Fastened CROWN: Tightened Neural Network Robustness Certificates

Citation: Lyu, Zhaoyang, Ko, Ching-Yun, Kong, Zhifeng, Wong, Ngai, Lin, Dahua et al. 2020. "Fastened CROWN: Tightened Neural Network Robustness Certificates." Proceedings of the AAAI Conference on Artificial Intelligence, 34 (04).

As Published: 10.1609/AAAI.V34I04.5944

Publisher: Association for the Advancement of Artificial Intelligence (AAAI)

Persistent URL: https://hdl.handle.net/1721.1/143106

Version: Final published version: final published article, as it appeared in a journal, conference proceedings, or other formally published context

Fastened CROWN: Tightened Neural Network Robustness Certificates

Zhaoyang Lyu,[1]* Ching-Yun Ko,[2]* Zhifeng Kong,[3] Ngai Wong,[4] Dahua Lin,[1] Luca Daniel[2]
[1]The Chinese University of Hong Kong, Hong Kong, China
[2]Massachusetts Institute of Technology, Cambridge, MA 02139, USA
[3]University of California San Diego, La Jolla, CA 92093, USA
[4]The University of Hong Kong, Hong Kong, China
lyuzhaoyang@link.cuhk.edu.hk, {cyko, luca}@mit.edu, z4kong@eng.ucsd.edu,
nwong@eee.hku.hk, dhlin@ie.cuhk.edu.hk

Abstract

The rapid growth of deep learning applications in real life is accompanied by severe safety concerns. To mitigate this uneasy phenomenon, much research has been done providing reliable evaluations of the fragility level in different deep neural networks. Apart from devising adversarial attacks, quantifiers that certify safeguarded regions have also been designed in the past five years. The summarizing work in (Salman et al. 2019) unifies a family of existing verifiers under a convex relaxation framework. We draw inspiration from such work and further demonstrate the optimality of deterministic CROWN (Zhang et al. 2018) solutions in a given linear programming problem under mild constraints. Given this theoretical result, the computationally expensive linear programming based method is shown to be unnecessary. We then propose an optimization-based approach *FROWN* (**F**astened C**ROWN**): a general algorithm to tighten robustness certificates for neural networks. Extensive experiments on various networks trained individually verify the effectiveness of FROWN in safeguarding larger robust regions.

Introduction

The vulnerability of deep neural networks remains an unrevealed snare in the beginning years of the deep learning resurgence. In 2014, Szegedy et al. uncovered the discovery of hardly-perceptible adversarial perturbations that could fool image classifiers. This discovery agonized the fast development of accuracy-oriented deep learning and shifted community's attentions to the fragility of trained models. Especially with the increasing adoption of machine learning and artificial intelligence in safety-critical applications, the vulnerability of machine learning models to adversarial attacks has become a vital issue (Sharif et al. 2016; Kurakin, Goodfellow, and Bengio 2017; Carlini and Wagner 2017; Wong and Kolter 2018). Addressing this urging issue requires reliable ways to evaluate the robustness of a neural network, namely by studying the safety region around a data point where no adversarial example exists. This understanding of machine learning models' vulnerability will, on the other hand, help industries build more robust intelligent systems.

Disparate ways of reasoning and quantifying vulnerability (or robustness) of neural networks have been exploited to approach this dilemma, among which *attack-based* methods have long been in a dominating position. In these years, a sequel of adversarial attack algorithms have been proposed to mislead networks' predictions in tasks such as object detection (Goodfellow, Shlens, and Szegedy 2015; Moosavi-Dezfooli, Fawzi, and Frossard 2016), visual question answering (Mudrakarta et al. 2018; Zeng et al. 2019; Gao et al. 2019b; 2019a), text classification (Papernot et al. 2016), speech recognition (Cisse et al. 2017; Gong and Poellabauer 2017), and audio systems (Carlini and Wagner 2018), where the level of model vulnerability is quantified by the distortion between successful adversaries and the original data points. Notably, the magnitudes of distortions suggested by attack-based methods are essentially upper bounds of the minimum adversarial distortion.

In contrast to attack-based approaches, attack-agnostic *verification-based* methods evaluate the level of network vulnerability by either directly estimating (Szegedy et al. 2014; Weng et al. 2018b) or lower bounding (Hein and Andriushchenko 2017; Raghunathan, Steinhardt, and Liang 2018; Dvijotham et al. 2018; Zhang et al. 2018; Singh et al. 2018; Weng et al. 2019) the minimum distortion networks can bear for a specific input sample. As an iconic robustness estimation, CLEVER (Weng et al. 2018a) converts the robustness evaluation task into the estimation of the local Lipschitz constant, which essentially associates with the maximum norm of the local gradients *w.r.t.* the original example. Extensions of CLEVER (Weng et al. 2018c) focuses on twice differentiable classifiers and works on first-order Taylor polynomial with Lagrange remainder.

A number of verification-based methods have been proposed in literature to compute a lower bound of the safeguarded region around a given input, *i.e.* a region where the network is guaranteed to make consistent predictions despite any input perturbations. A pioneering work in providing certifiable robustness verification (Szegedy et al. 2014) computes the product of weight matrix operator norms in ReLU networks to give a lower-bounding metric of the

*Equal contribution. Source code and the appendix is available at https://github.com/ZhaoyangLyu/FROWN.

Table 1: List of Notations

Notation	Definition	Notation	Definition	Notation	Definition
$F : \mathbb{R}^n \to \mathbb{R}^t$	neural network classifier	$\mathbf{x}_0 \in \mathbb{R}^n$	original input	$\mathbf{x} \in \mathbb{R}^n$	perturbed input
n	input size	$\mathbf{a}^{(k)}$	the hidden state of the k-th layer	$\mathbf{z}^{(k)}$	the pre-activation of the k-th layer
n_k	number of neurons in layer k	$[K]$	set $\{1, 2, \cdots, K\}$	$\mathbb{B}_p(\mathbf{x}_0, \epsilon)$	$\{\mathbf{x} \mid \|\mathbf{x} - \mathbf{x}_0\|_p \leq \epsilon\}$
$F_j^L(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$	linear lower bound of $F_j(\mathbf{x})$	$\gamma_j^{(k)L}$	global lower bound of $\mathbf{z}_j^{(k)}$	$\mathbf{l} \preccurlyeq \mathbf{z} \preccurlyeq \mathbf{u}$	$\mathbf{l}_r \leq \mathbf{z}_r \leq \mathbf{u}_r$,
$F_j^U(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$	linear upper bound of $F_j(\mathbf{x})$	$\gamma_j^{(k)U}$	global upper bound of $\mathbf{z}_j^{(k)}$		$\forall\, r \in [s], \mathbf{l}, \mathbf{z}, \mathbf{u} \in \mathbb{R}^s$
$s^{[k-1]U}$	set $\{s^{(1)U}, \ldots, s^{(k-1)U}\}$	$t^{[k-1]U}$	set $\{t^{(1)U}, \ldots, t^{(k-1)U}\}$	$neg(\mathbf{x}) =$	$\mathbf{x}$, if $\mathbf{x} \leq 0$;
$s^{[k-1]L}$	set $\{s^{(1)L}, \ldots, s^{(k-1)L}\}$	$t^{[k-1]L}$	set $\{t^{(1)L}, \ldots, t^{(k-1)L}\}$		0, otherwise.
$\mathbf{a}^{[k]}$	set $\{\mathbf{a}^{(1)}, , \mathbf{a}^{(2)}, ..., \mathbf{a}^{(k)}\}$	$\mathbf{z}^{[k]}$	set $\{\mathbf{z}^{(1)}, , \mathbf{z}^{(2)}, ..., \mathbf{z}^{(k)}\}$	σ	ReLU/ Sigmoid/ Tanh activation

