Towards Stable and Efficient Training of Verifiably Robust Neural Networks

Huan Zhang[1] **Hongge Chen**[2] **Chaowei Xiao**[3] **Bo Li**[4] **Duane Boning**[2] **Cho-Jui Hsieh**[1]
[1]Department of Computer Science, UCLA
[2]Department of EECS, MIT
[3] Department of Computer Science, University of Michigan
[4] Department of Computer Science, UIUC
huan@huan-zhang.com, chenhg@mit.edu, xiaocw@umich.edu
lbo@illinois.edu, boning@mtl.mit.edu, chohsieh@cs.ucla.edu

Abstract

Training neural networks with verifiable robustness guarantees is challenging. Several existing successful approaches utilize relatively tight linear relaxation based bounds of neural network outputs, but they can slow down training by a factor of hundreds and over-regularize the network. Meanwhile, interval bound propagation (IBP) based training is efficient and significantly outperform linear relaxation based methods on some tasks, yet it suffers from stability issues since the bounds are much looser. In this paper, we first interpret IBP training as training an augmented network which computes *non-linear* bounds, thus explaining its good performance. We then propose a new certified adversarial training method, **CROWN-IBP**, by combining the fast IBP bounds in the forward pass and a tight linear relaxation based bound, CROWN, in the backward pass. The proposed method is computationally efficient and consistently outperforms IBP baselines on training verifiably robust neural networks. We conduct large scale experiments using 53 models on MNIST, Fashion-MNIST and CIFAR datasets. On MNIST with $\epsilon = 0.3$ and $\epsilon = 0.4$ (ℓ_∞ norm distortion) we achieve **7.46%** and **12.96%** verified error on test set, respectively, outperforming previous certified defense methods[1].

1 Introduction

The success of deep learning networks has motivated their deployment in some safety-critical environments, such as autonomous driving and facial recognition systems. Applications in these areas make understanding the robustness and security of deep neural networks urgently needed, especially their resilience under malicious, finely crafted inputs. Unfortunately, deep learning models' performance are often so brittle that even imperceptibly modified inputs, also known as adversarial examples, are able to completely break the model (Goodfellow et al., 2015; Szegedy et al., 2013). Deep learning models' robustness under adversarial examples is well-studied from both attack (crafting powerful adversarial examples) and defence (making the model more robust) perspectives (Athalye et al., 2018; Carlini & Wagner, 2017a,b; Goodfellow et al., 2015; Madry et al., 2018; Papernot et al., 2016). Recently, it has been shown that defending deep learning models against adversarial examples is a very difficult task, especially under strong and adaptive attacks. Early defenses such as distillation (Papernot et al., 2016) have been broken by stronger attacks like C&W (Carlini & Wagner, 2017b). Many defense methods have been proposed recently (Guo et al., 2018; Song et al., 2017; Buckman et al., 2018; Ma et al., 2018; Samangouei et al., 2018), but their robustness improvement cannot be *certified* – no provable guarantees can be given to verify their robustness. In fact, most of these uncertified defenses are actually vulnerable under stronger attacks (Athalye et al., 2018; He et al., 2017).

There are, however, some methods in the literature seeking to give *provable* guarantees on the robustness performance, such as distributional robust optimization (Sinha et al., 2018), linear relaxations (Wong & Kolter, 2018; Mirman et al., 2018; Wang et al., 2018a; Dvijotham et al., 2018b; Weng et al., 2018; Zhang et al., 2018), interval bound propagation (Gowal et al., 2018), ReLU stability regularization (Xiao et al., 2019), and semidefinite relaxations (Raghunathan et al., 2018a). Linear relaxations of neural networks, first proposed by

[1]Source code of CROWN-IBP and IBP baselines is available at https://github.com/huanzhang12/CROWN-IBP

Wong & Kolter (2018), is one of the most popular categories among these certified defences. They use the dual of linear programming or several similar approaches to provide a linear relaxation of the network (referred to as a "convex adversarial polytope") and the resulting bounds are tractable for robust optimization. However, these methods are both computationally and memory intensive, and can increase model training time by a factor of hundreds. On the other hand, interval bound propagation (IBP) is a simple and efficient method that can also be used for training verifiable neural networks (Gowal et al., 2018), which achieved state-of-the-art verified error on many datasets despite its simplicity. However, since the IBP bounds are very loose during the initial phase of training, the training procedure can be unstable and sensitive to hyperparameters.

In this paper, we first investigate the limitation of existing linear relaxation based certified robust training methods and find that they over-penalize the L_∞ induced norm of weight matrices, due to their nature of being linear. On the other hand, we interpret IBP as an *augmented* neural network, which learns to optimize a *non-linear* bound. This explains the failure of linear relaxation based methods and the success of IBP training.

To improve the stability of IBP network, we propose a new certified robust training method, CROWN-IBP, which combines the power and efficiency of IBP with a tight linear relaxation based verification bound, CROWN. In our experiments, we show that CROWN-IBP significantly improves the training stability, and further reduces verified errors by a noticeable margin comparing to the existing IBP based approach (Gowal et al., 2018). On MNIST, we reach 7.46% and 12.96% verified error under ℓ_∞ distortions with $\epsilon = 0.3$ and $\epsilon = 0.4$, respectively, outperforming carefully tuned models in (Gowal et al., 2018), and significantly outperforming linear relaxation based methods (at $\epsilon = 0.3$ Wong et al. (2018) report over 40% verified error).

2 Related Work and Background

2.1 Robustness Verification and Relaxations of Neural Networks

Neural network robustness verification algorithms seek for upper and lower bounds of an output neuron for all possible inputs within a set S, typically a norm bounded perturbation. Most importantly, the margins of the outputs between the ground-truth class and any other classes determine model robustness. However, it has already been shown that finding the exact output range is a non-convex problem and NP-complete (Katz et al., 2017; Weng et al., 2018). Therefore, recent works resorted to giving relatively tight but computationally tractable bounds of the output range with necessary relaxations of the original problem. Many of these robustness verification approaches are based on linear relaxations of non-linear units in neural networks, including CROWN (Zhang et al., 2018), DeepPoly (Singh et al., 2019), Fast-Lin (Weng et al., 2018), DeepZ (Singh et al., 2018) and Neurify (Wang et al., 2018b). We refer the readers to (Salman et al., 2019) for a comprehensive survey on this topic. After linear relaxation, they essentially bound the output of a neural network $f(\cdot)$ by linear upper/lower hyperplanes:

$$\mathbf{A}\Delta\boldsymbol{x} + b_L \leq f(\boldsymbol{x}_0 + \Delta\boldsymbol{x}) \leq \mathbf{A}\Delta\boldsymbol{x} + b_U \tag{1}$$

where $\mathbf{A} = \mathbf{W}^{(L)}\mathbf{D}^{(L-1)}\mathbf{W}^{(L-1)}\cdots\mathbf{D}^{(1)}\mathbf{W}^{(1)}$ is the product of the network weight matrices $\mathbf{W}^{(l)}$ and diagonal matrices $\mathbf{D}^{(l)}$ reflecting the ReLU relaxations; b_L and b_U are two bias terms unrelated to Δx. Additionally, Dvijotham et al. (2018c,a); Qin et al. (2019) solve the Lagrangian dual of verification problem; Raghunathan et al. (2018a,b) propose semidefinite relaxations which are tighter compared to linear relaxation based methods, but computationally expensive. Bounds on neural network local Lipschitz constant can also be used for verification (Hein & Andriushchenko, 2017; Zhang et al., 2019). Besides these deterministic verification approaches, random smoothing can be used to increase the robustness of any model in a probabilistic manner (Cohen et al., 2019; Lecuyer et al., 2018; Li et al., 2018; Liu et al., 2018).

2.2 Robust Optimization and Verifiable Adversarial Defense

To improve the robustness of neural networks against adversarial perturbations, a natural idea is to generate adversarial examples by attacking itself and then use them to augment the training set (Kurakin et al., 2017). More recently, Madry et al. (2018) showed that adversarial training can be formulated as solving a min-max robust optimization problem as in (2). Given a model with parameter θ, loss function L, and training data distribution $\mathcal{X}$, the training algorithm aims to minimize the robust loss, which is defined as the max loss within a neighborhood $\{x + \delta | \delta \in S\}$ of each data point x, leading to the following robust optimization problem:

$$\min_{\theta} \underset{(x,y)\in\mathcal{X}}{E} \left[\max_{\delta\in S} L(x+\delta; y; \theta) \right]. \tag{2}$$

To solve this minimax problem, Madry et al. (2018) proposed to use projected gradient descent (PGD) attack to approximately solve the inner max and then use the loss on the perturbed example to update the model.

Networks trained by this procedure achieve state-of-the-art test accuracy under strong attacks (Athalye et al., 2018; Wang et al., 2018a; Zheng et al., 2018).

Despite being robust under strong attacks, models obtained by this PGD-based adversarial training do not have verified error guarantees. Due to the nonconvexity of neural networks, PGD attack can only compute the *lower bound* of robust loss (the inner maximization problem). Minimizing a lower bound of the inner max cannot guarantee (2) is minimized. In other words, even if PGD-attack cannot find a perturbation with large verified error, that does not mean there exists no such perturbation. This becomes problematic in safety-critical applications since those models need to be *provably* safe.

Verifiable adversarial training methods, on the other hand, aim to obtain a network with good robustness that can be verified. This can be done by combining adversarial training and robustness verification—instead of using PGD to find a lower bound of inner max, certified adversarial training uses a verification method to find an *upper bound* of the inner max, and then update the parameters based on this upper bound of robust loss. Minimizing an upper bound of the inner max guarantees to minimize the robust loss. There are two certified robust training methods that are related to our work and we will describe them in detail below.

2.2.1 Linear Relaxation Based Verifiable Adversarial Training

One of the most popular verifiable adversarial training method was proposed in (Wong & Kolter, 2018) using linear relaxations of neural networks to give an upper bound of the inner max. Other similar approaches include Mirman et al. (2018); Wang et al. (2018a); Dvijotham et al. (2018b). Since the bound propagation process of a convex adversarial polytope is too expensive, several methods were proposed to improve its efficiency, like Cauchy projection (Wong et al., 2018) and dynamic mixed training (Wang et al., 2018a). However, even with these speed-ups, the training process is still slow. Also, this method may significantly reduce the model's standard accuracy (accuracy on natural, unmodified test set). As will be discussed in our experiments in Section 4, we show that this method tends to over-regularize the network during training. Intuitively, regularizing the linear relaxation of the network results in regularizing the ℓ_1 norm of each row. Since they train the network to make this bound tight, an implicit regularization was added to the ℓ_∞ induced norm of weight matrices.

