Towards Evaluating and Training Verifiably Robust Neural Networks

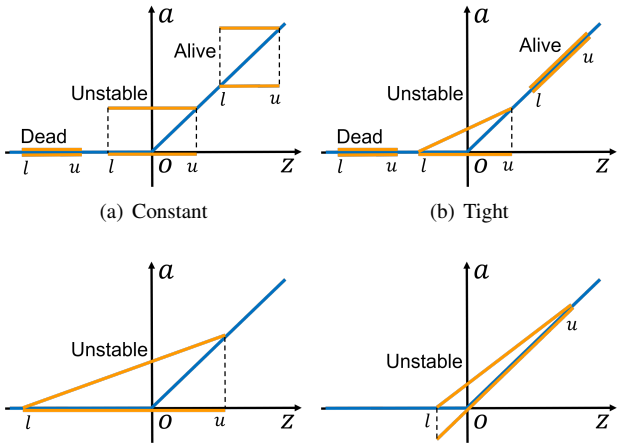
Zhaoyang Lyu[1,2] Minghao Guo[1,2] Tong Wu[1,2] Guodong Xu[1,2] Kehuan Zhang[2] Dahua Lin[1,3]
[1]SenseTime-CUHK Joint Lab [2]The Chinese University of Hong Kong
[3]Centre of Perceptual and Interactive Intelligence
lyuzhaoyang@link.cuhk.edu.hk, {gm019, wt020, xg018, khzhang, dhlin}@ie.cuhk.edu.hk

Abstract

Recent works have shown that interval bound propagation (IBP) can be used to train verifiably robust neural networks. Reseachers observe an intriguing phenomenon on these IBP trained networks: CROWN, a bounding method based on tight linear relaxation, often gives very loose bounds on these networks. We also observe that most neurons become dead during the IBP training process, which could hurt the representation capability of the network. In this paper, we study the relationship between IBP and CROWN, and prove that CROWN is always tighter than IBP when choosing appropriate bounding lines. We further propose a relaxed version of CROWN, linear bound propagation (LBP), that can be used to verify large networks to obtain lower verified errors than IBP. We also design a new activation function, parameterized ramp function (ParamRamp), which has more diversity of neuron status than ReLU. We conduct extensive experiments on MNIST, CIFAR-10 and Tiny-ImageNet with ParamRamp activation and achieve state-of-the-art verified robustness. Code is available at https://github.com/ZhaoyangLyu/VerifiablyRobustNN.

1. Introduction

Deep neural networks achieve state-of-the-art performance in many tasks, *e.g.*, image classification, object detection, and instance segmentation, but they are vulnerable to adversarial attacks. A small perturbation that is imperceptible to humans can mislead a neural network's prediction [22, 4, 1, 13, 3]. To mitigate this problem, Madry *et al*. [15] develop an effective framework to train robust neural networks. They formulate adversarial training as a robust optimization problem. Specifically, they use projected gradient descent (PGD) to find the worst-case adversarial example near the original image and then minimize the loss at this point during training. Networks trained under this framework achieve state-of-the-art robustness under many attacks [34, 26, 18]. However, these networks are only emperically robust, but not verifiably robust. They become vulnerable when stronger attacks are presented [24, 6, 23].



(a) Constant (b) Tight

(c) Adaptive: Case $|l| > u$ (d) Adaptive: Case $|l| \leq u$

Figure 1. Illustration of different strategies to choose bounding lines for the three status of a ReLU neuron. Dead: $l \leq u \leq 0$; Unstable: $l < 0 < u$; Alive: $0 \leq l \leq u$. $[l, u]$ is the input range of the neuron. (a) chooses constant bounding lines. (b) is the tight strategy. (c) and (d) are the two cases of unstable neurons in the adaptive strategy. The adaptive strategy chooses the same bounding lines as the tight strategy for dead and alive neurons. See more details in Appendix A.7.

This leads to the development of robustness verification, which aims to provide a certificate that a neural network gives consistent predictions for all inputs in some set, usually an l_p ball around a clean image. The key of robustness verification is to compute the lower and upper bounds of the output logits when input can take any value in the l_p ball. The exact bounds can be computed through Satisfiability Modulo Theory [11] or solving a Mixed Integer Linear Programming (MILP) problem [23, 5]. Relaxed bounds can be obtained by reduce the bound computation problem to a linear programming (LP) problem [28] or a semidefinite programming (SDP) problem [7]. However, these programming based methods are expensive and difficult to scale to large networks. To this end, another approach that makes linear relaxations of the nonlinear activation functions in a network is proposed [20, 21, 25, 27, 33, 12]. Figure 1 illustrates different strategies to make linear relaxations of a

ReLU neuron. These methods can compute bounds analytically and efficiently. In this paper, we focus on the study of CROWN [33], which can *compute relatively tight bounds while being fast*. Other similar approaches [20, 21, 25, 27] are either a special case of CROWN or a different view of it as demonstrated by Salman *et al*. [19].

Wong *et al*. [28] propose to incorporate bounds computed by the aforementioned linear relaxation based methods in the loss function to train verifiably robust networks. Similar approaches are proposed in several other works [16, 8, 17, 24]. However, these methods generally bring heavy computational overhead to the original training process. Gowal *et al*. [9] propose to use a simple technique, interval bound propagation (IBP), to compute bounds. IBP is fast and can scale to large networks. Despite being loose, IBP outperforms previous linear relaxation based methods in terms of training verifiably robust networks. Zhang *et al*. [32] further improve this method by combining IBP with the tighter linear relaxation based method, CROWN. The resulting method is named CROWN-IBP. They use CROWN-IBP to compute bounds at the initial training phase and achieve the lowest l_∞ verified errors.

We notice that both IBP trained networks [9] and CROWN-IBP trained networks [32] are verified by IBP after training. One natural question is whether we can use tighter linear relaxation based methods to verify the networks to achieve lower verified error. Surprisingly, Zhang *et al*. [32] find the typically much tighter method, CROWN, gives very loose bounds for IBP trained networks. It seems that IBP trained networks have very different verification properties from normally trained networks. We also find that CROWN cannot verify large networks due to its high memory cost. Another phenomenon we observe on IBP and CROWN-IBP trained networks is that most neurons become dead during training. We believe that this could restrict the representation capability of the network and thus hurt its performance. In this paper, we make the following contributions to tackle the aforementioned problems:

1. We develop a relaxed version of CROWN, linear bound propagation (LBP), which has better scalability. We demonstrate LBP can be used to obtain tighter bounds than IBP on both normally trained networks or IBP trained networks.
2. We prove IBP is a special case of CROWN and LBP. The reason that CROWN gives looser bounds than IBP on IBP trained networks is that CROWN chooses bad bounding lines when making linear relaxations of the nonlinear activation functions. We prove CROWN and LBP are always tighter than IBP if they adopt the tight strategy to choose bounding lines as shown in Figure 1.
3. We propose to use a new activation function, parameterized ramp function (ParamRamp), to train verifiably robust networks. Compared with ReLU, where most neurons become dead during training, ParamRamp brings more diversity of neuron status. Our experiments demonstrate networks with ParamRamp activation achieve state-of-the-art verified l_∞ robustness on MNIST, CIFAR-10 and Tiny-ImageNet.

2. Background and Related Work

In this section, we start by giving definition of an m-layer feed-forward neural network and then briefly introduce the concept of robustness verification. Next we present interval bound propagation, which is used to train networks with best verified errors. Finally we review two state-of-the-art verifiable adversarial training methods [9, 32] that are most related to our work.