minimum distortion. However, this certificate method was shown to be generally too conservative to be useful (Hein and Andriushchenko 2017; Weng et al. 2018b). Later, tighter bounds have also been provided for continuously-differentiable shallow networks by utilizing local Lipschitz constants of the network (Hein and Andriushchenko 2017). Then, for the first time, the formal verification problem is reduced from a mixed integer linear programming (MILP) problem to a linear programming (LP) problem when dealing with l_∞-norm box constraints (Wong and Kolter 2018). Its concurrent works include Fast-Lin (Weng et al. 2018b), which analytically calculates bounds for perturbed samples in given regions and finds the largest certifiable region for ReLU networks through binary search. Fast-Lin is further generalized for multilayer perceptrons with general activation functions in CROWN (Zhang et al. 2018). Recently, Salman et al. conclude a general framework for a genre of convex relaxed optimization problems and demonstrate existing approaches to be special cases of their proposal. Notably, although Wong and Kolter propose to verify the robustness for ReLU network by the use of LP, a feasible dual solution is instead used in practice to avoid any actual use of LP solvers. Comparatively, Salman et al. experiment with more than one linear function to bound nonlinear activations (*e.g.* 2 lower-bounding functions for ReLU act.) and stick to LP solvers.

Certifiable robustness lower bounds are especially vital in safety-critical scenarios (*e.g.* autonomous driving car) since any miss-classification can be lethal. However, albeit being useful, new challenges with these certifiable quantifiers arise. There are, in most cases, non-negligible gaps between the certified lower and upper bounds of the minimum distortion. This inconsistency in the quantification questions diminishes the use of these state-of-the-art robustness evaluation approaches.

In this article, we stay in line with the previous sequel of works that focus on linear bounds and provide two major contributions:

1. We prove that if we limit the constraint relaxation to be exactly one linear bound in each direction (upper and lower) in the LP-based method, the results provided by CROWN are optimal. Therefore the costly LP solving process is unnecessary under this relaxation.
2. We propose a general optimization framework that we name *FROWN* (**F**astened C**ROWN**) for tightening the certifiable regions guaranteed by CROWN, which is also theoretically applicable to tighten convolutional neural network certificate CNN-Cert (Boopathy et al. 2019) and recurrent neural network certificate POPQORN (Ko et al. 2019).

Backgrounds

This section summarizes the most relevant backgrounds of our proposals. Specifically, LP formulation (Salman et al. 2019) is summarized, together with the seminal work of Fast-Lin (Weng et al. 2018b) and CROWN (Zhang et al. 2018) (generalized Fast-Lin). We first begin by giving the definition of an m-layer neural network:

Definitions. Given a trained m-layer perceptron F, we denote the hidden unit, weight matrix, bias, and pre-activation unit of the k-th layer ($k \in [m]$) as $\mathbf{a}^{(k)}$, $\mathbf{W}^{(k)}$, $\mathbf{b}^{(k)}$, and $\mathbf{z}^{(k)}$, respectively. Hence, $\mathbf{z}^{(k)} = \mathbf{W}^{(k)}\mathbf{a}^{(k-1)} + \mathbf{b}^{(k)}$, $\mathbf{a}^{(k)} = \sigma(\mathbf{z}^{(k)})$, where $\mathbf{a}^0 = \mathbf{x}_0 \in \mathbb{R}^n$ is the original input and $F(\mathbf{x}) = \mathbf{z}^{(m)}$ is the network output. Denoting the number of neurons as n_k for the k-th layer, implies that $\mathbf{a}^{(k)}, \mathbf{z}^{(k)}, \mathbf{b}^{(k)} \in \mathbb{R}^{n_k}$ and $\mathbf{W}^{(k)} \in \mathbb{R}^{n_k \times n_{k-1}}$, for $k \in [m]$. Furthermore, we use square brackets in the superscripts to group a set of variables (*e.g.* $\mathbf{a}^{[m]}$ denotes the set of variables $\{\mathbf{a}^{(1)}, , \mathbf{a}^{(2)}, ..., \mathbf{a}^{(m)}\}$ and $\mathbf{z}^{[m]}$ denotes the set of variables $\{\mathbf{z}^{(1)}, , \mathbf{z}^{(2)}, ..., \mathbf{z}^{(m)}\}$). Table 1 summarizes all the notations we use in this paper.

When quantifying the robustness of the m-layer neural network, one essentially wants to know 1) how far the network output will deviate when the input is perturbed with distortions of a certain magnitude and 2) the critical point in terms of distortion magnitudes, beyond which the deviation might alter the model prediction. If we let $\mathbf{x} \in \mathbb{R}^n$ denote the perturbed input of $\mathbf{x_0}$ (class i) within an ϵ-bounded l_p-ball (*i.e.*, $\mathbf{x} \in \mathbb{B}_p(\mathbf{x}_0, \epsilon)$, or$\|\mathbf{x} - \mathbf{x}_0\|_p \leq \epsilon$), the task of robustness analysis for this network intrinsically involves the comparison between the i-th network output $F_i(\mathbf{x})$ and other outputs $F_{j \neq i}(\mathbf{x})$. In practice, one can translate the original problem to the problem of deriving a lower bound of $F_i(\mathbf{x})$ and upper bounds of $F_{j \neq i}(\mathbf{x})$ for perturbed inputs within the l_p-norm ball. With such quantifier, network F is guaranteed to make consistent predictions within the l_p-norm ball if the lower bound of the original class output is always larger than the upper bounds of all other classes' outputs.

We summarize below the LP problem (Salman et al. 2019) to solve the lower bound of $F_i(\mathbf{x})$. The LP problem for the

upper bound of $F_{j\neq i}(\mathbf{x})$ can be similarly derived.