2.2.2 Interval Bound Propagation (IBP)

Interval Bound Propagation (IBP) uses a very simple rule to compute the pre-activation outer bounds for each layer of the neural network. Unlike linear relaxation based methods, IBP does not relax ReLU neurons and does not consider the correlations between different layer weights and treat each layer individually. Gowal et al. (2018) presented a verifiably robust training method by using IBP to give output bounds. The motivation of (Gowal et al., 2018) is to speed up the training process of verifiably robust models.

However, IBP can be unstable to use in practice, since the bounds can be very loose especially during the initial phase of training, which poses a challenging problem to the optimizer. To help with instability, Gowal et al. (2018) use a mixture of regular cross entropy loss and robust loss as the model's training loss, controlled by a parameter κ; it can be tricky to balance the two losses. Due to the difficulty in parameter tuning, reproducing the results in (Gowal et al., 2018) is believed to be hard [2]. Achieving the best CIFAR results reported in (Gowal et al., 2018) requires training for 3200 epochs with a batch size of 1600 on 32 TPUs. In our proposed method, we avoid the mixture of regular cross entropy and robust loss, and thus there is no need to control the additional parameter κ.

3 Methodology

3.1 Background

We first give notations used throughout the paper, and backgrounds on verification and robust optimization.

Notation. We define an L-layer neural network recursively as:

$$f(\boldsymbol{x}) = z^{(L)} \qquad z^{(l)} = \mathbf{W}^{(l)} h^{(l-1)} + \boldsymbol{b}^{(l)} \qquad \mathbf{W}^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}} \qquad \boldsymbol{b}^{(l)} \in \mathbb{R}^{n_l}$$
$$h^{(l)} = \sigma^{(l)}(z^{(l)}), \quad \forall l \in \{1, \ldots, L-1\},$$

[2] https://github.com/deepmind/interval-bound-propagation/issues/1

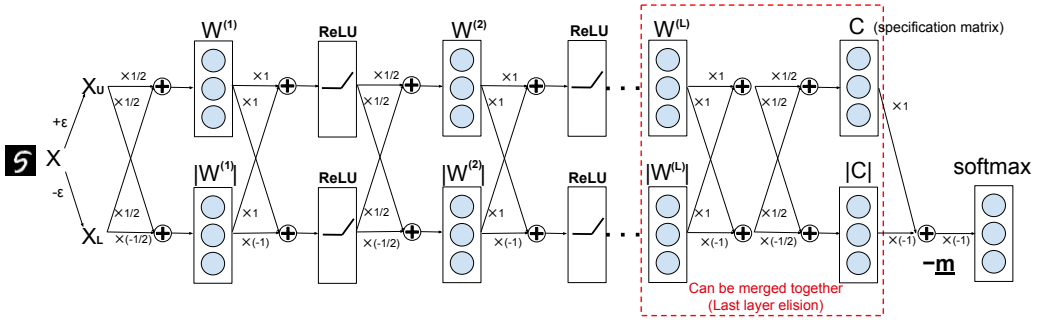


Figure 1: Interval Bound Propagation viewed as training an *augmented neural network* (IBP-NN). The inputs of IBP-NN are two images $\boldsymbol{x}_k + \epsilon$ and $\boldsymbol{x}_k - \epsilon$. The output of IBP-NN is a vector of lower bounds of margins (denoted as $\underline{\boldsymbol{m}}$) between ground-truth class and all classes (including the ground-truth class itself) for all $\boldsymbol{x}_k - \epsilon \leq \boldsymbol{x} \leq \boldsymbol{x}_k + \epsilon$. This vector $\underline{\boldsymbol{m}}$ is negated and sent into a regular softmax function to get model prediction. The top-1 prediction of softmax is correct if and only if all margins between the ground-truth class and other classes (except the ground truth class) are positive, i.e., the model is verifiably robust. Thus, an IBP-NN with low standard error guarantees low verified error on the original network.

where $h^{(0)}(\boldsymbol{x}) = \boldsymbol{x}$, n_0 represents input dimension and n_L is the number of classes, σ is an element-wise activation function. We use z to represent pre-activation neuron values and h to represent post-activation neuron values. Consider an input example $\boldsymbol{x}_k$ with ground-truth label y_k, we consider a set of $S(\boldsymbol{x}_k, \epsilon) = \{\boldsymbol{x} | \|\boldsymbol{x} - \boldsymbol{x}_k\|_\infty \leq \epsilon\}$ and we desire a robust network to have the property $y_k = \text{argmax}_j [f(\boldsymbol{x})]_j$ for all $\boldsymbol{x} \in S$. We define element-wise upper and lower bounds for $z^{(l)}$ and $h^{(l)}$ as $\underline{z}^{(l)} \leq z^{(l)} \leq \overline{z}^{(l)}$ and $\underline{h}^{(l)} \leq h^{(l)} \leq \overline{h}^{(l)}$.

Verification Specifications. Neural network verification literature typically defines a specification vector $\boldsymbol{c} \in \mathbb{R}^{n_L}$, that gives a linear combination for neural network output: $\boldsymbol{c}^\top f(\boldsymbol{x})$. In robustness verification, typically we set $\boldsymbol{c}_i = 1$ where i is the ground truth class label, $\boldsymbol{c}_j = -1$ where j is the attack target label and other elements in c are 0. This will represent the margin between class i and class j. For an n_L class classifier and a given label y, we define a specification matrix $C \in \mathbb{R}^{n_L \times n_L}$ as:

$$C_{i,j} = \begin{cases} 1, & \text{if } j = y, i \neq y \text{ (output of ground truth class)} \\ -1, & \text{if } i = j, i \neq y \text{ (output of other classes, negated)} \\ 0, & \text{otherwise (note that the } y\text{-th row contains all 0)} \end{cases} \tag{3}$$

Importantly, each element in vector $\boldsymbol{m} := Cf(\boldsymbol{x}) \in \mathbb{R}^{n_L}$ gives us margins between class y and all other classes. We define the lower bound of $Cf(\boldsymbol{x})$ for all $\boldsymbol{x} \in S(\boldsymbol{x}_k, \epsilon)$ as $\underline{\boldsymbol{m}}(\boldsymbol{x}_k, \epsilon)$, which is a very important quantity. Wong & Kolter (2018) showed that for cross-entropy loss we have:

$$\max_{\boldsymbol{x} \in S(\boldsymbol{x}_k, \epsilon)} L(f(\boldsymbol{x}); y; \theta) \leq L(f(-\underline{\boldsymbol{m}}(\boldsymbol{x}_k, \epsilon)); y; \theta). \tag{4}$$

(4) gives us the opportunity to solve the robust optimization problem (2) via minimizing the tractable upper bound of inner-max. Minimizing the upper bound guarantees that $\max_{\boldsymbol{x} \in S(\boldsymbol{x}_k, \epsilon)} L(f(\boldsymbol{x}), y)$ is also minimized.

3.2 Analysis of IBP and Linear Relaxation based Verifiable Training Methods

Interval Bound Propagation (IBP) Interval Bound Propagation (IBP) uses a very simple bound propagation rule. For input layer we have $\boldsymbol{x}_L \leq \boldsymbol{x} \leq \boldsymbol{x}_U$ (element-wise). For an affine layer we have

$$\overline{z}^{(l)} = \mathbf{W}^{(l)} \frac{\overline{h}^{(l-1)} + \underline{h}^{(l-1)}}{2} + |\mathbf{W}^{(l)}| \frac{\overline{h}^{(l-1)} - \underline{h}^{(l-1)}}{2} \tag{5}$$

$$\underline{z}^{(l)} = \mathbf{W}^{(l)} \frac{\overline{h}^{(l-1)} + \underline{h}^{(l-1)}}{2} - |\mathbf{W}^{(l)}| \frac{\overline{h}^{(l-1)} - \underline{h}^{(l-1)}}{2} \tag{6}$$

where $|\mathbf{W}^{(l)}|$ takes element-wise absolute value. Note that $\overline{h}^{(0)} = \boldsymbol{x}_U$ and $\underline{h}^{(0)} = \boldsymbol{x}_L$. And for element-wise monotonic increasing activation functions σ,

$$\overline{h}^{(l)} = \sigma(\overline{z}^{(l)}) \qquad \underline{h}^{(l)} = \sigma(\underline{z}^{(l)}). \tag{7}$$

Dataset	ϵ (ℓ_∞ norm)	IBP verified error	Convex-adv verified error
MNIST	0.1	5.83%	8.90%
	0.2	7.37%	45.37%
	0.3	10.68%	97.77%
	0.4	16.76%	99.98%
Fashion-MNIST	0.1	23.49%	44.64%
CIFAR-10	2/255	58.75%	62.94%
	8/255	73.34%	91.44%

Table 1: We train robust neural networks on 3 datasets using Pure-IBP. These models have low IBP verified errors but when they are evaluated with a typically much tighter bound (convex adversarial polytope by Wong et al. (2018)) the verified errors increase significantly, and sometimes the model becomes unverifiable. IBP is not always looser than linear relaxation based methods; it can sometimes be powerful. We use a small model in this experiment (see Appendix B). See Table 2 for full results on verified errors of our proposed method.

Given a fixed NN, IBP gives a very loose estimation of the output range of $f(\boldsymbol{x})$. However, during training, since the weights of NN can be updated, we can equivalently view IBP as an augmented neural network, referred to as IBP-NN (Figure 1). Unlike an usual network which takes an input $\boldsymbol{x}_k$ with label y_k, IBP-NN takes two points $\boldsymbol{x}_L = \boldsymbol{x}_k - \epsilon$ and $\boldsymbol{x}_U = \boldsymbol{x}_k + \epsilon$ as input (where $\boldsymbol{x}_L \leq \boldsymbol{x} \leq \boldsymbol{x}_U$, element-wise). The bound propagation process can be equivalently seen as forward propagation in a specially structured neural network, as shown in Figure 1. After the last specification layer C (typically merged into $\mathbf{W}^{(L)}$), we can obtain $\underline{\boldsymbol{m}}(\boldsymbol{x}_k, \epsilon)$. Then, $-\underline{\boldsymbol{m}}(\boldsymbol{x}_k, \epsilon)$ is sent to softmax layer for prediction. Importantly, since $[\underline{\boldsymbol{m}}(\boldsymbol{x}_k, \epsilon)]_{y_k} = 0$ (due to the y_k-th row in C is always 0), the top-1 prediction of the augmented IBP network is y_k if and only if all other elements of $\underline{\boldsymbol{m}}(\boldsymbol{x}_k, \epsilon)$ are positive, i.e., the original network will predict correctly for all $\boldsymbol{x}_L \leq \boldsymbol{x} \leq \boldsymbol{x}_U$.