Definition of an m-layer feed-forward network.

$$\mathbf{z}^{(k)} = \mathbf{W}^{(k)}\mathbf{a}^{(k-1)} + \mathbf{b}^{(k)}, \mathbf{a}^{(k)} = \sigma(\mathbf{z}^{(k)}), \quad k = 1, 2, \cdots, m. \tag{1}$$

$\mathbf{W}^{(k)}, \mathbf{b}^{(k)}, \mathbf{a}^{(k)}, \mathbf{z}^{(k)}$ are the weight matrix, bias, activation and pre-activation of the k-th layer in the network, respectively. σ is the elementwise activation function. Note that we always assume σ is a monotonic increasing function in rest part of the paper. $\mathbf{a}^{(0)} = \mathbf{x}$ and $\mathbf{z}^{(m)}$ are the input and output of the network. We also use n_k to denote the number of neurons in the k-th layer and n_0 is the dimension of the input. Although this network only contains fully connected layers, our discussions on this network in rest part of the paper readily generalize to convolutional layers as they are essentially a linear transformation as well [2].

Robustness verification. Robustness verification aims to guarantee a neural network gives consistent predictions for all inputs in some set, typically an l_p ball around the original input: $\mathbb{B}_p(\mathbf{x}_0, \epsilon) = \{\mathbf{x} \,|\, ||\mathbf{x} - \mathbf{x}_0||_p \leq \epsilon\}$, where $\mathbf{x}_0$ is the clean image. The key step is to compute the lower and upper bounds of the output logits $\mathbf{z}^{(m)}$ (or the lower bound of the margin between ground truth class and other classes as defined in (4)) when the input can take any value in $\mathbb{B}_p(\mathbf{x}_0, \epsilon)$. We can guarantee that the network gives correct predictions for all inputs in $\mathbb{B}_p(\mathbf{x}_0, \epsilon)$ if the lower bound of the ground truth class is larger than the upper bounds of all the other classes (or the lower bound of the margin is greater than 0). The verified robustness of a network is usually measured by the verified error: The percentage of images that we can not guarantee that the network always gives correct predictions for inputs in $\mathbb{B}_p(\mathbf{x}_0, \epsilon)$. Note that the verified error not only depends on the network and the allowed perturbation of the input, but also the method we use to compute bounds for the output. CROWN and IBP are the two bounding techniques that are most related to our work. We briefly walk through CROWN in Section 3 and introduce IBP right below.

Interval bound propagation. Assume we know the lower and upper bounds of the activation of the $(k-1)$-th layer: $\hat{\mathbf{l}}^{(k-1)} \leq \mathbf{a}^{(k-1)} \leq \hat{\mathbf{u}}^{(k-1)}$. Then IBP computes

bounds of $\mathbf{z}^{(k)}$, $\mathbf{l}^{(k)}$ and $\mathbf{u}^{(k)}$, in the following way:

$$\begin{aligned}\mathbf{l}^{(k)} &= relu(\mathbf{W}^{(k)})\hat{\mathbf{l}}^{(k-1)} + neg(\mathbf{W}^{(k)})\hat{\mathbf{u}}^{(k-1)} + \mathbf{b}^{(k)},\\ \mathbf{u}^{(k)} &= relu(\mathbf{W}^{(k)})\hat{\mathbf{u}}^{(k-1)} + neg(\mathbf{W}^{(k)})\hat{\mathbf{l}}^{(k-1)} + \mathbf{b}^{(k)},\end{aligned} \tag{2}$$

where $relu$ is the elementwise ReLU function and neg is the elementwise version of the function $neg(x) = x$, if $x \leq 0$; $neg(x) = 0$, else. Next, bounds of $\mathbf{a}^{(k)}$, $\hat{\mathbf{l}}^{(k)}$ and $\hat{\mathbf{u}}^{(k)}$, can be computed by

$$\hat{\mathbf{l}}^{(k)} = \sigma(\mathbf{l}^{(k)}), \hat{\mathbf{u}}^{(k)} = \sigma(\mathbf{u}^{(k)}). \tag{3}$$

IBP repeats the above procedure from the first layer and computes bounds layer by layer until the final output as shown in Figure 2(b). Bounds of $\mathbf{a}^{(0)} = \mathbf{x}$ is known if the allowed perturbation is in an l_∞ ball. Closed form bounds of $\mathbf{z}^{(1)}$ can be computed using Holder's inequality as shown in (13) if the allowed perturbation is in a general l_p ball.

Train Verifiably Robust Networks. Verifiable adversarial training first use some robustness verification method to compute a lower bound $\mathbf{l}^{\boldsymbol{\omega}}$ of the margin $\boldsymbol{\omega}$ between ground truth class y and other classes:

$$\begin{aligned}&\boldsymbol{\omega}_i(\mathbf{x}) = \mathbf{z}_y^{(m)}(\mathbf{x}) - \mathbf{z}_i^{(m)}(\mathbf{x}), i = 1, 2, \cdots, n_m.\\ &\mathbf{l}^{\boldsymbol{\omega}}(\mathbf{x}_0, \epsilon) \leq \boldsymbol{\omega}(\mathbf{x}), \forall \mathbf{x} \in \mathbb{B}_p(\mathbf{x}_0, \epsilon).\end{aligned} \tag{4}$$

Here we use "$\leq$" to denote elementwise less than or equal to. For simplicity, we won't differentiate operators between vectors and scalars in rest part of the paper when no ambiguity is caused. Gowal *et al.* [9] propose to use IBP to compute the lower bound $\mathbf{l}^{\boldsymbol{\omega}}_{\text{IBP}}(\mathbf{x}_0, \epsilon)$ and minimize the following loss during training:

$$\mathop{\mathbb{E}}_{(\mathbf{x}_0, y)\in\mathcal{X}} \kappa L(\mathbf{z}^{(m)}(\mathbf{x}_0), y) + (1-\kappa)L(-\mathbf{l}^{\boldsymbol{\omega}}_{\text{IBP}}(\mathbf{x}_0, \epsilon), y), \tag{5}$$

where $\mathcal{X}$ is the underlying data distribution, κ is a hyper parameter to balance the two terms of the loss, and L is the normal cross-entropy loss. This loss encourages the network to maximize the margin between ground truth class and other classes. Zhang *et al.* [32] argue that IBP bound is loose during the initial phase of training, which makes training unstable and hard to tune. They propose to use a convex combination of the IBP bound $\mathbf{l}^{\boldsymbol{\omega}}_{\text{IBP}}$ and CROWN-IBP bound $\mathbf{l}^{\boldsymbol{\omega}}_{\text{C.-IBP}}$ as the lower bound to provide supervision at the initial phase of training:

$$\mathbf{l}^{\boldsymbol{\omega}} = (1-\beta)\mathbf{l}^{\boldsymbol{\omega}}_{\text{IBP}} + \beta\mathbf{l}^{\boldsymbol{\omega}}_{\text{C.-IBP}}. \tag{6}$$

The loss they use is the same as the one in (5) except for replacing $\mathbf{l}^{\boldsymbol{\omega}}_{\text{IBP}}$ with the new $\mathbf{l}^{\boldsymbol{\omega}}$ defined in (6). They design a schedule for β: It starts from 1 and decreases to 0 during training. Their approach achieves state-of-the-art verified errors on MNIST and CIFAR-10 datasets. Xu *et al.* [30] propose a loss fusion technique to speed up the training process of CROWN-IBP and this enables them to train large networks on large datasets such as Tiny-ImageNet and Downscaled ImageNet.