The LP problem. The optimization problem for solving the lower bound of $F_i(\mathbf{x}) = \mathbf{z}_i^{(m)} = \mathbf{W}_{i,:}^{(m)}\mathbf{a}^{(m-1)} + \mathbf{b}_i^{(m)}$ reads:

$$\min_{\mathbf{a}^{(0)}\in\mathbb{B}_p(x_0,\epsilon),\mathbf{a}^{[m-1]},\mathbf{z}^{[m-1]}} \mathbf{W}_{i,:}^{(m)}\mathbf{a}^{(m-1)} + \mathbf{b}_i^{(m)} \quad (1)$$
$$\text{s.t.} \begin{cases} \mathbf{z}^{(k)} &= \mathbf{W}^{(k)}\mathbf{a}^{(k-1)} + \mathbf{b}^{(k)}, \forall\, k \in [m-1], \\ \mathbf{a}^{(k)} &= \sigma(\mathbf{z}^{(k)}), \forall\, k \in [m-1]. \end{cases}$$

The optimization problem for upper bounds can be readily obtained by replacing the "min" operation by "max". Then, the nonlinear constraint in (1) is lifted with linear relaxations. Specifically, suppose the lower and upper bounds of the pre-activation units $\mathbf{z}^{[m-1]}$ are known, namely, for k from 1 to $m-1$, as a result $\mathbf{l}^{(k)}$ and $\mathbf{u}^{(k)}$ satisfy

$$\mathbf{l}^{(k)} \preccurlyeq \mathbf{z}^{(k)} \preccurlyeq \mathbf{u}^{(k)}, \quad (2)$$

and therefore every element $\sigma(\mathbf{z}_i^{(k)})$ of the nonlinear activation $\sigma(\mathbf{z}^{(k)})$ in constraint (1) can be bounded by linear functions:

$$h_i^{(k)L}(\mathbf{z}_i^{(k)}) \leq \sigma(\mathbf{z}_i^{(k)}) \leq h_i^{(k)U}(\mathbf{z}_i^{(k)}), \forall \mathbf{z}_i^{(k)} \in [\mathbf{l}_i^{(k)}, \mathbf{u}_i^{(k)}], \quad (3)$$

for $i \in [n_k]$. The existence of linear bounding functions in (3) is guaranteed since $\mathbf{z}_i^{(k)}$ is bounded and compactness is a continuous invariant within any interval. For example, the following are bounding functions: $h_i^{(k)L}(\mathbf{z}_i^{(k)}) = \min_{z_i^{(k)}\in[\mathbf{l}_i^{(k)},\mathbf{u}_i^{(k)}]}(\sigma(\mathbf{z}_i^{(k)}))$, $h_i^{(k)U}(\mathbf{z}_i^{(k)}) = \max_{\mathbf{z}_i^{(k)}\in[\mathbf{l}_i^{(k)},\mathbf{u}_i^{(k)}]}(\sigma(\mathbf{z}_i^{(k)}))$. $h_i^{(k)L}$ and $h_i^{(k)U}$ can also be taken as the pointwise supremum and infimum of several linear functions, respectively, which is equivalent to using multiple linear constraints. In practice, Salman et al. use linear functions characterized by slopes and intercepts:

$$h_i^{(k)L}(\mathbf{z}_i^{(k)}) = s_i^{(k)L}\mathbf{z}_i^{(k)} + t_i^{(k)L},$$
$$h_i^{(k)U}(\mathbf{z}_i^{(k)}) = s_i^{(k)U}\mathbf{z}_i^{(k)} + t_i^{(k)U}. \quad (4)$$

The optimization problem can therefore be relaxed to an LP-alike problem[1]:

$$\min_{\mathbf{a}^{(0)}\in\mathbb{B}_p(x_0,\epsilon),\mathbf{a}^{[m-1]},\mathbf{z}^{[m-1]}} \mathbf{W}_{i,:}^{(m)}\mathbf{a}^{(m-1)} + \mathbf{b}_i^{(m)} \quad (5)$$
$$\text{s.t.} \begin{cases} \mathbf{z}^{(k)} = \mathbf{W}^{(k)}\mathbf{a}^{(k-1)} + \mathbf{b}^{(k)}, \forall\, k \in [m-1], \\ h^{(k)L}(\mathbf{z}^{(k)}) \preccurlyeq \mathbf{a}^{(k)} \preccurlyeq h^{(k)U}(\mathbf{z}^{(k)}), \forall\, k \in [m-1], \\ \mathbf{l}^{(k)} \preccurlyeq \mathbf{z}^{k} \preccurlyeq \mathbf{u}^{(k)}, \forall\, k \in [m-1]. \end{cases}$$

Recalling that with the optimization formed as in Problem (5), one is essentially optimizing for the lower (or upper) output bounds of the network (the pre-activation of the m-th layer), whereas these build upon the assumption that the pre-activation bounds are known as Equation (2). To satisfy this assumption, one actually only needs to substitute the layer index m in Problem (5) with the corresponding intermediate layer's index. In practice, one can recursively solve LP problems from the second layer to the m-th layer to obtain the pre-activation bounds for all layers. In this process, the pre-activation bounds computed in a layer also constitute the optimization constraint for the next to-be-solved optimization problem for the next layer's pre-activation. See details of this LP-based method in Appendix Section A.3.

CROWN Solutions. Here we briefly walk through the derivation of Fast-Lin (Weng et al. 2018b) and CROWN (Zhang et al. 2018), whose procedures are essentially the same except for activation-specific bounding rules adopted. The first steps include bounding $\mathbf{z}_i^{(k)}$, $k \in [m]$ [2].

$$\mathbf{z}_i^{(k)} = \sum_{j=1}^{n_{k-1}} \mathbf{W}_{i,j}^{(k)}\sigma(\mathbf{z}_j^{(k-1)}) + \mathbf{b}_i^{(k)}, \quad (6)$$
$$\geq \sum_{j=1}^{n_{k-2}} \tilde{\mathbf{W}}_{i,j}^{(k-1)}\sigma(\mathbf{z}_j^{(k-2)}) + \tilde{\mathbf{b}}_i^{(k-1)}, \quad (7)$$

where $neg(\mathbf{x}) = \mathbf{x}$, if $\mathbf{x} \leq 0$; $neg(\mathbf{x}) = 0$, otherwise. And $\tilde{\mathbf{W}}_{i,j}^{(k-1)} = [relu(\mathbf{W}_{i,:}^{(k)}) \odot (s^{(k-1)L})^\top + neg(\mathbf{W}_{i,:}^{(k)}) \odot (s^{(k-1)U})^\top]\mathbf{W}_{:,j}^{(k-1)}$, $\tilde{\mathbf{b}}_i^{(k-1)} = [relu(\mathbf{W}_{i,:}^{(k)}) \odot (s^{(k-1)L})^\top + neg(\mathbf{W}_{i,:}^{(k)}) \odot (s^{(k-1)U})^\top]\mathbf{b}^{(k-1)} + relu(\mathbf{W}_{i,:}^{(k)})t^{(k-1)L} + neg(\mathbf{W}_{i,:}^{(k)})t^{(k-1)U} + \mathbf{b}_i^{(k)}$, where $\odot$ denotes element-wise products. As Equations (6) and (7) are in similar forms, the above procedures can be repeated until all the nonlinearities in $k-1$ layers are unwrapped by linear bounding functions and $\mathbf{z}_i^{(k)}$ is upper bounded by $\sum_{j=1}^{n}\tilde{\mathbf{W}}_{i,j}^{(1)}\mathbf{x}_j + \tilde{\mathbf{b}}_i^{(1)}$, where $\tilde{\mathbf{W}}_{i,j}^{(1)}$ and $\tilde{\mathbf{b}}_i^{(1)}$ are similarly defined as shown above. Taking the dual form of the bound then yields the closed-form bound γ_j^L that satisfies

$$\mathbf{z}_i^{(k)} \geq \gamma_i^{(k)L} := \tilde{\mathbf{W}}_{i,:}^{(1)}\mathbf{x}_0 - \epsilon\|\tilde{\mathbf{W}}_{i,:}^{(1)}\|_q + \tilde{\mathbf{b}}_i^{(1)}, \quad (8)$$