When we train the augmented IBP network with ordinary cross-entropy loss and desire it to predict correctly on an input $\boldsymbol{x}_k$, we are implicitly doing robust optimization (Eq. (2)). We attribute the success of IBP in (Gowal et al., 2018) to the power of non-linearity – instead of using linear relaxations of neural networks (Wong & Kolter, 2018) to obtain $\underline{\boldsymbol{m}}$, we train a *non-linear* network that learns to gives us a good quality $\underline{\boldsymbol{m}}$. Additionally, we also found that a network trained using IBP is not verifiable using linear relaxation based verification methods, including CROWN (Zhang et al., 2018), convex adversarial polytope (Wong & Kolter, 2018) and Fast-Lin (Weng et al., 2018). A purely IBP trained network has low IBP verified error but its verified error using convex adversarial polytope (Wong & Kolter, 2018) or Fast-Lin (Weng et al., 2018) can be much higher; sometimes the network becomes unverifiable using these typically tighter bounds (see Table 1). This also indicates that IBP is a *non-linear mechanism* different from linear relaxation based methods.

However, IBP is a very loose bound during the initial phase of training, which makes training unstable; purely using IBP frequently leads to divergence. Gowal et al. (2018) proposed to use a ϵ *schedule* where ϵ is gradually increased during training, and a mixture of robust cross-entropy loss with regular cross-entropy loss as the objective to stablize training:

$$\min_{\theta} \underset{(\boldsymbol{x},y)\in\mathcal{X}}{E} \left[\kappa L(\boldsymbol{x}; y; \theta) + (1-\kappa) L(-\underline{\boldsymbol{m}}_{\text{IBP}}(\boldsymbol{x}, \epsilon); y; \theta)\right], \tag{8}$$

where κ starts with 1 and slowly decreases to 0.5.

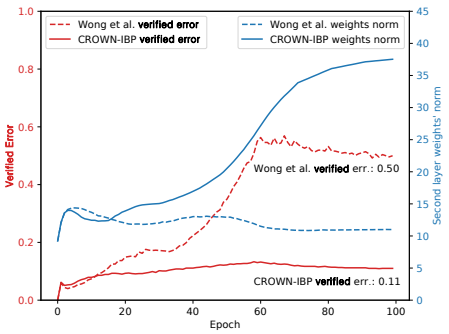


Figure 2: Verified error and 2nd CNN layer's ℓ_∞ induced norm for a model trained using (Wong et al., 2018) and CROWN-IBP. ϵ is increased from 0 to 0.3 in 60 epochs.

Issues with linear relaxation based training. Since IBP hugely outperforms linear relaxation based methods in the recent work (Gowal et al., 2018) on some datasets, we want to understand what is going wrong with linear relaxation based methods. We found that the models produced by linear relaxation methods such as (Wong & Kolter, 2018) and (Wong et al., 2018) are over-regularized especially at a larger ϵ. In Figure 2 we train a small 4-layer MNIST model and we increase ϵ from 0 to 0.3 in 60 epochs. We plot the ℓ_∞ induced norm of the 2nd CNN layer during the training process on models trained using our method and (Wong et al., 2018). We find that when ϵ becomes larger (roughly at $\epsilon = 0.15$, epoch 30), the norm of weight matrix using (Wong et al., 2018) starts to decrease, indicating that the model is forced to learn small norm weights and thus its representation power is severally reduced. Meanwhile,

the verified error also starts to ramp up. Our proposed IBP based training method, CROWN-IBP, does not have this issue; the norm of weight matrices keep increasing during the training process, and verifiable error does not significantly increase when ϵ reaches 0.3.

Another issue with current linear relaxation based training or verification methods, including convex adversarial polytope and CROWN (Zhang et al., 2018), is their high computational and memory cost, and poor scalability. For the small network in Figure 2, convex adversarial polytope (with 50 random Cauchy projections) is 8 times slower and takes 4 times more memory than CROWN-IBP (without using random projections). Convex adversarial polytope scales even worse for larger networks; see Appendix E for a comparison.

3.3 The proposed algorithm: CROWN-IBP

We have reviewed IBP and linear relaxation based methods above. IBP has great representation power due to its non-linearity, but can be tricky to tune due to its very imprecise bound at the beginning; on the other hand, linear relaxation based methods give tighter lower bounds which stabilize training, but it over-regularizes the network and forbids us to achieve good accuracy.

We propose CROWN-IBP, a new training method for certified defense where we optimize the following problem (θ are the network parameters):

$$\min_{\theta} \underset{(\boldsymbol{x},y)\in\mathcal{X}}{E} \left[L(-(\beta \underline{\boldsymbol{m}}_{\text{IBP}}(\boldsymbol{x},\epsilon) + (1-\beta)\underline{\boldsymbol{m}}_{\text{CROWN-IBP}}(\boldsymbol{x},\epsilon)); y; \theta)\right], \tag{9}$$

where our lower bound of margin $\underline{\boldsymbol{m}}(\boldsymbol{x},\epsilon)$ is a combination of two bounds with different natures: IBP, and a CROWN-style bound; L is the cross-entropy loss. CROWN is a tight linear relaxation based lower bound which is more general and often tighter than convex adversarial polytope. Importantly, CROWN-IBP *avoids the high computational cost* of ordinary CROWN (or other linear relaxation based methods, like convex adversarial polytope), by applying CROWN-style bound propagation on the final specifications only; intermediate layer bounds $\underline{z}^l$ and $\overline{z}^l$ are obtained by IBP. We start with $\beta = 0$ and use the tight bounds to stabilize initial training. Then we ramp up β from 0 to 1 while we increase ϵ from 0 to ϵ_{max}, until we reach the desired ϵ_{max} and $\beta = 1$. The network is trained using pure IBP at that point.

Benefits of CROWN-IBP. First, we compute tight linear relaxation based bounds during the early phase of training, thus greatly improve the stability and reproducibility of IBP. Second, we do not have the over-regularization problem as the CROWN-style bound is gradually phased out during training. Third, unlike the approach used in (Gowal et al., 2018) that mixes regular cross-entropy loss with robust cross-entropy loss (Eq. (8)) to stabilize IBP, we use the mixture of two lower bounds, which is still a valid lower bound of $\boldsymbol{m}$; thus, we are strictly within the robust optimization framework (Eq. (2)) and also obtain better empirical results. Forth, because we apply the CROWN-style bound propagation only to the last layer, the computational cost is greatly reduced comparing to other methods that purely relies on linear relaxation based bounds.

CROWN-IBP consists of IBP bound propagation in a forward pass and CROWN-style bound propagation in a backward pass. We discuss the details of CROWN-IBP below.

Forward Bound Propagation in CROWN-IBP. In CROWN-IBP, we first obtain $\overline{z}^{(l)}$ and $\underline{z}^{(l)}$ for all layers by applying (5), (6) and (7). Then we will obtain $\underline{\boldsymbol{m}}_{\text{IBP}}(\boldsymbol{x},\epsilon) = \underline{z}^{(L)}$ (assuming C is merged into $\mathbf{W}^{(L)}$). Obtaining these bounds is similar to forward propagation in IBP-NN (Figure 1). The time complexity of IBP is comparable to two forward propagation passes of the original network.

Linear Relaxation of ReLU neurons Given $\underline{z}^{(l)}$ and $\overline{z}^{(l)}$ computed in previous step, we first check if some neurons are always active ($\underline{z}_k^{(l)} > 0$) or always inactive ($\overline{z}_k^{(l)} < 0$), since these neurons are effectively linear and no relaxations are needed. For the remaining unstable neurons, Zhang et al. (2018); Wong et al. (2018) give a linear relaxation for the special case of element-wise ReLU activation function:

$$\alpha_k z_k^{(l)} \le \sigma(z_k^{(l)}) \le \frac{\overline{z}_k^{(l)}}{\overline{z}_k^{(l)} - \underline{z}_k^{(l)}} z_k^{(l)} - \frac{\overline{z}_k^{(l)}\underline{z}_k^{(l)}}{\overline{z}_k^{(l)} - \underline{z}_k^{(l)}}, \quad \text{for all } k \in [n_l] \text{ and } \underline{z}_k^{(l)} < 0 < \overline{z}_k^{(l)}, \tag{10}$$

where $0 \le \alpha_k \le 1$; Zhang et al. (2018) proposes to adaptively select $\alpha_k = 1$ when $\overline{z}_k^{(l)} > |\underline{z}_k^{(l)}|$ and 0 otherwise, which minimizes the relaxation error. In other words, for an input vector $z^{(l)}$, we effectively replace the ReLU layer with a linear layer, giving upper or lower bounds of the output:

$$\underline{\mathbf{D}}^{(l)} z^{(l)} \le \sigma(z^{(l)}) \le \overline{\mathbf{D}}^{(l)} z^{(l)} + \overline{c}_d^{(l)} \tag{11}$$

where $\underline{\mathbf{D}}^{(l)}$ and $\overline{\mathbf{D}}^{(l)}$ are two diagonal matrices representing the "weights" of the relaxed ReLU layer. In the following we focus on conceptually presenting the algorithm, while more details of each term can be found in the Appendix.