3. Relaxed CROWN

CROWN is considered an efficient robustness verification method compared with LP based methods [27, 33, 14], but these works only test CROWN on small multi-layer perceptrons with at most several thousand neurons in each hidden layer. Our experiment suggests that CROWN scales badly to large convolutional neural networks (CNNs): It consumes more than 12 GB memory when verifying a single image from CIFAR-10 for a small 4-layer CNN (See its detailed structure in Appendix B.1), which prevents it from utilizing modern GPUs to speed up computation. Therefore, it is crucial to improve CROWN's scalability to employ it on large networks. To this end, we develop a relaxed version of CROWN named **Linear Bound Propagation (LBP)**, whose computation complexity and memory cost grow linearly with the size of the network. We first walk through the deduction process of the original CROWN.

The original CROWN. Suppose we want to compute lower bound for the quantity $\mathbf{W}^{obj}\mathbf{z}^{(k)} + \mathbf{b}^{obj}$. $\mathbf{W}^{obj}$ and $\mathbf{b}^{obj}$ are the weight and bias that connect $\mathbf{z}^{(k)}$ to the quantity of interests. For example, the quantity becomes the margin $\boldsymbol{\omega}(\mathbf{x})$ if we choose appropriate $\mathbf{W}^{obj}$ and set $\mathbf{b}^{obj} = 0, k = m$. Assume we already know the bounds of pre-activation of the $(k-1)$-th layer:

$$\mathbf{l}^{(k-1)} \leq \mathbf{z}^{(k-1)} \leq \mathbf{u}^{(k-1)}, \forall \mathbf{x} \in \mathbb{B}_p(\mathbf{x}_0, \epsilon). \tag{7}$$

Next CROWN finds two linear functions of $\mathbf{z}^{(k-1)}$ to bound $\mathbf{a}^{(k-1)} = \sigma(\mathbf{z}^{(k-1)})$ in the intervals determined by $\mathbf{l}^{(k-1)}, \mathbf{u}^{(k-1)}$.

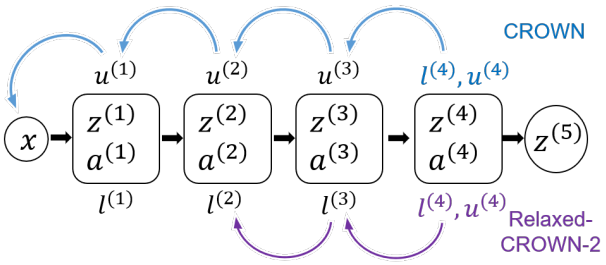
$$\begin{aligned}&\mathbf{h}^{(k-1)L}(\mathbf{z}^{(k-1)}) \leq \sigma(\mathbf{z}^{(k-1)}) \leq \mathbf{h}^{(k-1)U}(\mathbf{z}^{(k-1)}),\\ &\forall \mathbf{l}^{(k-1)} \leq \mathbf{z}^{(k-1)} \leq \mathbf{u}^{(k-1)},\end{aligned} \tag{8}$$

where

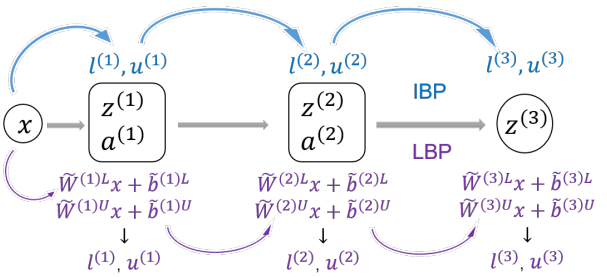
$$\begin{aligned}\mathbf{h}^{(k-1)L}(\mathbf{z}^{(k-1)}) &= \mathbf{s}^{(k-1)L} * \mathbf{z}^{(k-1)} + \mathbf{t}^{(k-1)L},\\ \mathbf{h}^{(k-1)U}(\mathbf{z}^{(k-1)}) &= \mathbf{s}^{(k-1)U} * \mathbf{z}^{(k-1)} + \mathbf{t}^{(k-1)U}.\end{aligned} \tag{9}$$

Here we use "$*$" to denote elementwise product. $\mathbf{s}^{(k-1)L/U}, \mathbf{t}^{(k-1)L/U}$ are constant vectors of the same dimension of $\mathbf{z}^{(k-1)}$. We use L, U in the superscripts to denote quantities related to lower bounds and upper bounds, respectively. We also use L/U in the superscripts to denote "lower bounds or upper bounds". The linear functions $\mathbf{h}^{(k-1)L/U}(\mathbf{z}^{(k-1)})$ are also called bounding lines, as they bound the nonlinear function $\sigma(\mathbf{z}^{(k-1)})$ in the intervals determined by $\mathbf{l}^{(k-1)}, \mathbf{u}^{(k-1)}$. See Figure 1 for a visualization of different strategies to choose bounding lines. Next CROWN utilizes these bounding lines to build a linear function of $\mathbf{z}^{(k-1)}$ to lower bound $\mathbf{W}^{obj}\mathbf{z}^{(k)} + \mathbf{b}^{obj}$:

$$\mathbf{W}^{obj}\mathbf{z}^{(k)} + \mathbf{b}^{obj} \geq \mathbf{W}^{(k,k-1)L}\mathbf{z}^{(k-1)} + \mathbf{b}^{(k,k-1)L}. \tag{10}$$



(a) CROWN vs Relaxed-CROWN-2



(b) IBP vs LBP

Figure 2. Illustration of CROWN, Relaxed-CROWN, IBP and LBP. (a) shows how CROWN and Relaxed-CROWN-2 compute bounds for the 4-th layer of a 5 layer network. (b) shows how IBP and LBP compute bounds layer by layer for a 3 layer network.

See the detailed formulas of $\mathbf{W}^{(k,k-1)L}, \mathbf{b}^{(k,k-1)L}$ in Appendix A.1. In the same manner, CROWN builds a linear function of $\mathbf{z}^{(k-2)}$ to lower bound $\mathbf{W}^{(k,k-1)L}\mathbf{z}^{(k-1)} + \mathbf{b}^{(k,k-1)L}$ if bounds of $\mathbf{z}^{(k-2)}$ are known. CROWN repeats this procedure: It back-propagates layer by layer until the first layer $\mathbf{z}^{(1)}$ as shown in Figure 2(a):

$$\mathbf{W}^{obj}\mathbf{z}^{(k)} + \mathbf{b}^{obj} \geq \mathbf{W}^{(k,k-1)L}\mathbf{z}^{(k-1)} + \mathbf{b}^{(k,k-1)L} \geq \cdots \\ \mathbf{W}^{(k,k-2)L}\mathbf{z}^{(k-2)} + \mathbf{b}^{(k,k-2)L} \geq \mathbf{W}^{(k,1)L}\mathbf{z}^{(1)} + \mathbf{b}^{(k,1)L}. \tag{11}$$

Notice $\mathbf{z}^{(1)} = \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}$. We plug it in the last term of (11) and obtain a linear function of $\mathbf{x}$.