$\forall\, \mathbf{x} \in \mathbb{B}_p(\mathbf{x}_0, \epsilon)$, where $1/p + 1/q = 1$. Although the steps above are used to derive the closed-form lower bound, the closed-form upper bound $\gamma_i^{(k)U}$ can be similarly derived. To quantify the robustness for an m-layer neural network, one needs to recursively adopt formulas in Equation (8) to calculate the bounds of pre-activation[3] $\mathbf{z}^{(k)}$, for $k = 2, \ldots, m$. These bounds, as will be explained in more details later, confine the feasible set for choosing bounding linear functions in Equation (4). Notably, lower bound $\gamma_i^{(k)L}$ and upper bound $\gamma_i^{(k)U}$ are implicitly reliant to slopes of bounding lines in previous layers $s^{[k-1]U} = \{s^{(1)U}, \ldots, s^{(k-1)U}\}$, $s^{[k-1]L} = \{s^{(1)L}, \ldots, s^{(k-1)L}\}$ and

[1] The optimization problem turns to a strict LP problem only when we have $p = \infty$ or 1 that makes the feasible set a polyhedron. However we coarsely denote all the cases in general as LP problems since all the constraints are now linear in the variables.

[2] Similar to the discussion in the LP-based method above, the bounds computed are exactly network output bounds when $k = m$; whereas $k \neq m$ gives the pre-activation bounds to fulfill the assumption in Inequality (2).

[3] $\mathbf{z}^{(1)}$ is deterministically computed by the input and $\mathbf{z}^{(m)} = F(\mathbf{x})$ is the output bound.

their intercepts $t^{[k-1]U} = \{t^{(1)U}, \ldots, t^{(k-1)U}\}$, $t^{[k-1]L} = \{t^{(1)L}, \ldots, t^{(k-1)L}\}$. A major difference that distinguishes CROWN from our contributions in the following sections is its deterministic rules of choosing upper/lower-bounding lines. The readers are referred to the literature (Zhang et al. 2018) or Sections A.2 and A.4 in the appendix for more details of CROWN.

Relation Between the LP Problem and CROWN Solutions

Now we discuss the relationship between the LP problem formulation and CROWN. A key conclusion is that: CROWN is not only a dual feasible solution of the presented LP problem as discussed by Salman et al., it gives the optimal solution under mild constraints.

Before introducing the optimality of CROWN solutions under the LP framework, we define an important condition in the computation process of CROWN as below:

Condition 1 ***Self-consistency.*** *Suppose* $\{\tilde{s}^{[v-1]U}, \tilde{s}^{[v-1]L}, \tilde{t}^{[v-1]U}, \tilde{t}^{[v-1]L}\}$ *are used to calculate* $\gamma_i^{(v)L}$ *and* $\gamma_i^{(v)U}$, $\{\hat{s}^{[k-1]U}, \hat{s}^{[k-1]L}, \hat{t}^{[k-1]U}, \hat{t}^{[k-1]L}\}$ *are used to calculate* $\gamma_j^{(k)L}$ *and* $\gamma_j^{(k)U}$, *then the following condition holds,*

$$\tilde{s}^{[v-1]U} = \hat{s}^{[v-1]U}, \tilde{s}^{[v-1]L} = \hat{s}^{[v-1]L},$$
$$\tilde{t}^{[v-1]U} = \hat{t}^{[v-1]U}, \tilde{t}^{[v-1]L} = \hat{t}^{[v-1]L},$$

for $\forall\, i \in [n_v], \forall j \in [n_k], 2 \leq v \leq k \leq m$ *and two sets equal to each other is defined as their corresponding elements equal to each other.*

A similar condition can also be defined in the LP-based method and is supplemented in Section A.3 in the appendix. The self-consistency condition guarantees the same set of bounding lines is used when computing bounds for different neurons in the process of CROWN or the LP-based method. We note that both the original CROWN and the LP-based method satisfy the self-consistency condition.

Theorem 1 *The lower bound obtained by Equation* (8) *is the optimal solution to Problem* (5) *when the following three conditions are met:*

- *Each of the* $h^{(k)L}(\mathbf{z}^k)$ *and* $h^{(k)U}(\mathbf{z}^k)$ *in Problem* (5) *is chosen to be one linear function*[4] *as in Equation* (4).
- *The LP problem shares the same bounding lines with CROWN.*
- *The self-consistency conditions for both CROWN and the LP-based method hold.*

We refer readers to Section A.5 in the appendix for the proof. We emphasize the cruciality of the self-consistency conditions in Theorem 1: We do observe CROWN and the LP-based method can give different bounds when Condition 1 is not met, even though the two use the same bounding lines. In essence, Theorem 1 allows one to compute bounds analytically and efficiently following steps in CROWN instead of solving the expensive LP problems under certain conditions.

[4]Theoretically one can use multiple linear functions to bound the nonlinearity in Problem (5) to obtain tighter bounds.

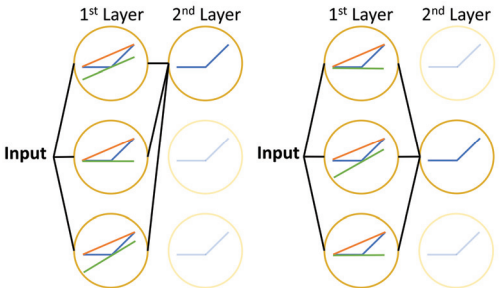


Figure 1: The process of CROWN using different bounding lines to compute the closed-form bounds for different neurons. The blue curves are the ReLU activation. The orange and green lines are the upper and lower bounding lines, respectively. When computing closed-form bounds of the pre-activation of neurons in the second layer, different neurons can choose different bounding lines in the previous layers to yield the tightest closed-form bounds for themselves.

Fastened CROWN

Recognizing the dependency of lower bounds and upper bounds to slopes and intercepts in the original CROWN method, a consistent use of these parameters is enforced through the self-consistency condition. In fact, we argue that this constraint can be lifted in general (we visualize this relaxation in Figure 1). The aim of this section stems from this relaxation and focuses on optimizing the pre-activation/output bounds over these tunable bounding parameters to achieve tighter bounds. In that merit, we propose an optimization framework called *FROWN* (**F**astened C**ROWN**) for tightening robustness certificates in CROWN. Moreover, FROWN is versatile and can be widely applied to tighten previously-proposed CNN-Cert (Boopathy et al. 2019) for convolutional neural networks and POPQORN (Ko et al. 2019) for recurrent neural networks. We formalize the objective as the following two optimization problems:

$$\max_{s^{[k-1]L}, s^{[k-1]U}, t^{[k-1]L}, t^{[k-1]U}} \gamma_i^{(k)L} \tag{9}$$
$$\text{s.t. } s_i^{(v)L}\mathbf{z}_i^{(v)} + t_i^{(v)L} \leq \sigma(\mathbf{z}_i^{(v)}) \leq s_i^{(v)U}\mathbf{z}_i^{(v)} + t_i^{(v)U},$$
$$\forall\, \mathbf{z}_i^{(v)} \in [\mathbf{l}_i^{(v)}, \mathbf{u}_i^{(v)}], i \in [n_v], v \in [k-1],$$