Backward Bound Propagation in CROWN-IBP. Unlike IBP, CROWN-style bounds start computation from the last layer. Suppose we want to obtain the lower bound $[\underline{\boldsymbol{m}}_{\text{CROWN-IBP}}(\boldsymbol{x}, \epsilon)]_i = \underline{z}_i^{(L)}$ (we assume the specification matrix C has been merged into $\mathbf{W}^{(L)}$). The input to layer $\mathbf{W}^{(L)}$ is $\sigma(z^{(L-1)})$, which has already been replaced by Eq. (11). CROWN-style bounds choose the lower bound of $\sigma(z_k^{(L-1)})$ (LHS of (11)) when $\mathbf{W}_{i,k}^{(L)}$ is positive, and choose the upper bound when $\mathbf{W}_{i,k}^{(L)}$ is negative. We then merge $\mathbf{W}^{(L)}$ and the linearized ReLU layer together and define:

$$\mathbf{A}_{i,:}^{(L-1)} = \mathbf{W}_{i,:}^{(L)} \mathbf{D}^{i,(L-1)}, \quad \mathbf{D}_{k,k}^{i,(L-1)} = \begin{cases} \underline{\mathbf{D}}_{k,k}^{(L-1)}, & \text{if } \mathbf{W}_{i,k}^{(L)} > 0 \\ \overline{\mathbf{D}}_{k,k}^{(L-1)}, & \text{if } \mathbf{W}_{i,k}^{(L)} \leq 0 \end{cases} \tag{12}$$

Now we have a lower bound $\underline{z}_i^{(L)} = \mathbf{A}_{i,:}^{(L-1)} z^{(L-1)} + \underline{b}_i^{(L-1)} \leq z_i^{(L)}$ where $\underline{b}_i^{(L-1)} = \sum_{k, \mathbf{W}_{i,k}^{(L)} < 0} \mathbf{W}_{i,k}^{(L)} \overline{c}_k^{(l)} + \boldsymbol{b}^{(L)}$ collects all terms not related to $z^{(L-1)}$. Note that the diagonal matrix $\mathbf{D}^{i,(L-1)}$ implicitly depends on i. Then, we merge $\mathbf{A}_{i,:}^{(L-1)}$ with the next linear layer, which is straight forward by plugging in $z^{(L-1)} = \mathbf{W}^{(L-1)} \sigma(z^{(L-2)}) + \boldsymbol{b}^{(L-1)}$:

$$z_i^{(L)} \geq \mathbf{A}_{i,:}^{(L-1)} \mathbf{W}^{(L-1)} \sigma(z^{(L-2)}) + \mathbf{A}_{i,:}^{(L-1)} \boldsymbol{b}^{(L-1)} + \underline{b}_i^{(L-1)}.$$

Then we continue to unfold the next ReLU layer $\sigma(z^{(L-2)})$ using its linear relaxations, and compute a new $\mathbf{A}^{(L-2)} \in \mathbb{R}^{n_L \times n_{L-2}}$ matrix, with $\mathbf{A}_{i,:}^{(L-2)} = \mathbf{A}_{i,:}^{(L-1)} \mathbf{W}^{(L-1)} \mathbf{D}^{i,(L-2)}$ in a similar manner as in (12). Along with the bound propagation process, we need to compute a series of matrices, $\mathbf{A}^{(L-1)}, \cdots, \mathbf{A}^{(0)}$, where $\mathbf{A}_{i,:}^{(l)} = \mathbf{A}_{i,:}^{(l+1)} \mathbf{W}^{(l+1)} \mathbf{D}^{i,(l)} \in \mathbb{R}^{n_L \times n_{(l)}}$, and $\mathbf{A}_{i,:}^{(0)} = \mathbf{A}_{i,:}^{(1)} \mathbf{W}^{(1)} = \mathbf{W}_{i,:}^{(L)} \mathbf{D}^{i,(L-1)} \mathbf{W}^{(L-2)} \mathbf{D}^{i,(L-2)} \mathbf{A}^{(L-2)} \cdots \mathbf{D}^{i,(1)} \mathbf{W}^{(1)}$. At this point, we merged all layers of the network into a linear layer: $z_i^{(L)} \geq \mathbf{A}_{i,:}^{(0)} \boldsymbol{x} + \underline{b}$, where $\underline{b}$ collects all terms not related to $\boldsymbol{x}$. A lower bound for $z_i^{(L)}$ with $\boldsymbol{x}_L \leq \boldsymbol{x} \leq \boldsymbol{x}_U$ can then be easily given as

$$[\underline{\boldsymbol{m}}_{\text{CROWN-IBP}}]_i \equiv \underline{z}_i^{(L)} = \mathbf{A}_{i,:}^{(0)} \boldsymbol{x} + \underline{b} \geq \sum\nolimits_{k, \mathbf{A}_{i,k}^{(0)} < 0} \mathbf{A}_{i,k}^{(0)} \boldsymbol{x}_{U,k} + \sum\nolimits_{k, \mathbf{A}_{i,k}^{(0)} > 0} \mathbf{A}_{i,k}^{(0)} \boldsymbol{x}_{L,k} + \underline{b} \tag{13}$$

For ReLU networks, convex adversarial polytope (Wong & Kolter, 2018) uses a very similar bound propagation procedure. CROWN-style bounds allow an adaptive selection of α_i in (10), thus offen gives slightly better bounds. We give details on each term in Appendix F.

Computational Cost of CROWN-IBP. In ordinary CROWN (Zhang et al., 2018) and convex adversarial polytope (Wong & Kolter, 2018), the bound propagation process is much more expensive than CROWN-IBP, since it needs to use (13) to compute *all intermediate layer's* $\underline{z}_i^{(m)}$ and $\overline{z}_i^{(m)}$ ($m \in [L]$), by considering $\mathbf{W}^{(m)}$ as the final layer of the network. In this case, for each layer m we need a different set of $\mathbf{A}$ matrices, defined as $\mathbf{A}^{m,(l)}, l \in \{m-1, \cdots, 0\}$. This causes three computational issues:

1. Unlike the last layer $\mathbf{W}^{(L)}$, an intermediate layer $\mathbf{W}^{(m)}$ has a much larger output dimension $n_m \gg n_L$ typically thus all $\mathbf{A}^{m,(l)} \in \{\mathbf{A}^{m,(m-1)}, \cdots, \mathbf{A}^{m,(0)}\}$ will have large dimensions $\mathbb{R}^{n_m \times n_l}$.
2. Computation of all $\mathbf{A}^{m,(l)}$ matrices is expensive. Suppose the network has n neurons for all $L-1$ intermediate and input layers and $n_L \ll n$ neurons for the output layer (assuming $L \geq 2$), the time complexity of ordinary CROWN or convex adversarial polytope is $O(\sum_{l=1}^{L-2} l n^3 + (L-1) n_L n^2) = O((L-1)^2 n^3 + (L-1) n_L n^2) = O(L n^2 (L n + n_L))$. A ordinary forward propagation only takes $O(Ln^2)$ time per example, thus ordinary CROWN does not scale up to large networks for training, due to its quadratic dependency in L and extra Ln times overhead.
3. When both $\mathbf{W}^{(l)}$ and $\mathbf{W}^{(l-1)}$ represent convolutional layers with small kernel tensors $\mathbf{K}^{(l)}$ and $\mathbf{K}^{(l-1)}$, there are no efficient GPU operations to form the matrix $\mathbf{W}^{(l)} \mathbf{D}^{(l-1)} \mathbf{W}^{(l-1)}$ using $\mathbf{K}^{(l)}$ and $\mathbf{K}^{(l-1)}$. Existing implementations either unfold at least one of the convolutional kernels to fully connected weights, or use sparse matrices to represent $\mathbf{W}^{(l)}$ and $\mathbf{W}^{(l-1)}$. They suffer from poor hardware efficiency on GPUs.

In CROWN-IBP, we do not have the first and second issues since we use IBP to obtain intermediate layer bounds, which is only slower than forward propagation by a constant factor. The time complexity of the backward bound propagation in CROWN-IBP is $O((L-1)n_L n^2)$, only n_L times slower than forward propagation and significantly more scalable than ordinary CROWN (which is Ln times slower than forward propagation, where typically $n \gg n_L$). The third issue is also not a concern, since we start from the last specification layer $\mathbf{W}^{(L)}$ which is a small fully connected layer. Suppose we need to compute $\mathbf{W}^{(L)}\mathbf{D}^{(L-1)}\mathbf{W}^{(L-1)}$ and $\mathbf{W}^{(L-1)}$ is a convolutional layer with kernel $\mathbf{K}^{(L-1)}$, we can efficiently compute $(\mathbf{W}^{(L-1)\top}(\mathbf{D}^{(L-1)}\mathbf{W}^{(L)\top}))^\top$ on GPUs using the *transposed convolution* operator with kernel $\mathbf{K}^{(L-1)}$. Conceptually, the backward pass of CROWN-IBP propagates a small specification matrix $\mathbf{W}^{(L)}$ backwards, replacing affine layers with their transposed operators, and activation function layers with a diagonal matrix product. This allows efficient implementation and better scalability.

4 Experiments

4.1 Setup and Models

To discourage hand-tuning on a small set of models, we use 20 different model architectures for a total of **53 models** for MNIST, Fashion-MNIST and CIFAR-10 datasets, from small 2×CNN+2×FC models to wide 4×CNN+3×FC models. The models are a mixture of networks with different depths, widths and convolution kernel sizes. Details are presented in Appendix B. We consider ℓ_∞ robustness and report both the best and worst verified error achieved over all models, and the median of error. Verified errors are evaluated using IBP on the test set. The median error implies that at least half of models trained are as good as this error. We list hyperparameters in Appendix A. Since we focus on improving stability and performance of IBP models, in experiments we compare three different variants of IBP[3]:

- CROWN-IBP: our proposed method, using CROWN-style linear relaxations in a backward manner to improve IBP training stability, and a mixture of CROWN and IBP lower bounds in robust cross-entropy (CE) loss, as in Eq. (9). No regular CE loss is used.
- Pure-IBP: using the lower bounds purely from IBP in robust CE loss. No regular CE loss is used. Equivalent to Eq. (9) with β fixed to 1.
- Natural-IBP-κ: proposed by (Gowal et al., 2018), using the lower bounds provided by IBP in robust CE loss (multiplied by $1-\kappa$), plus κ times a regular CE loss term as in Eq. (8). This κ is initialized as 1 and is gradually reduced during training. In our experiments we choose two final κ values, 0.5 and 0. Note that Gowal et al. (2018) uses 0.5 as the final κ value.

4.2 Comparisons to IBP based methods

In Table 2 we show the verified errors on test sets for CROWN-IBP and the other two IBP baselines. We also include best verified errors reported in literature for comparison. Numbers reported in Gowal et al. (2018) use the same training method as Natural IBP with final $\kappa = 0.5$, albeit they use different hyperparameters and sometimes different ϵ for training and evaluation; we always use the same ϵ value for training and evaluation. CROWN-IBP's best, median, and worst test verified errors are consistently better than all other IBP-based baselines across all models and ϵ's. Especially, on MNIST with $\epsilon = 0.3$ and $\epsilon = 0.4$ we achieve 7.46% and 12.96% best verified error, respectively, outperforming all previous works and significantly better than convex relaxation based training methods (Wong et al., 2018); a similar level of advantage can also be observed on Fashion-MNIST (Wong & Kolter, 2018). For small $\epsilon = \{0.1, 0.2\}$ on MNIST, we find that IBP based methods tend to overfit. For example, adding an ℓ_1 regularization term can decreases verified error at $\epsilon = 0.1$ from 5.63% to 3.60% (see Section 4.4 for more details); this explains the performance gap in Table 2 at small ϵ between CROWN-IBP and convex adversarial polytope, since the latter method provides implicit ℓ_1 regularization. On CIFAR-10 with $\epsilon = \frac{8}{255}$, CROWN-IBP is better than all other methods except (Gowal et al., 2018); however, Gowal et al. (2018) obtained the best result by using a large network trained for 3200 epochs with a fine-tuned ϵ schedule on 32 TPUs; practically, the reproduciable verified error by Gowal et al. (2018) is around 71% - 72% (see notes under table). In contrast, our results can be obtained in reasonable time using a single RTX 2080 Ti GPU. We include training time comparisons in Appendix E.