$$\mathbf{W}^{obj}\mathbf{z}^{(k)} + \mathbf{b}^{obj} \geq \tilde{\mathbf{W}}^{(k)L}\mathbf{x} + \tilde{\mathbf{b}}^{(k)L}, \tag{12}$$

where $\tilde{\mathbf{W}}^{(k)L} = \mathbf{W}^{(k,1)L}\mathbf{W}^{(1)}, \tilde{\mathbf{b}}^{(k)L} = \mathbf{W}^{(k,1)L}\mathbf{b}^{(1)} + \mathbf{b}^{(k,1)L}$. Now we can compute the closed-form lower bound of $\mathbf{W}^{obj}\mathbf{z}^{(k)} + \mathbf{b}^{obj}$ through Holder's inequality:

$$\mathbf{W}^{obj}\mathbf{z}^{(k)} + \mathbf{b}^{obj} \geq \tilde{\mathbf{W}}^{(k)L}\mathbf{x} + \tilde{\mathbf{b}}^{(k)L} \geq \\ \tilde{\mathbf{W}}^{(k)L}\mathbf{x}_0 + \tilde{\mathbf{b}}^{(k)L} - \epsilon||\tilde{\mathbf{W}}^{(k)L}||_q, \forall\, \mathbf{x} \in \mathbb{B}_p(\mathbf{x}_0, \epsilon), \tag{13}$$

where $1/p + 1/q = 1$ and $||\tilde{\mathbf{W}}^{(k)L}||_q$ denotes a column vector that is composed of the q-norm of every row in $\tilde{\mathbf{W}}^{(k)L}$. We can compute a linear function of $\mathbf{x}$ to upper bound $\mathbf{W}^{obj}\mathbf{z}^{(k)} + \mathbf{b}^{obj}$ in the same manner and then compute its closed-form upper bound. See details in Appendix A.1.

Let's review the process of computing bounds for $\mathbf{W}^{obj}\mathbf{z}^{(k)} + \mathbf{b}^{obj}$. It requires us to know the bounds of the previous $(k-1)$ layers: $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \cdots, \mathbf{z}^{(k-1)}$. We can fulfill this requirement by starting computing bounds from the first layer $\mathbf{z}^{(1)}$, and then computing bounds layer by layer in a forward manner until the $(k-1)$-th layer. Therefore, the computation complexity of CROWN is of the order $\mathcal{O}(m^2)$. And its memory cost is of the order $\mathcal{O}(\max_{k\neq v} n_k n_v)$, where $0 \leq k, v \leq m$, and n_k is the number of neurons in the k-th layer. This is because we need to record a weight matrix $\mathbf{W}^{(k,v)}$ between any two layers as shown in (11). This makes CROWN difficult to scale to large networks. To this end, we propose a relaxed version of CROWN in the next paragraph.

Relaxed CROWN. As the same in the above CROWN deduction process, suppose we want to compute bounds for the quantity $\mathbf{W}^{obj}\mathbf{z}^{(k)} + \mathbf{b}^{obj}$. In the original CROWN process, we first compute linear functions of $\mathbf{x}$ to bound the pre-activation of the first $(k-1)$ layers:

$$\tilde{\mathbf{W}}^{(v)L}\mathbf{x} + \tilde{\mathbf{b}}^{(v)L} \leq \mathbf{z}^{(v)} \leq \tilde{\mathbf{W}}^{(v)U}\mathbf{x} + \tilde{\mathbf{b}}^{(v)U}, \\ \forall\, \mathbf{x} \in \mathbb{B}_p(\mathbf{x}_0, \epsilon), v = 1, 2, \cdots, k-1, \tag{14}$$

and use these linear functions of $\mathbf{x}$ to compute closed-form bounds for the first $(k-1)$ layers. We argue that in the back-propagation process in (11), we don't need to back-propagate to the first layer. We can stop at any intermediate layer and plug in the linear functions in (14) of this intermediate layer to get a linear function of $\mathbf{x}$ to bound $\mathbf{W}^{obj}\mathbf{z}^{(k)} + \mathbf{b}^{obj}$. Specifically, assume we decide to back-propagate v layers:

$$\mathbf{W}^{obj}\mathbf{z}^{(k)} + \mathbf{b}^{obj} \geq \mathbf{W}^{(k,k-1)L}\mathbf{z}^{(k-1)} + \mathbf{b}^{(k,k-1)L} \\ \geq \cdots \geq \mathbf{W}^{(k,k-v)L}\mathbf{z}^{(k-v)} + \mathbf{b}^{(k,k-v)L}, v < k. \tag{15}$$

We already know

$$\tilde{\mathbf{W}}^{(k-v)L}\mathbf{x} + \tilde{\mathbf{b}}^{(k-v)L} \leq \mathbf{z}^{(k-v)} \leq \tilde{\mathbf{W}}^{(k-v)U}\mathbf{x} + \tilde{\mathbf{b}}^{(k-v)U}.$$

We can directly plug it to (15) to obtain a lower bound of $\mathbf{W}^{obj}\mathbf{z}^{(k)} + \mathbf{b}^{obj}$:

$$\mathbf{W}^{obj}\mathbf{z}^{(k)} + \mathbf{b}^{obj} \geq \\ relu(\mathbf{W}^{(k,k-v)L})[\tilde{\mathbf{W}}^{(k-v)L}\mathbf{x} + \tilde{\mathbf{b}}^{(k-v)L}] + \mathbf{b}^{(k,k-v)L} \\ + neg(\mathbf{W}^{(k,k-v)L})[\tilde{\mathbf{W}}^{(k-v)U}\mathbf{x} + \tilde{\mathbf{b}}^{(k-v)U}]. \tag{16}$$

Now the last line of (16) is already a linear function of $\mathbf{x}$ and we can compute the closed-form lower bound of $\mathbf{W}^{obj}\mathbf{z}^{(k)} + \mathbf{b}^{obj}$ in the same manner as shown in (13). The upper bound of $\mathbf{W}^{obj}\mathbf{z}^{(k)} + \mathbf{b}^{obj}$ can also be computed by back-propagating only v layers in the same gist.

We have shown we can only back-propagate v layers, instead of back-propagating to the first layer, when computing bounds for the k-th layer. In fact, we can only back-propagate v layers when computing bounds for any layer. If the layer index k is less than or equal to v, we just

back-propagate to the first layer. In other words, we back-propagate at most v layers when computing bounds for any layer in the process of CROWN. We call this relaxed version of CROWN **Relaxed-CROWN-v**. See a comparison of CROWN and Relaxed-CROWN in Figure 2(a).

Linear Bound Propagation. We are particularly interested in the special case of Relaxed-CROWN-1, namely, we only back-propagate 1 layer in the process of CROWN. This leads us to the following theorem.