and

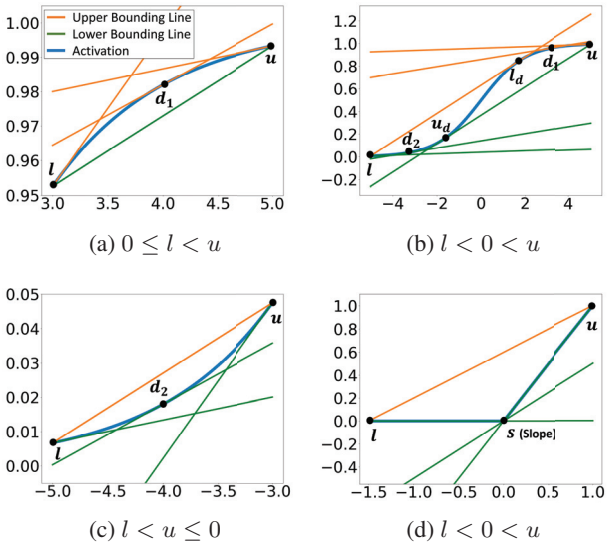
$$\min_{s^{[k-1]L}, s^{[k-1]U}, t^{[k-1]L}, t^{[k-1]U}} \gamma_i^{(k)U} \tag{10}$$
$$\text{s.t. } s_i^{(v)L}\mathbf{z}_i^{(v)} + t_i^{(v)L} \leq \sigma(\mathbf{z}_i^{(v)}) \leq s_i^{(v)U}\mathbf{z}_i^{(v)} + t_i^{(v)U},$$
$$\forall\, \mathbf{z}_i^{(v)} \in [\mathbf{l}_i^{(v)}, \mathbf{u}_i^{(v)}], i \in [n_v], v \in [k-1].$$

However, we stress that Problems (9) and (10) are generally non-convex optimization problems when there are more than two layers in the target network. We enclose the proof as Section A.7 in the appendix. Therefore, optimizing for a non-convex objective function over parameters in large search spaces with infinite number of constraints is impractical. To this end, our ideas are to limit the search space to

Table 2: Search space of bounding lines for ReLU, Sigmoid, and Tanh functions. "Variable" is the optimization variable that characterizes the bounding line. "Range" is the feasible region of the variable. "-" indicates the case when the tightest bounding line is unique and chosen. The slope and intercept of ReLU upper-bounding line are always set to be s_0 and $t(s_0, l)$, respectively.

Nonlinearity	ReLU (Lower bnd.)			Sigmoid & Tanh (Upper bnd.)				Sigmoid & Tanh (Lower bnd.)			
Pre-activation Bounds	$l < u \leq 0$	$l < 0 < u$	$0 \leq l < u$	$l < u \leq 0$	$l < 0 < u$ case 1		$0 \leq l < u$	$l < u \leq 0$	$l < 0 < u$ case 3		$0 \leq l < u$
						case 2				case 4	
Variable	-	s	-	-	d_1	-	d_1	d_2	d_2	-	-
Range	-	$[0, 1]$	-	-	$[l_d, u]$	-	$[l, u]$	$[l, u]$	$[l, u_d]$	-	-
Slope	s_0	s	s_0	s_0	$\sigma'(d_1)$	s_0	$\sigma'(d_1)$	$\sigma'(d_2)$	$\sigma'(d_2)$	s_0	s_0
Intercept	$t(s_0, l)$	0	$t(s_0, l)$	$t(s_0, l)$	$t(\sigma'(d_1), d_1)$	$t(s_0, l)$	$t(\sigma'(d_1), d_1)$	$t(\sigma'(d_2), d_2)$	$t(\sigma'(d_2), d_2)$	$t(s_0, l)$	$t(s_0, l)$

Notes: Case 1 refers to $\sigma'(u)l + t(\sigma'(u), u) \geq \sigma(l)$; and case 2, otherwise. Case 3 refers to $\sigma'(l)u + t(\sigma'(l), l) \leq \sigma(u)$; and case 4, otherwise. $s_0 = [\sigma(u) - \sigma(l)]/(u - l)$, $t(s, y) = \sigma(y) - sy$. l_d and u_d are defined as the abscissas of the points at which the tangent passes the left endpoint $(l, \sigma(l))$ and the right endpoint $(u, \sigma(u))$, respectively. d_1 and d_2 are the abscissas of the points of tangency. See Figure 2 for the visualization of l_d, u_d, d_1 and d_2.



(a) $0 \leq l < u$ (b) $l < 0 < u$

(c) $l < u \leq 0$ (d) $l < 0 < u$

Figure 2: Illustration of the search space of bounding lines for Sigmoid and ReLU activation. See definitions of d_1, d_2, u_d, l_d in Table 2.

smaller ones. We present our solutions by first introducing the ideas of "tighter" bounding lines.

Definition 1 *Suppose* $\tilde{h}_i^{(k)L}(\mathbf{z}_i^{(k)}) = \tilde{s}_i^{(k)L}\mathbf{z}_i^{(k)} + \tilde{t}_i^{(k)L}$ *and* $\hat{h}_i^{(k)L}(\mathbf{z}_i^{(k)}) = \hat{s}_i^{(k)L}\mathbf{z}_i^{(k)} + \hat{t}_i^{(k)L}$ *are two lower-bounding lines that satisfy*

$$\begin{cases} \tilde{h}_i^{(k)L}(\mathbf{z}_i^{(k)}) > \hat{h}_i^{(k)L}(\mathbf{z}_i^{(k)}),\ \forall\, \mathbf{z}_i^{(k)} \in (\mathbf{l}_i^{(k)}, \mathbf{u}_i^{(k)}) \\ \sigma(\mathbf{z}_i^{(k)}) \geq \tilde{h}_i^{(k)L}(\mathbf{z}_i^{(k)}),\ \forall\, \mathbf{z}_i^{(k)} \in [\mathbf{l}_i^{(k)}, \mathbf{u}_i^{(k)}] \end{cases}$$

then we say $\tilde{h}_i^{(k)L}(\mathbf{z}_i^{(k)}) = \tilde{s}_i^{(k)L}\mathbf{z}_i^{(k)} + \tilde{t}_i^{(k)L}$ *is a tighter lower-bounding line than* $\hat{h}_i^{(k)L}(\mathbf{z}_i^{(k)}) = \hat{s}_i^{(k)L}\mathbf{z}_i^{(k)} + \hat{t}_i^{(k)L}$ *for the nonlinear activation* σ *in the interval* $[\mathbf{l}_i^{(k)}, \mathbf{u}_i^{(k)}]$.

Similarly, we define a tighter upper-bounding line. Accordingly, the tightest bounding line refers to the $\hat{h}_i^{(k)L}$ when there is, by definition, no tighter bounding line than itself. Note that the tightest bounding line may not be unique. For example, any line passing through the origin with a slope between 0 and 1 is a tightest lower-bounding line for the ReLU activation in an interval across the origin ($\mathbf{l}_i^{(v)} < 0 < \mathbf{u}_i^{(v)}$).

With the notion of tightness, a straightforward idea is to adopt one of the tightest bounding lines in every layer for generally tighter closed-form pre-activation/output bounds. However, the proposition:

tighter bounding lines $\Rightarrow$ tighter closed-form bounds

is not always true. We observe tighter bounding lines can sometimes lead to looser bounds. However, if we roll back to Condition 1, we can prove that it constitutes a sufficient condition for the proposition.

