018.

[11] Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Teh, Y. W. and Titterington, M. (eds.), *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. URL http://proceedings.mlr.press/v9/glorot10a.html.

[12] Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

[13] Gowal, S., Dvijotham, K., Stanforth, R., Bunel, R., Qin, C., Uesato, J., Mann, T., and Kohli, P. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.

[14] He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[15] He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

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

[17] Huang, X. S., Perez, F., Ba, J., and Volkovs, M. Improving transformer optimization through better initialization. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 4475–4483. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/huang20f.html.

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

[19] Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pp. 97–117. Springer, 2017.

[20] Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. *Technical Report TR-2009*, 2009.

[22] Kumar, A., Levine, A., Goldstein, T., and Feizi, S. Curse of dimensionality on randomized smoothing for certifiable robustness. In *International Conference on Machine Learning*, pp. 5458–5467. PMLR, 2020.

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

[24] Le, Y. and Yang, X. Tiny imagenet visual recognition challenge. *CS 231N*, 2015.

[25] LeCun, Y., Cortes, C., and Burges, C. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[26] Lecuyer, M., Atlidakis, V., Geambasu, R., Hsu, D., and Jana, S. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 656–672. IEEE, 2019.

[27] Lee, S., Lee, W., Park, J., and Lee, J. Loss landscape matters: Training certifiably robust models with favorable loss landscape. 2021. URL https://openreview.net/forum?id=lvXLfNeCQdK.

[28] Li, B., Chen, C., Wang, W., and Carin, L. Certified adversarial robustness with additive noise. In *Advances in Neural Information Processing Systems*, pp. 9464–9474, 2019.

[29] Lu, L., Shin, Y., Su, Y., and Karniadakis, G. E. Dying relu and initialization: Theory and numerical examples. *arXiv preprint arXiv:1903.06733*, 2019.

[30] Lyu, Z., Guo, M., Wu, T., Xu, G., Zhang, K., and Lin, D. Towards evaluating and training verifiably robust neural networks, 2021.

[31] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

[32] Mirman, M., Gehr, T., and Vechev, M. Differentiable abstract interpretation for provably robust neural networks. In *International Conference on Machine Learning*, pp. 3575–3583, 2018.

[33] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

[34] Raghunathan, A., Steinhardt, J., and Liang, P. Certified defenses against adversarial examples. *International Conference on Learning Representations (ICLR), arXiv preprint arXiv:1801.09344*, 2018.

[35] Raghunathan, A., Steinhardt, J., and Liang, P. S. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems*, pp. 10877–10887, 2018.

[36] Salman, H., Li, J., Razenshteyn, I., Zhang, P., Zhang, H., Bubeck, S., and Yang, G. Provably robust deep learning via adversarially trained smoothed classifiers. In *Advances in Neural Information Processing Systems*, pp. 11289–11300, 2019.

[37] Saxe, A. M., McClelland, J. L., and Ganguli, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[38] Singh, G., Gehr, T., Mirman, M., Püschel, M., and Vechev, M. Fast and effective robustness certification. In *Advances in Neural Information Processing Systems*, pp. 10825–10836, 2018.

[39] Singh, G., Gehr, T., Püschel, M., and Vechev, M. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL):41, 2019.

[40] Su, D., Zhang, H., Chen, H., Yi, J., Chen, P.-Y., and Gao, Y. 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.

[41] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *ICLR*, 2013.

[42] Taki, M. Deep residual networks and weight initialization. *CoRR*, abs/1709.02956, 2017. URL http://arxiv.org/abs/1709.02956.

[43] Wang, S., Chen, Y., Abdou, A., and Jana, S. Mixtrain: Scalable training of formally robust neural networks. *arXiv preprint arXiv:1811.02625*, 2018.

[44] Wang, S., Pei, K., Whitehouse, J., Yang, J., and Jana, S. Efficient formal safety analysis of neural networks. In *Advances in Neural Information Processing Systems*, pp. 6367–6377, 2018.

[45] Wang, S., Zhang, H., Xu, K., Lin, X., Jana, S., Hsieh, C.-J., and Kolter, J. Z. Beta-crown: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification. *arXiv preprint arXiv:2103.06624*, 2021.

[46] Wong, E. and Kolter, Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pp. 5283–5292, 2018.

[47] Wong, E., Schmidt, F., Metzen, J. H., and Kolter, J. Z. Scaling provable adversarial defenses. In *NIPS*, 2018.

[48] Xiao, K. Y., Tjeng, V., Shafiullah, N. M., and Madry, A. Training for faster adversarial robustness verification via inducing relu stability. In *ICLR*, 2019.

[49] Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

[50] Xu, K., Shi, Z., Zhang, H., Wang, Y., Chang, K.-W., Huang, M., Kailkhura, B., Lin, X., and Hsieh, C.-J. Automatic perturbation analysis for scalable certified robustness and beyond. *Advances in Neural Information Processing Systems*, 33, 2020.

[51] Yang, G., Duan, T., Hu, J. E., Salman, H., Razenshteyn, I., and Li, J. Randomized smoothing of all shapes and sizes. In *International Conference on Machine Learning*, pp. 10693–10705. PMLR, 2020.

[52] Zagoruyko, S. and Komodakis, N. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[53] Zhang, B., Cai, T., Lu, Z., He, D., and Wang, L. Towards certifying ℓ_∞ robustness using neural networks with ℓ_∞-dist neurons. *arXiv preprint arXiv:2102.05363*, 2021.

[54] Zhang, H., Weng, T.-W., Chen, P.-Y., Hsieh, C.-J., and Daniel, L. Efficient neural network robustness certification with general activation functions. In *Advances in neural information processing systems*, pp. 4939–4948, 2018.

[55] Zhang, H., Chen, H., Xiao, C., Li, B., Boning, D., and Hsieh, C.-J. Towards stable and efficient training of verifiably robust neural networks. In *International Conference on Learning Representations*, 2020.

[56] Zhu, C., Ni, R., Xu, Z., Kong, K., Huang, W. R., and Goldstein, T. Gradinit: Learning to initialize neural networks for stable and efficient training. *arXiv preprint arXiv:2102.08098*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We have discussions in Section 4.4.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] We believe there is no negative societal impact.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] Additional proofs are in Appendix D.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We submit our code as a supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4 and Appendix C.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Table 1.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Sec. 4.3 and Appendix C.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Sec. 4.1 and Appendix C.
   (b) Did you mention the license of the assets? [N/A] The datasets we use are all widely adopted benchmark datasets.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] We do not have crowdsourcing experiments or human evaluations.

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]