Theorem 1 *Assume we already know two linear functions of* $\mathbf{x}$ *to bound* $\mathbf{z}^{(k-1)}$*:*

$$\tilde{\mathbf{W}}^{(k-1)L}\mathbf{x}+\tilde{\mathbf{b}}^{(k-1)L} \leq \mathbf{z}^{(k-1)} \leq \tilde{\mathbf{W}}^{(k-1)U}\mathbf{x}+\tilde{\mathbf{b}}^{(k-1)U}.$$

We then compute the closed-form bounds $\mathbf{l}^{(k-1)}, \mathbf{u}^{(k-1)}$ *of* $\mathbf{z}^{(k-1)}$ *using these two linear functions, and choose two linear functions* $\mathbf{h}^{(k-1)L}(\mathbf{z}^{(k-1)}), \mathbf{h}^{(k-1)U}(\mathbf{z}^{(k-1)})$ *to bound* $\mathbf{a}^{(k-1)} = \sigma(\mathbf{z}^{(k-1)})$ *as shown in (9). Then under the condition that* $\mathbf{s}^{(k-1)L} \geq 0, \mathbf{s}^{(k-1)U} \geq 0$*,* $\mathbf{z}^{(k)} = \mathbf{W}^{(k)}\sigma(\mathbf{z}^{(k-1)}) + \mathbf{b}^{(k)}$ *can be bounded by*

$$\tilde{\mathbf{W}}^{(k)L}\mathbf{x}+\tilde{\mathbf{b}}^{(k)L} \leq \mathbf{z}^{(k)} \leq \tilde{\mathbf{W}}^{(k)U}\mathbf{x}+\tilde{\mathbf{b}}^{(k)U}, \tag{17}$$

where

$$\begin{aligned}
\tilde{\mathbf{W}}^{(k)L} &= [relu(\mathbf{W}^{(k)}) * \mathbf{s}^{(k-1)L}]\tilde{\mathbf{W}}^{(k-1)L} \\
&\quad + [neg(\mathbf{W}^{(k)}) * \mathbf{s}^{(k-1)U}]\tilde{\mathbf{W}}^{(k-1)U}, \\
\tilde{\mathbf{b}}^{(k)L} &= \mathbf{b}^{(k)} + [relu(\mathbf{W}^{(k)}) * \mathbf{s}^{(k-1)L}]\tilde{\mathbf{b}}^{(k-1)L} \\
&\quad + [neg(\mathbf{W}^{(k)}) * \mathbf{s}^{(k-1)U}]\tilde{\mathbf{b}}^{(k-1)U} \\
&\quad + relu(\mathbf{W}^{(k)})\mathbf{t}^{(k-1)L} + neg(\mathbf{W}^{(k)})\mathbf{t}^{(k-1)U},
\end{aligned} \tag{18}$$

where the operator "$*$" between a matrix $\mathbf{W}$ *and a vector* $\mathbf{s}$ *is defined as* $(\mathbf{W} * \mathbf{s})_{ij} = \mathbf{W}_{ij}\mathbf{s}_j$.

We refer readers to Appendix A.3 for the formulas of $\tilde{\mathbf{W}}^{(k)U}, \tilde{\mathbf{b}}^{(k)U}$ and the proof of Theorem 1. Note that the condition $\mathbf{s}^{(k-1)L} \geq 0, \mathbf{s}^{(k-1)U} \geq 0$ in Theorem 1 is not necessary. We impose this condition because it simplifies the expressions of $\tilde{\mathbf{W}}^{(k)L}, \tilde{\mathbf{b}}^{(k)L}$, and it generally holds true when people choose bounding lines.

The significance of Theorem 1 is that it allows us to compute bounds starting from the first layer $\mathbf{z}^{(1)}$, which can be bounded by $\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)} \leq \mathbf{z}^{(1)} \leq \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}$, and then compute bounds layer by layer in a forward manner until the final output just like IBP. The computation complexity is reduced to $\mathcal{O}(m)$ and memory cost is reduced to $\mathcal{O}(n_0 \max\{n_1, n_2, \cdots, n_m\})$, since we only need to record a matrix $\tilde{\mathbf{W}}^{(k)}$ from the input $\mathbf{x}$ to every intermediate layer $\mathbf{z}^{(k)}$. We call this method **Linear Bound Propagation (LBP)**, which is equivalent to Relaxed-CROWN-1. See a comparison of LBP and IBP in Figure 2(b). As expected, there is no free lunch. As we will show in the next section, the reduction of computation and memory cost of LBP makes it less tight than CROWN. Although developed from a different perspective, we find LBP similar to the forward mode in the work [30]. See a detailed comparison between them in Appendix A.3.

Zhang *et al*. [32] propose to compute bounds for the first $(m-1)$ layers using IBP and then use CROWN to compute bounds for the last layer to obtain tighter bounds of the last layer. The resulting method is named CROWN-IBP. In the same gist, we can use LBP to compute bounds for the first $(m-1)$ layers and then use CROWN to compute bounds for the last layer. We call this method **CROWN-LBP**.

4. Relationship of IBP, LBP and CROWN

In Section 3, we develop a relaxed version of CROWN, LBP. In this section, we study the relationship between IBP, LBP and CROWN, and investigate why CROWN gives looser bounds than IBP on IBP trained networks [32].

First, we manage to prove IBP is a special case of CROWN and LBP where the bounding lines are chosen as constants as shown in Figure 1(a):

$$\begin{gathered}
\mathbf{h}^{(k)L}(\mathbf{z}^{(k)}) = \sigma(\mathbf{l}^{(k)}), \mathbf{h}^{(k)U}(\mathbf{z}^{(k)}) = \sigma(\mathbf{u}^{(k)}), \\
k = 1, 2, \cdots, m-1.
\end{gathered} \tag{19}$$

In other words, CROWN and LBP degenerate to IBP when they choose constant bounding lines for every neuron in every layer. See the proof of this conclusion in Appendix A.5. On the other hand, Lyu *et al*. [14] prove tighter bounding lines lead to tighter bounds in the process of CROWN, where $\tilde{\mathbf{h}}_i^{(k)L/U}(\mathbf{z}_i^{(k)})$ is defined to be tighter than $\hat{\mathbf{h}}_i^{(k)L/U}(\mathbf{z}_i^{(k)})$ in the interval $[\mathbf{l}_i^{(k)}, \mathbf{u}_i^{(k)}]$ if

$$\begin{gathered}
\hat{\mathbf{h}}_i^{(k)L}(\mathbf{z}_i^{(k)}) \leq \tilde{\mathbf{h}}_i^{(k)L}(\mathbf{z}_i^{(k)}), \tilde{\mathbf{h}}_i^{(k)U}(\mathbf{z}_i^{(k)}) \leq \hat{\mathbf{h}}_i^{(k)U}(\mathbf{z}_i^{(k)}), \\
\forall \mathbf{z}_i^{(k)} \in [\mathbf{l}_i^{(k)}, \mathbf{u}_i^{(k)}].
\end{gathered} \tag{20}$$

We manage to prove it is also true for LBP in Appendix A.3. Therefore, if CROWN and LBP adopt the tight strategy in Figure 1(b) to choose bounding lines, which is guaranteed to be tighter than the constant bounding lines in a specified interval, CROWN and LBP are guaranteed to give tighter bounds than IBP. We formalize this conclusion and include conclusions for CROWN-IBP and CROWN-LBP in the following theorem.

Theorem 2 *Assume the closed-form bounds of the last layer computed by IBP, CROWN-IBP, LBP, CROWN-LBP, and CROWN are* $\mathbf{l}_I^{(m)}, \mathbf{u}_I^{(m)}$*;* $\mathbf{l}_{CI}^{(m)}, \mathbf{u}_{CI}^{(m)}$*;* $\mathbf{l}_L^{(m)}, \mathbf{u}_L^{(m)}$*;* $\mathbf{l}_{CL}^{(m)}, \mathbf{u}_{CL}^{(m)}$*;* $\mathbf{l}_C^{(m)}, \mathbf{u}_C^{(m)}$*, respectively. And CROWN-IBP, LBP, CROWN-LBP, CROWN adopt the tight strategy to choose bounding lines as shown in Figure 1(b). Then we have*

$$\begin{aligned}
\mathbf{l}_I^{(m)} &\leq \{\mathbf{l}_L^{(m)}, \mathbf{l}_{CI}^{(m)}\} \leq \mathbf{l}_{CL}^{(m)} \leq \mathbf{l}_C^{(m)}, \\
\mathbf{u}_I^{(m)} &\geq \{\mathbf{u}_L^{(m)}, \mathbf{u}_{CI}^{(m)}\} \geq \mathbf{u}_{CL}^{(m)} \geq \mathbf{u}_C^{(m)},
\end{aligned} \tag{21}$$

where the sets in the inequalities mean that the inequalities hold true for any element in the sets.