Theorem 2 *If the robustness of a neural network is evaluated by CROWN on two trials with two different sets of bounding lines characterized by* $\{\tilde{s}^{[k-1]U}, \tilde{s}^{[k-1]L}, \tilde{t}^{[k-1]U}, \tilde{t}^{[k-1]L}\}$ *and* $\{\hat{s}^{[k-1]U}, \hat{s}^{[k-1]L}, \hat{t}^{[k-1]U}, \hat{t}^{[k-1]L}\}$*, and in both of which the self-consistency condition is met, then the closed-form bounds obtained via CROWN satisfy*

$$\gamma_i^{(k)L}(\tilde{s}^{[k-1]U}, \tilde{s}^{[k-1]L}, \tilde{t}^{[k-1]U}, \tilde{t}^{[k-1]L}) \geq \gamma_i^{(k)L}(\hat{s}^{[k-1]U}, \hat{s}^{[k-1]L}, \hat{t}^{[k-1]U}, \hat{t}^{[k-1]L}),$$
$$\gamma_i^{(k)U}(\tilde{s}^{[k-1]U}, \tilde{s}^{[k-1]L}, \tilde{t}^{[k-1]U}, \tilde{t}^{[k-1]L}) \leq \gamma_i^{(k)U}(\hat{s}^{[k-1]U}, \hat{s}^{[k-1]L}, \hat{t}^{[k-1]U}, \hat{t}^{[k-1]L}),$$

for $\forall\ i \in [n_k]$*, when bounding lines determined by* $\{\tilde{s}^{[k-1]U}, \tilde{s}^{[k-1]L}, \tilde{t}^{[k-1]U}, \tilde{t}^{[k-1]L}\}$ *are the same as or tighter than those determined by* $\{\hat{s}^{[k-1]U}, \hat{s}^{[k-1]L}, \hat{t}^{[k-1]U}, \hat{t}^{[k-1]L}\}$.

Proof: The self-consistency guarantees the optimality of bounds given by CROWN to the corresponding LP problem (5) according to Theorem 1. As the use of tighter bounding lines in Problem (5) narrows the feasible set, the optimal value of Problem (5) (lower pre-activation/output bound) will stay or grow larger, which means the lower bound given by CROWN will stay or grow larger. Similar arguments extend to tightened upper bounds. □

Till now, we have confirmed the connection between the tightest bounding lines and the tightest CROWN pre-activation/output bound under Condition 1. In addition, we manage to prove Theorem 2 under a weaker condition (see its proof in Section A.6 in the appendix).

Condition 1 is too strong a condition for our proposed optimization framework to be practical. Actually we propose to improve CROWN by breaking Condition 1. Our problem can be eased by only considering the dependency of

closed-form pre-activation/output bounds on the intercepts (with the slopes fixed). We provide the following theorem:

Theorem 3 *If the robustness of a neural network is evaluated by CROWN on two trials with bounding lines characterized by* $\{s^{[k-1]U}, s^{[k-1]L}, \tilde{t}^{[k-1]U}, \tilde{t}^{[k-1]L}\}$ *and* $\{s^{[k-1]U}, s^{[k-1]L}, \hat{t}^{[k-1]U}, \hat{t}^{[k-1]L}\}$*, then the closed-form bounds obtained via CROWN satisfy*

$$\gamma_i^{(k)L}(s^{[k-1]U}, s^{[k-1]L}, \tilde{t}^{[k-1]U}, \tilde{t}^{[k-1]L}) \geq \gamma_i^{(k)L}(s^{[k-1]U}, s^{[k-1]L}, \hat{t}^{[k-1]U}, \hat{t}^{[k-1]L}),$$
$$\gamma_i^{(k)U}(s^{[k-1]U}, s^{[k-1]L}, \tilde{t}^{[k-1]U}, \tilde{t}^{[k-1]L}) \leq \gamma_i^{(k)U}(s^{[k-1]U}, s^{[k-1]L}, \hat{t}^{[k-1]U}, \hat{t}^{[k-1]L}),$$

for $\forall\ i \in [n_k]$*, when* $\tilde{t}^{(v-1)L} \succcurlyeq \hat{t}^{(v-1)L}$ *and* $\tilde{t}^{(v-1)U} \preccurlyeq \hat{t}^{(v-1)U}$, $\forall v \in [k-1]$.

This theoretical guarantee limits the freedom in choosing the intercepts: we should always choose upper-bounding lines with smaller intercepts and lower-bounding lines with larger intercepts if different bounding lines are allowed for different network neurons. Note that this conclusion holds under no assumptions on the choice of bounding lines and hence can be used to instruct how to choose bounding lines in FROWN. We demonstrate Theorem 3 can be used to reduce the search space of the upper (or lower) bounding line to one that can be characterized by a single variable continuously in Appendix Section A.8. This enables the usage of gradient-based method to search over candidate bounding lines to obtain tighter bounds. We further limit the search space to the tightest bounding lines (which is a subset of the search space narrowed only by Theorem 3) as demonstrated in Table 2 and exemplified in Figure 2 in order to simplify implementation. We emphasize that this limit is not necessary. FROWN can be readily generalized to the case in which the search space is reduced only by Theorem 3, and the obtained bounds should be even tighter as the search space is larger. Since the tightest bounding lines defined in Table 2 automatically satisfy the optimization constraints in Problems (9) and (10), the constrained optimization problems are then converted to unconstrained ones. Furthermore, notice that the objective functions in the two problems are differentiable to bounding line parameters. This allows us to solve the problems by projected gradient descent (Nesterov 2014) (see details in Appendix Section A.9).

By and large, given an m-layer network F, input sample $\mathbf{x}_0 \in \mathbb{R}^n$, l_p ball parameters $p \geq 1$, and $\epsilon \geq 0$, for $\forall\ j \in [n_m]$, $1/q = 1 - 1/p$, we can compute two fixed values γ_j^L and γ_j^U such that $\forall \mathbf{x} \in \mathbb{B}_p(\mathbf{x}_0, \epsilon)$, the inequality $\gamma_j^L \leq F_j(\mathbf{x}) \leq \gamma_j^U$ holds. Suppose the label of the input sequence is i, the largest possible lower bound ϵ_i of untargeted and targeted (target class j) attacks is found by solving:

$$\text{Untargeted:} \quad \epsilon_i = \max_{\epsilon} \epsilon,\ \text{s.t.}\ \gamma_i^L(\epsilon) \geq \gamma_j^U(\epsilon),\ \forall j \neq i.$$
$$\text{Targeted:} \quad \hat{\epsilon}(i,j) = \max_{\epsilon} \epsilon,\ \text{s.t.}\ \gamma_i^L(\epsilon) \geq \gamma_j^U(\epsilon).$$

We conduct binary search procedures to compute the largest possible ϵ_i (or $\hat{\epsilon}$).