[3]Our implementations of all IBP variants used in this paper is available at https://github.com/huanzhang12/CROWN-IBP

Table 2: The verified and standard (clean) test errors for models trained on MNIST, Fashion-MNIST and CIFAR using different methods. Here we train 53 models using CROWN-IBP, pure-IBP and natural-IBP (with final $\kappa = 0$ or 0.5). For each scenario, we pick 3 representative models among all models: the models with smallest, median, and largest verified error. We also report the standard error of *these three selected models*.

Dataset	ϵ (ℓ_∞)	Model Family	Method	Verified Test Error (%) best	median	worst	Standard Test Error(%) best	median	worst	**Best** results reported in literature (%) Source	Verified Err.	Std. Err.
MNIST	0.1[1]	10 small models	Pure-IBP	4.79	5.74	7.32	1.48	1.59	2.5	Gowal et al. (2018)	2.23[2]	1.06
			Natural-IBP, final $\kappa = 0$	4.87	5.72	7.24	1.51	1.34	2.46			
			Natural-IBP, final $\kappa = 0.5$	5.24	5.95	7.36	1.41	1.88	**1.87**	Xiao et al. (2019)	4.4	1.05
			CROWN-IBP	**4.21**[1]	**5.18**	**6.8**	**1.41**	**1.83**	2.58			
		8 large models	Pure-IBP	5.9	6.25	7.82	**1.14**	1.12	1.23	Wong et al. (2018)	3.67	1.08
			Natural-IBP, final $\kappa = 0$	5.77	6.3	7.5	1.21	1.13	1.34			
			Natural-IBP, final $\kappa = 0.5$	6.05	6.4	7.7	1.19	1.33	1.24	Dvijotham et al. (2018b)	4.44	1.2
			CROWN-IBP	**5.22**[1]	**5.63**	**6.34**	1.19	**1.05**	**1.03**	Mirman et al. (2018)	3.4	1.0
	0.2[1]	10 small models	Pure-IBP	6.9	8.24	12.67	1.93	2.76	4.14	Gowal et al. (2018)	4.48[2]	1.66
			Natural-IBP, final $\kappa = 0$	6.84	8.16	12.92	2.01	2.56	3.93			
			Natural-IBP, final $\kappa = 0.5$	7.31	8.71	13.54	**1.62**	2.36	**3.22**			
			CROWN-IBP	**6.11**[1]	**7.29**	**11.97**	1.93	**2.3**	3.86			
		8 large models	Pure-IBP	7.56	8.6	9.8	1.96	2.19	1.39	Xiao et al. (2019)	10.21	1.9
			Natural-IBP, final $\kappa = 0$	8.26	8.72	9.84	1.45	1.73	**1.31**			
			Natural-IBP, final $\kappa = 0.5$	8.42	8.9	10.09	1.76	1.42	1.53			
			CROWN-IBP	**6.06**[1]	**6.42**	**7.64**	**1.09**	**1.33**	1.36			
	0.3	10 small models	Pure-IBP	10.54	12.02	20.47	2.78	**3.31**	6.07	Gowal et al. (2018)	8.05[2]	1.66
			Natural-IBP, final $\kappa = 0$	9.96	12.09	21.0	2.7	3.48	6.68			
			Natural-IBP, final $\kappa = 0.5$	10.37	12.78	21.99	**2.11**	3.44	**5.19**			
			CROWN-IBP	**8.87**	**11.29**	**16.83**	2.43	3.62	7.26	Wong et al. (2018)	43.1	14.87
		8 large models	Pure-IBP	10.43	10.83	11.99	2.01	2.38	3.29			
			Natural-IBP, final $\kappa = 0$	10.74	11.73	12.16	2.17	2.46	**1.6**			
			Natural-IBP, final $\kappa = 0.5$	11.23	11.71	12.4	1.72	2.09	1.63	Xiao et al. (2019)	19.32	2.67
			CROWN-IBP	**7.46**	**8.47**	**8.57**	**1.48**	**1.52**	1.99			
	0.4	10 small models	Pure-IBP	16.72	18.89	37.42	4.2	5.4	9.63	Gowal et al. (2018)	14.88	1.66
			Natural-IBP, final $\kappa = 0$	16.1	18.75	35.3	3.8	4.93	11.32			
			Natural-IBP, final $\kappa = 0.5$	16.54	19.14	35.42	**3.4**	**3.65**	**7.54**			
			CROWN-IBP	**15.38**	**18.57**	**24.56**	3.61	4.83	8.46			
		8 large models	Pure-IBP	15.17	16.54	18.98	2.83	3.79	4.91			
			Natural-IBP, final $\kappa = 0$	15.63	16.06	17.11	2.93	3.4	3.75			
			Natural-IBP, final $\kappa = 0.5$	15.74	16.42	17.98	**2.35**	**2.31**	**3.15**			
			CROWN-IBP	**12.96**	**13.43**	**14.25**	2.76	2.85	3.36			
Fasthion-MNIST	0.1	10 small models	Pure-IBP	22.91	23.92	**25.70**	16.14	16.23	18.66	Wong & Kolter (2018)	34.53	21.73
			Natural-IBP, final $\kappa = 0$	22.34	23.72	25.43	15.10	16.22	17.65			
			Natural-IBP, final $\kappa = 0.5$	23.44	24.62	26.21	**13.20**	**14.58**	**15.02**			
			CROWN-IBP	**22.00**	**23.43**	25.87	15.11	16.44	18.08			
		8 large models	Pure-IBP	22.37	23.01	23.73	15.57	15.43	15.25			
			Natural-IBP, final $\kappa = 0$	22.07	23.74	25.56	14.82	13.95	15.31			
			Natural-IBP, final $\kappa = 0.5$	23.87	25.47	27.19	12.72	**13.62**	**13.07**			
			CROWN-IBP	**21.27**	**21.71**	**22.43**	**14.19**	14.70	15.41			
CIFAR-10	2/255	9 small models	Pure-IBP	54.69	57.84	60.58	40.59	45.51	51.38	Gowal et al. (2018)	49.98	29.84
			Natural-IBP, final $\kappa = 0$	54.56	58.42	60.69	40.32	47.42	50.73			
			Natural-IBP, final $\kappa = 0.5$	56.89	60.66	63.58	**34.28**	**39.28**	**48.03**	Wong et al. (2018)	46.11	31.72
			CROWN-IBP	**52.46**	**57.55**	**60.67**	39.11	46.76	50.86			
		8 large models	Pure-IBP	55.47	56.41	58.54	41.59	44.33	46.54	Xiao et al. (2019)	54.07	38.88
			Natural-IBP, final $\kappa = 0$	55.51	56.74	57.85	42.41	43.71	44.74			
			Natural-IBP, final $\kappa = 0.5$	57.05	59.7	60.25	**34.77**	**35.8**	**38.95**	Mirman et al. (2018)	47.8	38.0
			CROWN-IBP	**52.52**	**53.9**	**56.05**	39.34	40.07	43.57			
	8/255	9 small models	Pure-IBP	72.07	73.34	73.88	61.11	61.01	64.0	Gowal et al. (2018)	67.96[3]	50.51
			Natural-IBP, final $\kappa = 0$	72.42	72.57	73.49	62.26	60.98	63.5			
			Natural-IBP, final $\kappa = 0.5$	73.88	75.16	76.93	**55.66**	**52.53**	**53.79**	Xiao et al. (2019)	79.73	59.55
			CROWN-IBP	**71.28**	**72.15**	**73.66**	59.07	59.18	63.17			
		8 large models	Pure-IBP	72.75	73.23	73.82	59.23	65.96	66.35	Wong et al. (2018)	78.22	71.33
			Natural-IBP, final $\kappa = 0$	72.18	72.83	74.38	62.54	59.6	61.99			
			Natural-IBP, final $\kappa = 0.5$	74.84	75.59	97.93	**51.71**	**54.41**	**54.12**			
			CROWN-IBP	**70.79**	**71.17**	**72.29**	57.82	58.68	59.73	Dvijotham et al. (2018b)	73.33	51.36

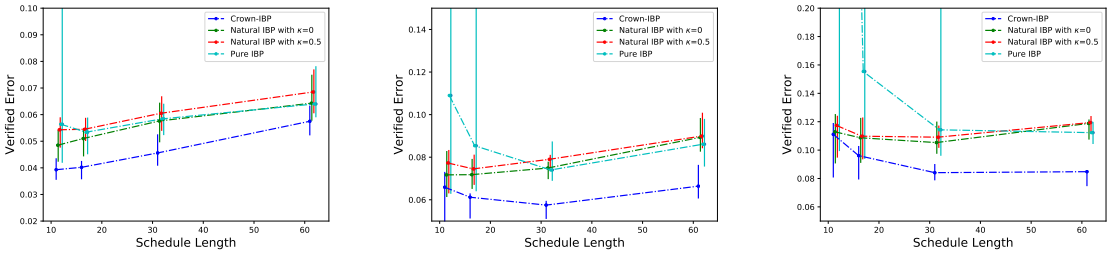
[1] For small ϵ IBP based methods tend to overfit on training set, thus we observe suboptimal verified errors on this table as we do not fine-tune hyperparameter for each individual setting. By adding explicit ℓ_1 regularization, CROWN IBP can achieve **3.60%** verified error at $\epsilon = 0.1$ and **5.48%** at $\epsilon = 0.2$ (see Section 4.4). Additionally, early stopping can also help achieve a better verified error; with early stopping the best verified errors achieved by CROWN-IBP are **3.55%** at $\epsilon = 0.1$ and **4.98%** for $\epsilon = 0.2$. See Figure 3.

[2] These models reported in Gowal et al. (2018) are trained at a larger ϵ_{train} than the ϵ used for evaluation. We always use the same ϵ for training and evaluation. Gowal et al. (2018) trained models with $\epsilon_{\text{train}} = 0.2$ and evaluate them with $\epsilon = 0.1$; they also trained models with $\epsilon_{\text{train}} = 0.4$ and evaluate them with $\epsilon = 0.3$ and $\epsilon = 0.2$.