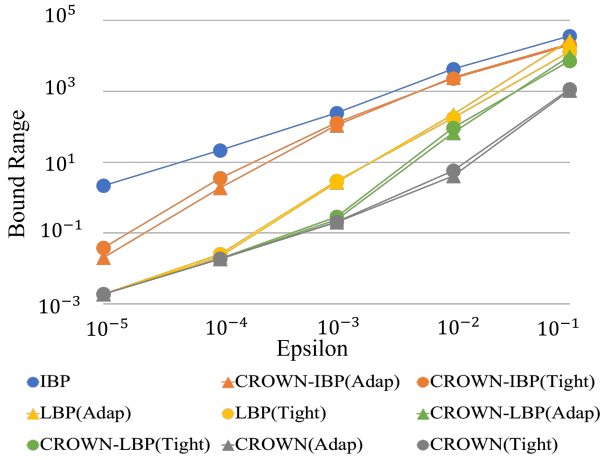


Figure 3. Tightness comparison of IBP, CROWN-IBP, LBP, CROWN-LBP, CROWN on a normally trained MNIST classifier. "Bound Range" is mean of $\mathbf{u}^{(m)} - \mathbf{l}^{(m)}$. The mean is taken over the 10 output logits and averaged over 100 test images in MNIST. "Epsilon" is the radius of the l_∞ ball. "Adap" and "Tight" are the adaptive and tight strategies as shown in Figure 1.

Table 1. Mean lower bound of the margin defined in (4) and verified errors obtained by IBP, CROWN-IBP(C.-IBP), LBP, CROWN-LBP(C.-LBP) on an IBP trained CIFAR-10 classifier. The network is trained with $\epsilon = 8.8/255$ and tested with $\epsilon = 8/255$ (l_∞ norm). Results are taken over 100 test images.

Adaptive	IBP	C.-IBP	LBP	C.-LBP
Verified Err(%)	70.10	85.66	100	99.99
Lower Bound	2.1252	-12.016	-2.4586E5	-1.5163E5
Tight	IBP	C.-IBP	LBP	C.-LBP
Verified Err(%)	70.10	70.01	70.05	69.98
Lower Bound	2.1252	2.1520	2.1278	2.1521

See proof of Theorem 2 in Appendix A.6. Now we can answer the question proposed at the beginning of this section. The reason that CROWN gives looser bounds than IBP [32] is because CROWN uses the adaptive strategy as shown in Figure 1(c) and 1(d) to choose bounding lines by default. The lower bounding line chosen in the adaptive strategy for an unstable neuron is not always tighter than the one chosen by the constant strategy adopted by IBP. Zhang *et al.* [33] emperically show the adaptive strategy gives tighter bounds for normally trained networks. An intuitive explanation is that this strategy minimizes the area between the lower and upper bounding lines in the interval, but there is no guarantee for this intuition. On the other hand, for IBP trained networks, the loss is optimized at the point where bounding lines are chosen as constants. Therefore we should choose the same constant bounding lines or tighter bounding lines for LBP or CROWN when verifying IBP trained networks, which is exactly what we are doing in the tight strategy.

We conduct experiments to verify our theory. We first compare IBP, LBP and CROWN on a normally trained MNIST classifier (See its detailed structures in Appendix B.1). Result is shown in Figure 3. The average verification time for a single image of IBP, CROWN-IBP, LBP, CROWN-LBP, CROWN are 0.006s, 0.011s, 0.027s, 0.032s, 0.25s, respectively, tested on one NVIDIA GeForce GTX TITAN X GPU. We can see LBP is tighter than IBP while being faster than CROWN. And the adaptive strategy usually obtains tighter bounds than the tight strategy. See more comparisons of these methods in Appendix B.2.

Next, we compare them on an IBP trained network. The network we use is called DM-large (See its detailed structure in Appendix B.1), which is the same model in the work[32, 9]. Results are shown in Table 1. We don't test CROWN on this network because it exceeds GPU memory (12 GB) and takes about half an hour to verify a single image on one Intel Xeon E5-2650 v4 CPU. We can see CROWN-IBP, LBP and CROWN-LBP give worse verified errors than IBP when adopting adaptive strategy to choose bounding lines, but give better results when adopting the tight strategy as guaranteed by Theorem 2. However, we can see the improvement of LBP and CROWN-LBP over IBP and CROWN-IBP is small compared with the normally trained network. We investigate this phenomenon in the next section.

5. Parameterized Ramp Activation

This section starts by investigating the phenomenon discovered in Section 4: Why the improvement of LBP and CROWN-LBP over IBP and CROWN-IBP is so small on the IBP trained network compared with the normally trained network. Study of this phenomenon inspires us to design a new activation function to achieve lower verified errors.

Investigate the limited improvement of LBP. We argue that the limited improvement of LBP and CROWN-LBP is because most neurons are dead in IBP trained networks. Recall that we define three status of a ReLU neuron according to the range of its input in Figure 1: Dead, Alive, Unstable. We demonstrate neuron status in each layer of an IBP trained network in Figure 4. We can see most neurons are dead. However, we find most neurons (more than 95%) are unstable in a normally trained network. For unstable neurons, bounding lines in the tight strategy adopted by LBP and CROWN are tighter than the constant bounding lines chosen by IBP. This explains why LBP and CROWN are several orders tighter than IBP for a normally trained network. However, for dead neurons, the bounding lines chosen by LBP and CROWN are the same as those chosen by IBP, which explains the limited improvement of LBP and CROWN-LBP on IBP trained networks. We conduct experiments in Appendix B.3 to further verify this explanation.

It seems reasonable that most neurons are dead in IBP trained networks, since dead neurons can block perturbations from the input, which makes the network more robust. However, we argue that there are two major drawbacks caused by this phenomenon: First, gradients from both the

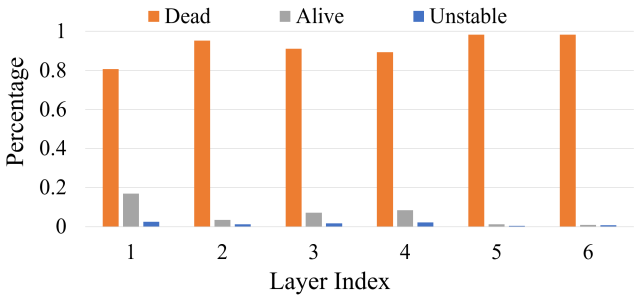


Figure 4. Neuron status of an IBP trained DM-large network at $\epsilon = 2.2/255$ (l_∞ norm) on CIFAR-10. Bounds are computed using IBP at $\epsilon = 2/255$. The horizontal axis is the layer index, and the vertical axis is the percentage of every neuron status in the layer. The percentage is averaged over 100 CIFAR-10 test images.