Experimental Results

Overview. In this section, we aim at comparing the LP-based method[5] and FROWN as two approaches to improve CROWN. We allow the LP-based method to use more than one bounding lines (which also increases computation cost) in order to make improvement on CROWN. Specifically, two lower-bounding lines are considered for ReLU networks while up to three upper/lower-bounding lines are adopted for Sigmoid (or Tanh) networks in the LP-based method (more details supplemented as Section A.4 in the appendix). On the other hand, FROWN improves CROWN solutions by optimizing over the bounding lines to give tighter bounds. These two approaches are evaluated and compared herein by both the safeguarded regions they certify and their time complexity. We run the LP-based method on a single Intel Xeon E5-2640 v3 (2.60GHz) CPU. We implement our proposed method FROWN using PyTorch to enable the use of an NVIDIA GeForce GTX TITAN X GPU. However, we time FROWN on a single Intel Xeon E5-2640 v3 (2.60GHz) CPU when comparing with the LP-based method for fair comparisons. We leave the detailed experimental set-ups and complete experimental results to Appendix Section A.9.

Experiment I. In the first experiment, we compare the improvements of FROWN and the LP-based method over CROWN on sensorless drive diagnosis networks [6] and MNIST classifiers. We present their results in Table 3. As shown in the table, we consider ReLU and Sigmoid (results of Tanh networks are included in Appendix Section A.9) networks that are trained independently on two datasets. The size of the networks ranges from 3 layers to 20 layers and 20 neurons to 100 neurons. We remark that even on networks with only 100 neurons, the LP-based method scales badly and is unable to provide results in 100 minutes for only one image. The improved bounds in Table 3 verify the effectiveness of both FROWN and LP-based approach in tightening CROWN results. Specifically, an up to 93% improvement in the magnitude of bounds is witnessed on sensorless drive diagnosis networks. And in general, the deeper the target network is, the greater improvement can be achieved. When comparing FROWN to the LP-based method, it is demonstrated that FROWN computes bounds up to two orders of magnitudes faster than the LP-based method and is especially advantageous when certifying l_1-norm regions. On the other hand, while the LP-based method gives larger certified bounds for ReLU networks in most cases, FROWN certifies larger bounds for Sigmoid and Tanh networks.

Experiment II. In our second experiment, we compute the robustness certificate on CIFAR10 networks that have 2048 neurons in each layer. With the width of neural networks, the LP-based method is **unusable** due to its high computational-complexity. Therefore, we only show the improvements FROWN has brought to the original CROWN solutions. In

[5] The highly-efficient Gurobi LP solver is adopted here.

[6] https://archive.ics.uci.edu/ml/datasets/Dataset+for+Sensorless+Drive+Diagnosis

Table 3: (Experiment I) Averaged certified l_∞ bounds and l_p bounds ($p = 1, 2, \infty$) of Sensorless Drive Diagnosis classifiers and MNIST classifiers, respectively. "N/A" indicates no results can be obtained in the given runtime. The up arrow "↑" means " more than". "$m \times [N]$ σ" means an m-layer network with N neurons and σ activation.

Network	p	Certified Bounds			Improvement		Avg. Time per Image (s)		Speedups of
		CROWN	FROWN	LP	FROWN	LP	FROWN	LP	FROWN over LP
Sensorless Drive Diagnosis classifiers									
$4 \times [20]$ ReLU	∞	0.2019	0.2247	0.2269	11.27%	**12.38%**	0.35	0.96	2.8 X
$8 \times [20]$ ReLU	∞	0.2094	0.2365	0.2526	12.95%	**20.66%**	1.81	4.19	2.3 X
$12 \times [20]$ ReLU	∞	0.1996	0.2496	0.2740	25.05%	**37.28%**	4.39	9.46	2.2 X
$4 \times [20]$ Sigmoid	∞	0.1019	0.1418	0.1388	**39.08%**	36.21%	0.74	2.11	2.8 X
$8 \times [20]$ Sigmoid	∞	0.0858	0.1618	0.1626	88.62%	**89.59%**	4.15	32.15	7.7 X
$12 \times [20]$ Sigmoid	∞	0.0782	0.1510	0.1081	**93.06%**	38.22%	8.71	152.91	17.6 X
MNIST classifiers									
$5 \times [20]$ ReLU	1	3.8018	4.0835	4.2215	7.41%	**11.04%**	1.69	195.07	115.5 X
	2	0.5346	0.5710	0.5587	**6.81%**	4.52%	1.12	41.63	37.1 X
	∞	0.0261	0.0278	0.0287	6.52%	**9.89%**	1.26	11.93	9.5 X
$20 \times [20]$ ReLU	1	2.3853	3.1062	3.1735	30.22%	**33.04%**	35.53	1301.11	36.6 X
	2	0.3656	0.4925	0.5074	34.71%	**38.78%**	31.10	229.87	7.4 X
	∞	0.0183	0.0240	0.0245	30.84%	**33.75%**	43.31	199.40	4.6 X
$5 \times [20]$ Sigmoid	1	1.8009	2.1340	2.1126	**18.49%**	17.31%	1.75	310.27	177.2 X
	2	0.3100	0.3581	0.3417	**15.51%**	10.20%	1.23	44.00	35.9 X
	∞	0.0153	0.0174	0.0170	**13.94%**	11.11%	1.39	17.19	12.4 X
$20 \times [20]$ Sigmoid	1	1.5348	1.9779	1.9730	**28.87%**	28.55%	31.80	4904.99	154.3 X
	2	0.2524	0.3261	0.2657	**29.21%**	5.28%	31.77	1354.54	42.6 X
	∞	0.0131	0.0166	0.0153	**27.01%**	17.12%	44.86	1859.68	41.5 X
$5 \times [100]$ ReLU	1	3.8928	4.2343	N/A	**8.77%**	N/A	13.11	6000.00↑	457.8 X↑
	2	0.5499	0.5789	0.5934	5.28%	**7.91%**	10.90	826.79	75.9 X
	∞	0.0261	0.0276	0.0278	5.85%	**6.57%**	10.42	446.07	42.8 X
$7 \times [100]$ ReLU	1	3.5509	3.9907	N/A	**12.38%**	N/A	55.07	10000.00↑	181.6 X↑
	2	0.4997	0.5347	N/A	**7.01%**	N/A	49.49	10000.00↑	202.1 X↑
	∞	0.0241	0.0258	N/A	**7.09%**	N/A	44.94	10000.00↑	222.5 X↑
$5 \times [100]$ Sigmoid	1	2.0998	2.5184	N/A	**19.93%**	N/A	13.64	10000.00↑	733.0 X↑
	2	0.3233	0.3774	0.2494	**16.74%**	−22.84%	12.38	9368.58	757.1 X
	∞	0.0156	0.0186	0.0177	**18.76%**	13.10%	13.09	1319.25	100.8 X
$7 \times [100]$ Sigmoid	1	1.7294	2.2380	N/A	**29.40%**	N/A	48.96	10000.00↑	204.3 X↑
	2	0.2833	0.3427	N/A	**20.96%**	N/A	42.04	10000.00↑	237.9 X↑
	∞	0.0140	0.0168	N/A	**19.96%**	N/A	42.00	10000.00↑	238.1 X↑

Table 4: (Experiment II) Averaged certified l_p bounds of different classifiers on CIFAR10 Networks.