[3] Practically reproducible verified error is about 71% - 72%, matching our reported numbers for Natural-IBP. See https://github.com/deepmind/interval-bound-propagation/issues/1#issuecomment-492552237

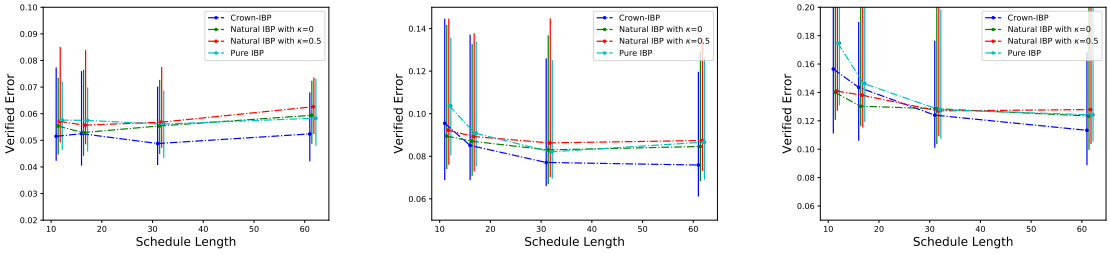
4.3 Training stability

To evaluate training stability, we compare the verified errors obtained by training processes under different ϵ schedule length (10, 15, 30, 60). We compare the best, worst and median verified errors over all 18 models for MNIST. Our results are presented in Figure 3 (for 8 large models) and Figure 4 (for 10 small models) at $\epsilon = 0.1, 0.2, 0.3$. The upper and lower bound of an error bar are the worst and best verified error, respectively, and the lines go through median values. We can see that both Natural-IBP and CROWN-IBP can improve training stability when the schedule length is not sufficient (10, 20 epochs). When schedule length is above 30, CROWN-IBP's verified errors are consistently better than any other method. Pure-IBP cannot stably converge



(a) $\epsilon = 0.1$, best $\mathbf{3.55}\%$ (b) $\epsilon = 0.2$, best $\mathbf{4.98}\%$ (c) $\epsilon = 0.3$, best $\mathbf{7.46}\%$

Figure 3: Verified error vs. ϵ schedule length (10, 20, 30, 60) on 8 large MNIST models. The upper and lower bound of an error bar are the worst and best verified error, respectively. The dotted lines go through median values. For a small ϵ, using a shorter schedule length improves verified error due to early stopping, which prevents overfitting.



(a) $\epsilon = 0.1$, best $\mathbf{4.05}\%$ (b) $\epsilon = 0.2$, best $\mathbf{6.11}\%$ (c) $\epsilon = 0.3$, best $\mathbf{8.87}\%$

Figure 4: Verified error vs. ϵ schedule length (10, 20, 30, 60) on 10 small MNIST models. The upper and lower bound of an error bar are worst and best verified error, respectively. The dotted lines go through median values.

on all models when ϵ schedule is short, espeically for a larger ϵ. We conduct additional training stability experiments on CIFAR-10 dataset and the observations are similar (see Appendix D).

Another interesting observation is that at a small $\epsilon = \{0.1, 0.2\}$, a shorter ϵ schedule improves results for large models (Figure 3). This is due to early stopping which controls overfitting (see Section 4.4). Since we decrease the learning rate by half every 10 epochs after the schedule ends, a shorter ϵ schedule implies that the learning process stops earlier.

To further test the training stability of CROWN-IBP, we run each MNIST experiment (in Table 2) 5 times on 10 small models. The mean and standard deviation of the verified and standard errors on test set are presented in Appendix C. Standard deviations of verified errors are very small, giving us further evidence of good stability.

4.4 Overfitting issue with small ϵ

We found that on MNIST for a small ϵ, the verified error obtained by IBP based methods are not as good as linear relaxation based methods (Wong et al., 2018; Mirman et al., 2018). Gowal et al. (2018) thus propose to train models using a larger ϵ and evaluate them under a smaller ϵ, for example $\epsilon_{\text{train}} = 0.4$ and $\epsilon_{\text{eval}} = 0.3$. Instead, we investigated this issue further and found that many CROWN-IBP trained models achieve very small verified errors (close to 0 and sometimes exactly 0) on training set (see Table 3). This indicates possible overfitting during training. As we discussed in Section 3.1, linear relaxation based methods implicitly regularize the weight matrices so the network does not overfit when ϵ is small. Inspired by this finding, we want to see if adding an explicit ℓ_1 regularization term in CROWN-IBP training helps when $\epsilon = 0.1$ or 0.2. The verified and standard errors on the training and test sets with and without regularization can be found in Table 3. We can see that with a small ℓ_1 regularization added ($\lambda = 5 \times 10^{-5}$) we can reduce verified error on test set significantly. This makes CROWN-IBP results comparable to the numbers reported in convex adversarial polytope (Wong et al., 2018); at $\epsilon = 0.1$, the best model using convex adversarial polytope training can achieve 3.67% certified error, while CROWN-IBP achieves 3.60% best certified error on the models

presented in Table 3. The overfitting is likely caused by IBP's strong representation power, which also explains why IBP based methods significantly outperform linear relaxation based methods at larger ϵ values. Using early stopping can also improve verified error on test set; see Section 4.3.

ϵ	Model Name (see Appendix B)	λ: ℓ_1 regularization	Training		Test	
			standard error	verified error	standard error	verified error
0.1	P	0	0.01%	0.01%	1.05%	5.63%
	P	5×10^{-5}	0.32%	0.98%	1.30%	**3.60%**
	O	0	0.02%	0.05%	0.82%	6.02%
	O	5×10^{-5}	0.38%	1.34%	1.43%	4.02%
0.2	P	0	0.35%	1.40%	1.09%	6.06%
	P	5×10^{-5}	1.02%	3.73%	1.48%	**5.48%**
	O	0	0.31%	1.54%	1.22%	6.64%
	O	5×10^{-5}	1.09%	4.08%	1.69%	5.72%

Table 3: ℓ_1 regularized and unregularized models' standard and verified errors on training and test set. At a small ϵ, CROWN-IBP may overfit and adding regularization helps robust generalization; on the other hand, convex relaxation based methods (Wong et al., 2018) provides implicitly regularization which helps generalization under small ϵ but deteriorate model performance at larger ϵ.

5 Conclusions

In this paper, we propose a new certified defense method, CROWN-IBP, by combining the fast interval bound propagation (IBP) in the forward pass and a tight linear relaxation based bound, CROWN, in the backward pass. Our method enjoys the non-linear representation power and high computational efficiency provided by IBP while facilitating the tight CROWN bound to stabilize training strictly under the robust optimization framework. Our experiments on a variety of model structures and three datasets show that CROWN-IBP consistently outperforms other IBP baselines and achieves state-of-the-art verified errors.

References

Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *International Conference on Machine Learning (ICML)*, 2018.

Buckman, J., Roy, A., Raffel, C., and Goodfellow, I. Thermometer encoding: One hot way to resist adversarial examples. *International Conference on Learning Representations*, 2018.

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, 2017a.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *2017 38th IEEE Symposium on Security and Privacy (SP)*, pp. 39–57. IEEE, 2017b.

Cohen, J. M., Rosenfeld, E., and Kolter, J. Z. Certified adversarial robustness via randomized smoothing. *arXiv preprint arXiv:1902.02918*, 2019.

Dvijotham, K., Garnelo, M., Fawzi, A., and Kohli, P. Verification of deep probabilistic models. *CoRR*, abs/1812.02795, 2018a. URL http://arxiv.org/abs/1812.02795.

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*, 2018b.

Dvijotham, K., Stanforth, R., Gowal, S., Mann, T., and Kohli, P. A dual approach to scalable verification of deep networks. *UAI*, 2018c.

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

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.

Guo, C., Rana, M., Cisse, M., and van der Maaten, L. Countering adversarial images using input transformations. In *ICLR*, 2018.

He, W., Wei, J., Chen, X., Carlini, N., and Song, D. Adversarial example defenses: ensembles of weak defenses are not strong. In *Proceedings of the 11th USENIX Conference on Offensive Technologies*, pp. 15–15. USENIX Association, 2017.

Hein, M. and Andriushchenko, M. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2266–2276, 2017.

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.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial machine learning at scale. In *International Conference on Learning Representations*, 2017.

Lecuyer, M., Atlidakis, V., Geambasu, R., Hsu, D., and Jana, S. Certified robustness to adversarial examples with differential privacy. *arXiv preprint arXiv:1802.03471*, 2018.

Li, B., Chen, C., Wang, W., and Carin, L. Second-order adversarial attack and certifiable robustness. *arXiv preprint arXiv:1809.03113*, 2018.

Liu, X., Cheng, M., Zhang, H., and Hsieh, C.-J. Towards robust neural networks via random self-ensemble. In *European Conference on Computer Vision*, pp. 381–397. Springer, 2018.

Ma, X., Li, B., Wang, Y., Erfani, S. M., Wijewickrema, S., Houle, M. E., Schoenebeck, G., Song, D., and Bailey, J. Characterizing adversarial subspaces using local intrinsic dimensionality. In *International Conference on Learning Representations (ICLR)*, 2018.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

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.

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 582–597. IEEE, 2016.

Qin, C., Dvijotham, K. D., O'Donoghue, B., Bunel, R., Stanforth, R., Gowal, S., Uesato, J., Swirszcz, G., and Kohli, P. Verification of non-linear specifications for neural networks. *ICLR*, 2019.

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

Raghunathan, A., Steinhardt, J., and Liang, P. S. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems*, pp. 10900–10910, 2018b.

Salman, H., Yang, G., Zhang, H., Hsieh, C.-J., and Zhang, P. A convex relaxation barrier to tight robust verification of neural networks. *arXiv preprint arXiv:1902.08722*, 2019.

Samangouei, P., Kabkab, M., and Chellappa, R. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. *arXiv preprint arXiv:1805.06605*, 2018.

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.

Singh, G., Gehr, T., Püschel, M., and Vechev, M. Robustness certification with refinement. *ICLR*, 2019.

Sinha, A., Namkoong, H., and Duchi, J. Certifying some distributional robustness with principled adversarial training. In *ICLR*, 2018.

Song, Y., Kim, T., Nowozin, S., Ermon, S., and Kushman, N. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*, 2017.

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

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

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. 6369–6379, 2018b.

Weng, T.-W., Zhang, H., Chen, H., Song, Z., Hsieh, C.-J., Boning, D., Dhillon, I. S., and Daniel, L. Towards fast computation of certified robustness for ReLU networks. In *International Conference on Machine Learning*, 2018.

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.

Wong, E., Schmidt, F., Metzen, J. H., and Kolter, J. Z. Scaling provable adversarial defenses. *Advances in Neural Information Processing Systems (NIPS)*, 2018.

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

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 (NIPS)*, dec 2018.

Zhang, H., Zhang, P., and Hsieh, C.-J. Recurjac: An efficient recursive algorithm for bounding jacobian matrix of neural networks and its applications. *AAAI Conference on Artificial Intelligence*, 2019.

Zheng, T., Chen, C., and Ren, K. Distributionally adversarial attack. *arXiv preprint arXiv:1808.05537*, 2018.