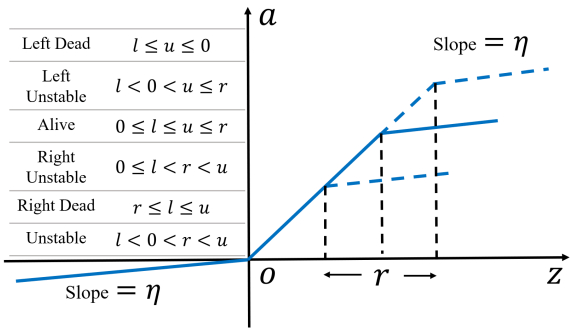


Figure 5. Parameterized Ramp (ParamRamp) function. The bending point r is tunable. We can define six status of a neuron according to the input range $[l, u]$ as shown in the left side. See Appendix A.7 for how to choose bounding lines for ParamRamp.

normal cross-entropy loss and IBP bound loss in (5) can not back-propagate through dead neurons. This may prevent the network from learning at some point of the training process. Second, it restricts the representation capability of the network, since most activations are 0 in intermediate layers.

Parameterized Ramp function. To mitigate these two problems, one simple idea is to use LeakyReLU instead of ReLU during training. We will consider this approach as the baseline and compare with it. We propose to use a **Parameterized Ramp (ParamRamp)** function to achieve better result. The Parameterized Ramp function can be seen as a LeakyReLU function with the right part being bent flat at some point r, as shown in Figure 5. The parameter r is tunable for every neuron. We include it to the parameters of the network and optimize over it during training. The intuition behind this activation function is that it provides another robust (function value changes very slowly with respect to the input) region on its right part. This right part has function values greater than 0 and tunable, in comparison to the left robust region with function values close to 0. Therefore during the IBP training process, a neuron has two options to become robust: to become either left dead or right dead as shown in Figure 5. This could increase the representation capability of the network while allow it to become robust. We compare effects of ReLU, LeakyReLU and ParamRamp functions in terms of training verifiably robust networks in the next section.

6. Experiments

In this section, we conduct experiments to train verifiably robust networks using our proposed activation function, ParamRamp, and compare it with ReLU and LeakyReLU. We use the loss defined in (5) and consider l_∞ robustness in all experiments. The experiments are conducted on 3 datasets: MNIST, CIFAR-10, and Tiny-ImageNet. For MNIST and CIFAR-10 datasets, we use the same DM-large network, and follow the same IBP training and CROWN-IBP training procedures in the works [9, 32].

For the Tiny-ImageNet dataset, we follow the training procedure in the work [30]. The networks we train on Tiny-ImageNet are a 7-layer CNN with Batch Normalization layers (CNN-7+BN) and a WideResNet. We refer readers to the original works or Appendix B.4 for detailed experimental set-ups and network structures. During the training of ParamRamp networks, it is important to initialize the tunable parameters r appropriately. We also find ParamRamp networks have overfitting problems in some cases. See how we initialize r and solve the overfitting problem in Appendix B.4. After training, we use IBP and CROWN-LBP with the tight strategy to compute verified errors. IBP verified errors allow us to compare results with previous works, and CROWN-LBP gives us the best verified errors as guaranteed in Theorem 2. CROWN is not considered because it exceeds GPU memory (12 GB) to verify a single image on the networks we use and is extremely slow running on CPU. We also use 200-step PGD attacks [15] with 10 random starts to emperically evaluate robustness of the networks.

Results on CIFAR-10 and MNIST datasets are presented in Table 2 and Table 3, respectively. We can see networks with ParamRamp activation achieve better verified errors, clean errors, and PGD attack errors than ReLU networks in almost all settings. And our proposed bound computation method, CROWN-LBP, can always provide lower verified errors than IBP. See more experiments for networks of different structures in Appendix B.5. For Tiny-ImageNet dataset, the CNN-7+BN and WideResNet networks with ParamRamp activation achieve 84.99% and 82.94% IBP verified errors at $\epsilon = 1/255$, respectively. To the best of our knowledge, 82.94% is the best verified error at $\epsilon = 1/255$ ever achieved on Tiny-ImageNet. See a comparison with ReLU networks from the work [30] in Appendix B.5.

ParamRamp activation brings additional parameters to the network. We are concerned about its computational overhead compared with ReLU networks. On MNIST, we find the average training time per epoch of a ParamRamp network is 1.09 times of that of a ReLU network

Table 2. Errors of IBP trained and CROWN-IBP trained networks with different activations on CIFAR-10. We report errors on clean images(Clean: Percentage of images wrongly classified), IBP verified errors(IBP), CROWN-LBP verified errors(C.-LBP), and PGD attack errors(PGD: Percentage of images successfully attacked). Experiments are conducted on 3 variants of ParamRamp: Ramp(0), ParamRamp with $\eta = 0$; Ramp(0.01), $\eta = 0.01$; Ramp(0.01→0), η starts from 0.01 and gradually decreases to 0 during training. 3 variants of ReLU are similarly designed, *e.g.*, ReLU(0.01) means LeakyReLU with leakage slope 0.01. Networks are trained at $\epsilon = 2.2/255, 8.8/255$ and evaluated at $\epsilon = 2/255, 8/255$ respectively. Results of ReLU(0) are directly copied from the original works [9, 32]. We compute C.-LBP verified errors based on our re-run networks for these experiments. Therefore, C.-LBP verified errors are not comparable to IBP verified errors on these networks. We also report results run on large ReLU networks in the work [30] at the right side.

Training Method	Activation	Errors (%) for $\epsilon = 2/255$				Errors (%) for $\epsilon = 8/255$				Errors (%) for $\epsilon = 8/255$			
		Clean	IBP	C.-LBP	PGD	Clean	IBP	C.-LBP	PGD	Model	Clean	IBP	PGD
IBP	ReLU(0)	39.22	55.19	54.38	50.40	58.43	70.81	69.98	68.73				
	ReLU(0.01)	**32.3**	52.02	47.26	44.22	55.16	69.05	68.45	66.05	CNN-7+BN	57.95	**69.56**	**67.10**
	ReLU(0.01→0)	34.6	53.77	51.62	46.71	55.62	68.32	68.22	65.29	Densenet	57.21	69.59	67.75
	Ramp(0)	36.47	53.09	52.28	46.52	56.32	68.89	68.82	63.89	WideResNet	58.07	70.04	67.23
	Ramp(0.01)	33.45	48.39	**47.19**	43.87	**54.16**	68.26	67.78	65.06	ResNeXt	**56.32**	70.41	67.55
	Ramp(0.01→0)	34.17	**47.84**	47.46	**42.74**	55.28	**67.26**	**67.09**	**60.39**				
CROWN -IBP	ReLU(0)	28.48	46.03	45.04	40.28	54.02	66.94	66.69	65.42				
	ReLU(0.01)	28.49	46.68	44.09	39.29	55.18	68.54	68.13	66.41	CNN-7+BN	**53.71**	**66.62**	64.31
	ReLU(0.01→0)	**28.07**	46.82	44.40	39.29	63.88	72.28	72.13	70.34	Densenet	56.03	67.57	65.09
	Ramp(0)	28.48	**45.67**	44.03	39.43	52.52	65.24	65.12	62.51	WideResNet	53.89	67.77	64.42
	Ramp(0.01)	28.63	46.17	44.28	39.61	52.15	66.04	65.75	63.85	ResNeXt	53.85	68.25	**64.16**
	Ramp(0.01→0)	28.18	45.74	**43.37**	**39.17**	**51.94**	**65.19**	**65.08**	**62.05**				

Table 3. Comparison of ParamRamp and ReLU on MNIST dataset. Notations are the same as those in Table 2. The networks are both trained and tested at $\epsilon = 0.4$. See more experiments tested with different activation functions and at different ϵ in Appendix B.5.