Network	p	Certified Bounds		Improve-ment	Network	p	Certified Bounds		Improve-ment	Network	p	Certified Bounds		Improve-ment
		CROWN	FROWN				CROWN	FROWN				CROWN	FROWN	
$4 \times [2048]$ ReLU	1	7.5460	7.5989	0.70%	$6 \times [2048]$ ReLU	1	4.5990	4.7241	2.72%	$8 \times [2048]$ ReLU	1	4.2349	4.9416	16.69%
	2	0.6175	0.6303	2.09%		2	0.3745	0.3723	−0.60%		2	0.3476	0.3809	9.59%
	∞	0.0151	0.0157	4.14%		∞	0.0092	0.0093	0.95%		∞	0.0087	0.0094	8.83%
$4 \times [2048]$ Sigmoid	1	3.6558	4.4726	22.34%	$6 \times [2048]$ Sigmoid	1	1.5093	1.8430	22.11%	$8 \times [2048]$ Sigmoid	1	1.2875	1.6862	30.97%
	2	0.2847	0.3353	17.77%		2	0.1180	0.1448	22.71%		2	0.1006	0.1295	28.65%
	∞	0.0069	0.0082	18.33%		∞	0.0029	0.0035	21.03%		∞	0.0024	0.0033	37.65%

this experiment, we further speed up FROWN by optimizing neurons in a layer group by group, instead of one by one, and we provide a parameter to balance the trade-off between tightness of bounds and time cost in FROWN(see details in Appendix Section A.10). We observe consistent trends on CIFAR10 networks: the deeper the neural network is, the more significant improvement can be made by FROWN in tightening CROWN solutions. Notably, an up to 38% improvement in certified bounds is achieved when considering l_∞-norm balls in Sigmoid networks.

Discussion

Overall, we have shown the trade-off between the computational costs and certified adversarial distortions in ReLU networks: the LP-based approach certifies larger bounds than FROWN at the cost of 2 to 178 times longer runtime. However, the LP-based approach suffers poor scalability and soon becomes computationally infeasible as the network grows deeper or wider. In contrast, FROWN manages to increase the certified region of CROWN much more efficiently and wins over the LP-based approach in almost all Sigmoid/Tanh networks. Notably, in some cases, the LP-based method gives even worse result than CROWN (those with negative improvements). We conclude two possible reasons: i) the Gurobi LP solver is not guaranteed to converge to the optimal solution and ii) statistical fluctuations caused by random sample selections. More discussions on this are included in Appendix Section A.9.

Conclusion

In this paper, we have proved the optimality of CROWN in the relaxed LP framework under mild conditions. Furthermore, we have proposed a general and versatile optimization framework named *FROWN* for optimizing state-of-the-art formal robustness verifiers including CROWN, CNN-Cert, and POPQORN. Experiments on various networks have verified the usefulness of FROWN in providing tightened robustness certificates at a significantly lower cost than the LP-based method.

Acknowledgement

This work is partially supported by the General Research Fund (Project 14236516) of the Hong Kong Research Grants Council, and MIT-Quest program.

References

Boopathy, A.; Weng, T.-W.; Chen, P.-Y.; Liu, S.; and Daniel, L. 2019. Cnn-cert: An efficient framework for certifying robustness of convolutional neural networks. In *AAAI*.

Carlini, N., and Wagner, D. 2017. Towards evaluating the robustness of neural networks. In *SP*.

Carlini, N., and Wagner, D. A. 2018. Audio adversarial examples: Targeted attacks on speech-to-text. *CoRR* abs/1801.01944.

Cisse, M. M.; Adi, Y.; Neverova, N.; and Keshet, J. 2017. Houdini: Fooling deep structured visual and speech recognition models with adversarial examples. In *NeurIPS*.

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

Gao, P.; Jiang, Z.; You, H.; Lu, P.; Hoi, S. C. H.; Wang, X.; and Li, H. 2019a. Dynamic fusion with intra- and inter-modality attention flow for visual question answering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Gao, P.; You, H.; Zhang, Z.; Wang, X.; and Li, H. 2019b. Multi-modality latent interaction network for visual question answering. In *The IEEE International Conference on Computer Vision (ICCV)*.

Gong, Y., and Poellabauer, C. 2017. Crafting adversarial examples for speech paralinguistics applications. *CoRR* abs/1711.03280.

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

Hein, M., and Andriushchenko, M. 2017. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *NeurIPS*.

Ko, C.-Y.; Lyu, Z.; Weng, L.; Daniel, L.; Wong, N.; and Lin, D. 2019. POPQORN: Quantifying robustness of recurrent neural networks. In *ICML*.

Kurakin, A.; Goodfellow, I.; and Bengio, S. 2017. Adversarial examples in the physical world. *ICLR Workshop*.

Moosavi-Dezfooli, S.-M.; Fawzi, A.; and Frossard, P. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *CVPR*.

Mudrakarta, P. K.; Taly, A.; Sundararajan, M.; and Dhamdhere, K. 2018. Did the model understand the question?

Nesterov, Y. 2014. *Introductory Lectures on Convex Optimization: A Basic Course*. Springer Publishing Company, Incorporated, 1 edition.

Papernot, N.; McDaniel, P. D.; Swami, A.; and Harang, R. E. 2016. Crafting adversarial input sequences for recurrent neural networks. *MILCOM*.

Raghunathan, A.; Steinhardt, J.; and Liang, P. 2018. Certified defenses against adversarial examples. *ICLR*.

Salman, H.; Yang, G.; Zhang, H.; Hsieh, C.; and Zhang, P. 2019. A convex relaxation barrier to tight robustness verification of neural networks. *CoRR* abs/1902.08722.

Sharif, M.; Bhagavatula, S.; Bauer, L.; and Reiter, M. K. 2016. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 1528–1540.

Singh, G.; Gehr, T.; Mirman, M.; Püschel, M.; and Vechev, M. 2018. Fast and effective robustness certification. In *NeurIPS*. 10825–10836.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2014. Intriguing properties of neural networks. *ICLR*.

Weng, T.-W.; Zhang, H.; Chen, P.-Y.; Yi, J.; Su, D.; Gao, Y.; Hsieh, C.-J.; and Daniel, L. 2018a. Evaluating the robustness of neural networks: An extreme value theory approach. In *ICLR*.

Weng, T.-W.; Zhang, H.; Chen, H.; Song, Z.; Hsieh, C.-J.; Boning, D.; Dhillon, I. S.; and Daniel, L. 2018b. Towards fast computation of certified robustness for relu networks. *ICML*.

Weng, T.-W.; Zhang, H.; Chen, P.-Y.; Lozano, A.; Hsieh, C.-J.; and Daniel, L. 2018c. On extensions of clever: A neural network robustness evaluation algorithm. In *GlobalSIP*.

Weng, L.; Chen, P.-Y.; Nguyen, L.; Squillante, M.; Boopathy, A.; Oseledets, I.; and Daniel, L. 2019. PROVEN: Verifying robustness of neural networks with a probabilistic approach. In *ICML*.

Wong, E., and Kolter, Z. 2018. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *ICML*, volume 80, 5286–5295.

Zeng, X.; Liu, C.; Wang, Y.-S.; Qiu, W.; Xie, L.; Tai, Y.-W.; Tang, C.-K.; and Yuille, A. L. 2019. Adversarial attacks beyond the image space. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhang, H.; Weng, T.-W.; Chen, P.-Y.; Hsieh, C.-J.; and Daniel, L. 2018. Efficient neural network robustness certification with general activation functions. In *NeurIPS*. 4944–4953.