A Hyperparameters in Experiments

All our MNIST, CIFAR-10 models are trained on a single NVIDIA 2080 Ti GPU. In all our experiments, if not mentioned otherwise, we use the following hyperparameters:

- For MNIST, we train 100 epochs with batch size 256. We use Adam optimizer and the learning rate is 5×10^{-4}. The first epoch is standard training for warming up. We gradually increase ϵ linearly per batch in our training process with a ϵ schedule length of 60. We reduce the learning rate by 50% every 10 epochs after ϵ schedule ends. No data augmentation technique is used and the whole 28×28 images are used (normalized to 0 - 1 range).
- For CIFAR, we train 200 epoch with batch size 128. We use Adam optimizer and the learning rate is 0.1%. The first 10 epochs are standard training for warming up. We gradually increase ϵ linearly per batch in our training process with a ϵ schedule length of 120. We reduce the learning rate by 50% every 10 epochs after ϵ schedule ends. We use random horizontal flips and random crops as data augmentation. The three channels are normalized with mean (0.4914, 0.4822, 0.4465) and standard deviation (0.2023, 0.1914, 0.2010). These numbers are per-channel statistics from the training set used in (Gowal et al., 2018).

For all experiments, we set $\beta = 1$ when the ϵ schedule starts. We decrease β linearly to 0 when ϵ finishes its increasing schedule and reaches ϵ_{max}. We did not tune the schedule for parameter β, and it always has the same schedule length as the ϵ schedule. All verified error numbers are evaluated on the test set, using IBP, since the networks are trained using pure IBP after ϵ reaches the target. We found that CROWN (Zhang et al., 2018) or Fast-Lin (Weng et al., 2018) cannot give tight verification bounds on IBP trained models (some comparison results are given in Table 1).

B Model Structure

Name	Model Structure (all models have a last FC 10 layer, which are omitted)
A (MNIST Only)	Conv 4 4×4+2, Conv 8 4×4+2, FC 128
B	Conv 8 4×4+2, Conv 16 4×4+2, FC 256
C	Conv 4 3×3+1, Conv 8 3×3+1, Conv 8 4×4+4, FC 64
D	Conv 8 3×3+1, Conv 16 3×3+1, Conv 16 4×4+4, FC 128
E	Conv 4 5×5+1, Conv 8 5×5+1, Conv 8 5×5+4, FC 64
F	Conv 8 5×5+1, Conv 16 5×5+1, Conv 16 5×5+4, FC 128
G	Conv 4 3×3+1, Conv 4 4×4+2, Conv 8 3×3+1, Conv 8 4×4+2, FC 256, FC 256
H	Conv 8 3×3+1, Conv 8 4×4+2, Conv 16 3×3+1, Conv 16 4×4+2, FC 256, FC 256
I	Conv 4 3×3+1, Conv 4 4×4+2, Conv 8 3×3+1, Conv 8 4×4+2, FC 512, FC 512
J	Conv 8 3×3+1, Conv 8 4×4+2, Conv 16 3×3+1, Conv 16 4×4+2, FC 512, FC 512
K	Conv 16 3×3+1, Conv 16 4×4+2, Conv 32 3×3+1, Conv 32 4×4+2, FC 256, FC 256
L	Conv 16 3×3+1, Conv 16 4×4+2, Conv 32 3×3+1, Conv 32 4×4+2, FC 512, FC 512
M	Conv 32 3×3+1, Conv 32 4×4+2, Conv 64 3×3+1, Conv 64 4×4+2, FC 512, FC 512
N	Conv 64 3×3+1, Conv 64 4×4+2, Conv 128 3×3+1, Conv 128 4×4+2, FC 512, FC 512
O(MNIST Only)	Conv 64 5×5+1, Conv 128 5×5+1, Conv 128 4×4+4, FC 512
P(MNIST Only)	Conv 32 5×5+1, Conv 64 5×5+1, Conv 64 4×4+4, FC 512
Q	Conv 16 5×5+1, Conv 32 5×5+1, Conv 32 5×5+4, FC 512
R	Conv 32 3×3+1, Conv 64 3×3+1, Conv 64 3×3+4, FC 512
S(CIFAR Only)	Conv 32 4×4+2, Conv 64 4×4+2, FC 128
T(CIFAR Only)	Conv 64 4×4+2, Conv 128 4×4+2, FC 256

Table 4: Model structures used in all of our experiments. We use ReLU activations for all models. To save space, we omit the last fully connected layer as its output dimension is always 10. In the table, "Conv k $w \times w + s$" represents to a 2D convolutional layer with k filters of size $w \times w$ and a stride of s.

Table 4 gives the 18 model structures used in our paper. MNIST and Fashion-MNIST use exactly the same model structures. Most CIFAR models share the same structures as MNIST models (unless noted on the table) except that their input dimensions are different. Model A is too small for CIFAR thus we remove it for CIFAR experiments. Models A - J are the "small models" reported in Table 2. Models K - T are the "large models" reported in Table 2. For results in Table 1, we use a small model (model structure B) for all three datasets.

C Reproducibility

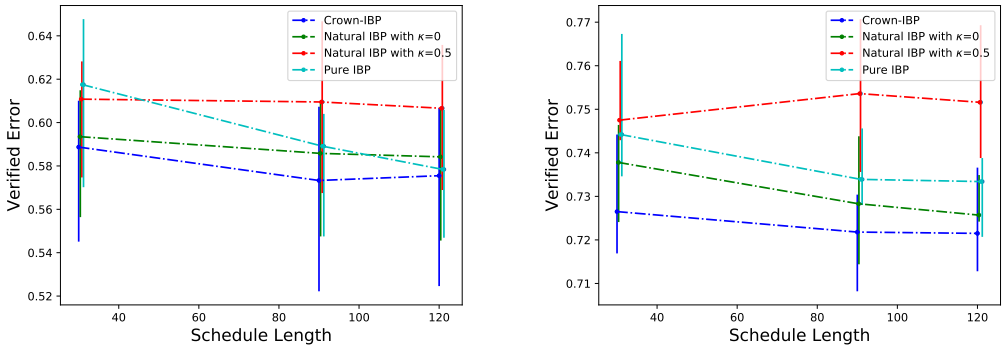
ϵ	error	model A	model B	model C	model D	model E	model F	model G	model H	model I	model J
0.1	std. err. (%)	$2.57 \pm .04$	$1.45 \pm .05$	$3.02 \pm .04$	$1.77 \pm .04$	$2.13 \pm .08$	$1.35 \pm .05$	$2.03 \pm .08$	$1.32 \pm .08$	$1.77 \pm .04$	$1.45 \pm .05$
	verified err. (%)	$6.85 \pm .04$	$4.88 \pm .04$	$6.67 \pm .1$	$5.10 \pm .1$	$4.82 \pm .2$	$4.18 \pm .008$	$5.23 \pm .2$	$4.59 \pm .08$	$5.92 \pm .09$	$5.40 \pm .09$
0.2	std. err. (%)	$3.87 \pm .04$	$2.43 \pm .04$	$4.40 \pm .2$	$2.32 \pm .04$	$3.45 \pm .3$	1.90 ± 0	$2.67 \pm .1$	$2.00 \pm .07$	$2.22 \pm .04$	$1.65 \pm .05$
	verified err. (%)	$12.0 \pm .03$	$6.99 \pm .04$	$10.3 \pm .2$	$7.37 \pm .06$	$9.01 \pm .9$	$6.05 \pm .03$	$7.50 \pm .1$	$6.45 \pm .06$	$7.50 \pm .3$	$6.31 \pm .08$
0.3	std. err. (%)	$5.97 \pm .08$	3.20 ± 0	$6.78 \pm .1$	$3.70 \pm .1$	$3.85 \pm .2$	$3.10 \pm .1$	$4.20 \pm .3$	$2.85 \pm .05$	$3.67 \pm .08$	$2.35 \pm .09$
	verified err. (%)	$15.4 \pm .08$	$10.6 \pm .06$	$16.1 \pm .3$	$11.3 \pm .1$	$11.7 \pm .2$	$9.96 \pm .09$	$12.2 \pm .6$	$9.90 \pm .2$	$11.2 \pm .09$	$9.21 \pm .3$
0.4	std. err. (%)	$8.43 \pm .04$	$4.93 \pm .1$	$8.53 \pm .2$	$5.83 \pm .2$	$5.48 \pm .2$	$4.65 \pm .09$	$6.80 \pm .2$	$4.28 \pm .1$	$5.60 \pm .1$	$3.60 \pm .07$
	verified err. (%)	$24.6 \pm .1$	$18.5 \pm .2$	$24.6 \pm .7$	$19.2 \pm .2$	$18.8 \pm .2$	$17.3 \pm .04$	$20.4 \pm .3$	$16.3 \pm .2$	$18.5 \pm .07$	$15.2 \pm .3$

Table 5: Mean and standard deviation of different CROWN-IBP models' verified and standard error rates on MNIST test set. The architectures of the models are presented in Table 4. We run each model 5 times to compute the mean and standard deviation.

We run CROWN-IBP on 10 small MNIST models, 5 times each, and report the mean and standard deviation of standard and verified errors in Table 5. We can observe that the results from multiple runs are very similar with small standard deviations, so reproducibility is not an issue for CROWN-IBP.

D Training Stability Experiments on CIFAR

Similar to our experiments in Section 4.3, we compare the verified errors obtained by CROWN-IBP, Natural-IBP and Pure-IBP under different ϵ schedule lengths (30, 90, 120). We present the best, worst and median verified errors over all 17 models for CIFAR-10 in Figure 5 and 6, at $\epsilon = \{\frac{2}{255}, \frac{8}{255}\}$. The upper and lower bound of an error bar are the worst and best verified error, respectively, and the lines go through median values. Similar to our observations on MNIST dataset, both Natural-IBP and CROWN-IBP can improve training stability, but CROWN-IBP consistently outperforms other methods. Pure-IBP cannot stably converge on all models when ϵ schedule is short, and verified errors tend to be higher.