Training Method	Activation	Errors (%) for $\epsilon = 0.4$			
		Clean	IBP	C.-LBP	PGD
IBP	ReLU(0)	2.74	14.80	16.13	11.14
	Ramp(0.01→0)	2.16	10.90	10.88	6.59
CROWN -IBP	ReLU(0)	2.17	12.06	11.90	9.47
	Ramp(0.01→0)	2.36	10.68	10.61	6.61

in IBP training, and is 1.51 times in CROWN-IBP training. We observe an overhead of similar level on CIFAR-10 and Tiny-ImageNet datasets. See a full comparison in Appendix B.5. Comparing ParamRamp with ReLU on the same network may not be convincing enough to demonstrate the superiority of ParamRamp, as it has additional parameters. We compare it with larger size ReLU networks trained in the work [30]. We report their results on CNN-7+BN, Densenet [10], WideResNet [31] and ResNeXt [29] in the right part of Table 2. Despite being larger than the DM-large network with ParamRamp activation, these ReLU networks still can not obtain lower IBP verified errors than our model. We think this is because ParamRamp activation brings more diversity of neuron status, which increases the representation capability of the network. Recall that most neurons are dead in IBP trained ReLU networks as shown in Figure 4. We present neuron status of an IBP trained ParamRamp network in Figure 6. We can see although lots of neurons are still left dead, there is a considerable amount of neurons are right dead. Note that the activation value of right dead neurons are not 0 and tunable. This allows the network to become robust while preserving representation capability. See more neuron status comparisons of ReLU and ParamRamp networks in Appendix B.5.

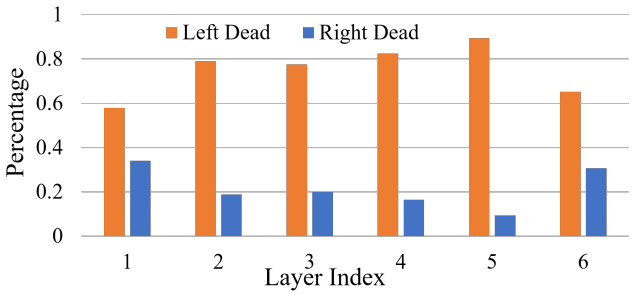


Figure 6. Neuron status of an IBP trained network on CIFAR-10 with ParamRamp activation. We only present left dead status and right dead status since most neurons are in these two status. The network is trained at $\epsilon = 2.2/255$ (l_∞ norm) and bounds are computed using IBP at $\epsilon = 2/255$.

7. Conclusion

We propose a new verification method, LBP, which has better scalability than CROWN while being tighter than IBP. We further prove CROWN and LBP are always tighter than IBP when choosing appropriate bounding lines, and can be used to verify IBP trained networks to obtain lower verified errors. We also propose a new activation function, ParamRamp, to mitigate the problem that most neurons become dead in ReLU networks during IBP training. Extensive experiments demonstrate networks with ParamRamp activation outperforms ReLU networks and achieve state-of-the-art l_∞ verified robustness on MNIST, CIFAR-10 and Tiny-ImageNet datasets.

Acknowledgement. This work is partially supported by General Research Fund (GRF) of Hong Kong (No. 14203518), Collaborative Research Grant from SenseTime Group (CUHK Agreement No. TS1712093, and No. TS1711490), and the Shanghai Committee of Science and Technology, China (Grant No. 20DZ1100800).

References

[1] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018. 1

[2] Akhilan Boopathy, Tsui-Wei Weng, Pin-Yu Chen, Sijia Liu, and Luca Daniel. Cnn-cert: An efficient framework for certifying robustness of convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3240–3247, 2019. 2

[3] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14, 2017. 1

[4] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017. 1

[5] Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess. Maximum resilience of artificial neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pages 251–268. Springer, 2017. 1

[6] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks, 2020. 1

[7] Sumanth Dathathri, Krishnamurthy Dvijotham, Alexey Kurakin, Aditi Raghunathan, Jonathan Uesato, Rudy Bunel, Shreya Shankar, Jacob Steinhardt, Ian Goodfellow, Percy Liang, and Pushmeet Kohli. Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming. *arXiv e-prints*, page arXiv:2010.11645, Oct. 2020. 1

[8] Krishnamurthy Dvijotham, Sven Gowal, Robert Stanforth, Relja Arandjelovic, Brendan O'Donoghue, Jonathan Uesato, and Pushmeet Kohli. Training verified learners with learned verifiers. *arXiv preprint arXiv:1805.10265*, 2018. 2

[9] S. Gowal, K. Dvijotham, R. Stanforth, R. Bunel, C. Qin, J. Uesato, R. Arandjelovic, T. A. Mann, and P. Kohli. Scalable verified training for provably robust image classification. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4841–4850, 2019. 2, 3, 6, 7, 8

[10] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017. 8

[11] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117. Springer, 2017. 1

[12] CY KO, Z Lyu, TW Weng, L Daniel, N Wong, and D Lin. Popqorn: Certifying robustness of recurrent neural networks. In *International Conference on Machine Learning (ICML)*, 2019. 1

[13] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016. 1

[14] Zhaoyang Lyu, Ching-Yun Ko, Zhifeng Kong, Ngai Wong, Dahua Lin, and Luca Daniel. Fastened crown: Tightened neural network robustness certificates. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:5037–5044, 04 2020. 3, 5

[15] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. 1, 7

[16] Matthew Mirman, Timon Gehr, and Martin Vechev. Differentiable abstract interpretation for provably robust neural networks. volume 80 of *Proceedings of Machine Learning Research*, pages 3578–3586, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. 2

[17] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. In *International Conference on Learning Representations*, 2018. 2

[18] Leslie Rice, Eric Wong, and J Zico Kolter. Overfitting in adversarially robust deep learning. *arXiv preprint arXiv:2002.11569*, 2020. 1

[19] Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. A convex relaxation barrier to tight robustness verification of neural networks. In *Advances in Neural Information Processing Systems*, pages 9835–9846, 2019. 2

[20] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. Fast and effective robustness certification. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 10802–10813. Curran Associates, Inc., 2018. 1, 2

[21] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–30, 2019. 1, 2

[22] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 1

[23] Vincent Tjeng, Kai Y. Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *International Conference on Learning Representations*, 2019. 1

[24] Shiqi Wang, Yizheng Chen, Ahmed Abdou, and Suman Jana. Mixtrain: Scalable training of formally robust neural networks. *CoRR*, abs/1811.02625, 2018. 1, 2

[25] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Efficient formal safety analysis of neural networks. In *Advances in Neural Information Processing Systems*, pages 6367–6377, 2018. 1, 2

[26] Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*, 2020. 1

[27] Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Duane Boning, Inderjit S Dhillon, and Luca

Daniel. Towards fast computation of certified robustness for relu networks. *ICML*, 2018. 1, 2, 3

[28] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *ICML*, volume 80, pages 5286–5295, 2018. 1, 2

[29] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5987–5995, 2017. 8

[30] Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. Automatic perturbation analysis for scalable certified robustness and beyond, 2020. 3, 5, 7, 8

[31] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12. BMVA Press, September 2016. 8

[32] Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. In *International Conference on Learning Representations*, 2020. 2, 3, 5, 6, 7, 8

[33] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In *NeurIPS*, pages 4944–4953. 2018. 1, 2, 3, 6

[34] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. volume 97 of *Proceedings of Machine Learning Research*, pages 7472–7482, Long Beach, California, USA, 09–15 Jun 2019. PMLR. 1