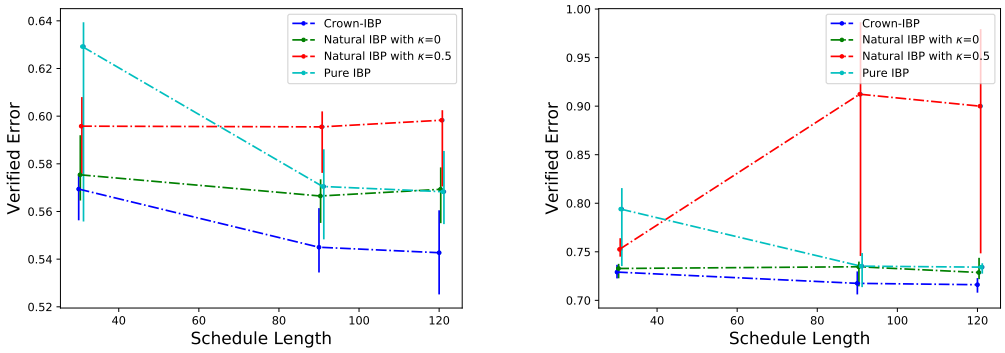


(a) $\epsilon = \frac{2}{255}$, best $\mathbf{52.22}\%$ (b) $\epsilon = \frac{8}{255}$, best $\mathbf{70.82}\%$

Figure 5: Verified error vs. ϵ schedule length (30, 90, 120) on 9 small CIFAR models. The upper and lower bound of an error bar are worst and best verified error, respectively. The dotted lines go through median values.

E Training Time

In Table 6 we present the training time of CROWN-IBP, Pure-IBP and convex adversarial polytope (Wong et al., 2018) on several representative models. All experiments are measured on a single RTX 2080 Ti GPU with 11 GB RAM. We can observe that CROWN-IBP is practically 2 to 7 times slower than Pure-IBP (theoretically, CROWN-IBP is up to $n_L = 10$ times slower than Pure-IBP); convex adversarial polytope (Wong et al., 2018), as a representative linear relaxation based method, can be over hundreds times slower than Pure-IBP especially on deeper networks. Note that we use 50 random Cauchy projections for (Wong et al., 2018). Using random projections alone is not sufficient to scale purely linear relaxation based methods to larger datasets, thus we advocate a combination of non-linear IBP bounds with linear relaxation based methods as in CROWN-IBP,



(a) $\epsilon = \frac{2}{255}$, best $\mathbf{52.52}\%$ (b) $\epsilon = \frac{8}{255}$, best $\mathbf{70.61}\%$

Figure 6: Verified error vs. ϵ schedule length (30, 90, 120) on 8 large CIFAR models. The upper and lower bound of an error bar are worst and best verified error, respectively. The dotted lines go through median values.

which offers good scalability, stability and representation power. We also note that the random projection based acceleration can also be applied to the backward bound propagation (CROWN-style bound) in CROWN-IBP to further speed CROWN-IBP up.

Data	MNIST					CIFAR				
Model Name	A	C	G	L	O	B	D	H	S	M
Pure-IBP (s)	245	264	290	364	1032	734	908	1048	691	1407
CROWN-IBP (s)	423	851	748	1526	7005	1473	3351	2962	1989	6689
Convex adv (Wong et al., 2018) (s)	1708	9263	12649	35518	160794	2372	12688	18691	6961	51145

Table 6: Pure-IBP and CROWN-IBP's training time on different models in seconds. For Pure-IBP and CROWN-IBP, we use a batchsize of 256 for MNIST and 128 for CIFAR. For convex adversarial polytope, we use 50 random Cauchy projections, and reduce batch size if necessary to fit into GPU memory.

F Exact Forms of the CROWN-IBP Backward Bound

CROWN (Zhang et al., 2018) is a general framework that replaces non-linear functions in a neural network with linear upper and lower hyperplanes with respect to pre-activation variables, such that the entire neural network function can be bounded by a linear upper hyperplane and linear lower hyperplane for all $\boldsymbol{x} \in S$ (S is typically a norm bounded ball, or a box region):

$$\underline{\mathbf{A}}x + \underline{\mathbf{b}} \leq f(\boldsymbol{x}) \leq \overline{\mathbf{A}}x + \overline{\mathbf{b}}$$

CROWN achieves such linear bounds by replacing non-linear functions with linear bounds, and utilizing the fact that the linear combinations of linear bounds are still linear, thus these linear bounds can propagate through layers. Suppose we have a non-linear vector function σ, applying to an input (pre-activation) vector z, CROWN requires the following bounds in a general form:

$$\underline{\mathbf{A}}_\sigma z + \underline{\mathbf{b}}_\sigma \leq \sigma(z) \leq \overline{\mathbf{A}}_\sigma z + \overline{\mathbf{b}}_\sigma$$

In general the specific bounds $\underline{\mathbf{A}}_\sigma, \underline{\mathbf{b}}_\sigma, \overline{\mathbf{A}}_\sigma, \overline{\mathbf{b}}_\sigma$ for different σ needs to be given in a case-by-case basis, depending on the characteristics of σ and the preactivation range $\underline{z} \leq z \leq \overline{z}$. In neural network common σ can be ReLU, $\tanh$, sigmoid, maxpool, etc. Convex adversarial polytope (Wong et al., 2018) is also a linear relaxation based techniques that is closely related to CROWN, but only for ReLU layers. For ReLU such bounds are simple, where $\underline{\mathbf{A}}_\sigma, \overline{\mathbf{A}}_\sigma$ are diagonal matrices, $\underline{\mathbf{b}}_\sigma = \mathbf{0}$:

$$\underline{\mathbf{D}}z \leq \sigma(z) \leq \overline{\mathbf{D}}z + \overline{c} \tag{14}$$

where $\underline{\mathbf{D}}$ and $\overline{\mathbf{D}}$ are two diagonal matrices:

$$\underline{\mathbf{D}}_{k,k} = \begin{cases} 1, & \text{if } \underline{z}_k > 0, \text{ i.e., this neuron is always active} \\ 0, & \text{if } \overline{z}_k < 0, \text{ i.e., this neuron is always inactive} \\ \alpha, & \text{otherwise, any } 0 \le \alpha \le 1 \end{cases} \tag{15}$$

$$\overline{\mathbf{D}}_{k,k} = \begin{cases} 1, & \text{if } \underline{z}_k > 0, \text{ i.e., this neuron is always active} \\ 0, & \text{if } \overline{z}_k < 0, \text{ i.e., this neuron is always inactive} \\ \frac{\overline{z}_k}{\overline{z}_k - \underline{z}_k}, & \text{otherwise} \end{cases} \tag{16}$$

$$\overline{c}_k = \begin{cases} 0, & \text{if } \underline{z}_k > 0, \text{ i.e., this neuron is always active} \\ 0, & \text{if } \overline{z}_k < 0, \text{ i.e., this neuron is always inactive} \\ \frac{\overline{z}_k \underline{z}_k}{\overline{z}_k - \underline{z}_k}, & \text{otherwise} \end{cases} \tag{17}$$

Note that CROWN-style bounds require to know all pre-activation bounds $\underline{z}^{(l)}$ and $\overline{z}^{(l)}$. We assume these bounds are valid for $\boldsymbol{x} \in S$. In CROWN-IBP, these bounds are obtained by interval bound propagation (IBP). With pre-activation bounds $\underline{z}^{(l)}$ and $\overline{z}^{(l)}$ given (for $\boldsymbol{x} \in S$), we rewrite the CROWN lower bound for the special case of ReLU neurons:

Theorem F.1 (CROWN Lower Bound)**.** *For a L-layer neural network function $f(\boldsymbol{x}) : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$, $\forall j \in [n_L]$, $\forall \boldsymbol{x} \in S$, we have $\underline{f_j}(\boldsymbol{x}) \le f_j(\boldsymbol{x})$, where*

$$\underline{f_j}(\boldsymbol{x}) = \mathbf{A}^{(0)}_{j,:}\boldsymbol{x} + \sum_{l=1}^{L} \mathbf{A}^{(l)}_{j,:}(\boldsymbol{b}^{(l)} + \underline{\mathbf{b}}^{j,(l)}), \tag{18}$$

$$\mathbf{A}^{(l)}_{j,:} = \begin{cases} \mathbf{e}_j^\top & \text{if } l = L; \\ \mathbf{A}^{(l+1)}_{j,:}\mathbf{W}^{(l+1)}\mathbf{D}^{j,(l)} & \text{if } l \in \{0, \cdots, L-1\}. \end{cases}$$

and $\forall i \in [n_k]$, we define diagonal matrices $\mathbf{D}^{j,(l)}$, bias vector $\underline{\mathbf{b}}^{(l)}$:

$$\mathbf{D}^{j,(0)} = \boldsymbol{I}, \quad \underline{\mathbf{b}}^{j,(L)} = \mathbf{0}$$

$$\mathbf{D}^{j,(l)}_{k,k} = \begin{cases} 1 & \text{if } \mathbf{A}^{(l+1)}_{j,:}\mathbf{W}^{(l+1)}_{:,i} \ge 0, \overline{z}^{(l)}_k > |\underline{z}^{(l)}_k|, \ l \in \{1, \cdots, L-1\}; \\ 0 & \text{if } \mathbf{A}^{(l+1)}_{j,:}\mathbf{W}^{(l+1)}_{:,i} \ge 0, \overline{z}^{(l)}_k < |\underline{z}^{(l)}_k|, \ l \in \{1, \cdots, L-1\}; \\ \frac{\overline{z}^{(l)}_k}{\overline{z}^{(l)}_k - \underline{z}^{(l)}_k} & \text{if } \mathbf{A}^{(k+1)}_{j,:}\mathbf{W}^{(k+1)}_{:,i} < 0, \ l \in \{1, \cdots, L-1\}. \end{cases}$$

$$\underline{\mathbf{b}}^{j,(l)}_k = \begin{cases} 0 & \text{if } \mathbf{A}^{(l+1)}_{j,:}\mathbf{W}^{(l+1)}_{:,i} \ge 0; \ l \in \{1, \cdots, L-1\} \\ \frac{\overline{z}^{(l)}_k \underline{z}^{(l)}_k}{\overline{z}^{(l)}_k - \underline{z}^{(l)}_k} & \text{if } \mathbf{A}^{(l+1)}_{j,:}\mathbf{W}^{(l+1)}_{:,i} < 0 \ l \in \{1, \cdots, L-1\}. \end{cases}$$

$\mathbf{e}_j \in \mathbb{R}^{n_L}$ is a standard unit vector with j-th coordinate set to 1.

Note that unlike the ordinary CROWN (Zhang et al., 2018), in CROWN-IBP we only need the lower bound to compute $\underline{\boldsymbol{m}}$ and do not need to compute the $\mathbf{A}$ matrices for the upper bound. This save half of the computation cost in ordinary CROWN. Also, $\mathbf{W}$ represents any affine layers in a neural network, including convolutional layers in CNNs. In Section 3.3, we discussed how to use transposed convolution operators to efficiently implement CROWN-IBP on GPUs.

Although in this paper we focus on the common case of ReLU activation function, other general activation functions (sigmoid, max-pooling, etc) can be used in the network as CROWN is a general framework to deal with non-linearity. For a more general derivation we refer the readers to (Zhang et al., 2018) and (Salman et al., 2019).