Fastened CROWN: Tightened Neural Network Robustness Certificates

Zhaoyang Lyu,1* Ching-Yun Ko,2* Zhifeng Kong,3 Ngai Wong,4 Dahua Lin,1 Luca Daniel2
1The Chinese University of Hong Kong, Hong Kong, China
2Massachusetts Institute of Technology, Cambridge, MA 02139, USA
3University of California San Diego, La Jolla, CA 92093, USA
4The University of Hong Kong, Hong Kong, China
lyuzhaoyang@link.cuhk.edu.hk, cyko@mit.edu, z4kong@eng.ucsd.edu
nwong@eee.hku.hk, dhl@ie.cuhk.edu.hk, luca@mit.edu

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 of [Salman et al.] 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 (Fastened CROWN): 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 agonzied 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., Kurakin, Goodfellow, and Bengio 2017; Carlini and Wagner 2017; Wong and Kolter 2018]. Addressing this urgent 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; Gao et al. 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 Andruschenko 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
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 $\ell_\infty$-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. 2019] 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 2018] 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. 2019] 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 (Fastened CROWN) 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].

### 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 $a^{(k)}$, $W^{(k)}$, $b^{(k)}$, and $z^{(k)}$, respectively. Hence, $z^{(k)} = W^{(k)}a^{(k-1)} + b^{(k)}$, $a^{(k)} = \sigma(z^{(k)})$, where $a^{0} = x_0 \in \mathbb{R}^n$ is the original input and $F(x) = z^{(m)}$ is the network output. Denoting the number of neurons as $n_k$ for the $k$-th layer, implies that $a^{(k)}$, $z^{(k)}$, $b^{(k)} \in \mathbb{R}^{n_k}$ and $W^{(k)} \in \mathbb{R}^{n_k \times n_{k-1}}$, for $k \in [m]$. Furthermore, we use square brackets in the superscripts to group a set of variables (e.g. $a^{[m]}$ denotes the set of variables $\{a^{(1)}, a^{(2)}, ..., a^{(m)}\}$ and $z^{[m]}$ denotes the set of variables $\{z^{(1)}, z^{(2)}, ..., z^{(m)}\}$). 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 $x \in \mathbb{R}^n$ denote the perturbed input of $x_0$ (class $i$) within an $\epsilon$-bounded $l_p$-ball (i.e., $x \in B_p(x_0, \epsilon)$, or $\|x - x_0\|_p \leq \epsilon$), the task of robustness analysis for this network intrinsically involves the comparison between the $i$-th network output $F_i(x)$ and other outputs $F_{\neq i}(x)$. In practice, one can translate the original problem to the problem of deriving a lower bound of $F_i(x)$ and upper bounds of $F_{\neq i}(x)$ 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 $F_i(x)$. The LP problem for the

| Notation | Definition |
|----------|------------|
| $F : \mathbb{R}^n \rightarrow \mathbb{R}$ | neural network classifier |
| $n$ | input size |
| $n_k$ | number of neurons in layer $k$ |
| $F_i^L(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ | linear lower bound of $F_i(x)$ |
| $F_i^U(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ | linear upper bound of $F_i(x)$ |
| $s_{k-1}^{[1]}$ | set $\{0, 1, \ldots, s_{k-1}^{[1]}\}$ |
| $s_{k-1}^{[2]}$ | set $\{0, 1, \ldots, s_{k-1}^{[2]}\}$ |
| $a[k]$ | set $\{a^{(1)}, a^{(2)}, \ldots, a^{(k)}\}$ |

| Notation | Definition |
|----------|------------|
| $x_0 \in \mathbb{R}^n$ | original input |
| $x^{(k)}$ | the hidden state of the $k$-th layer |
| $[K]$ | set $\{1, 2, \ldots, K\}$ |

| Notation | Definition |
|----------|------------|
| $\mathbb{B}_p(x_0, \epsilon)$ | the pre-activation of the $k$-th layer |
| $\{x \mid \|x - x_0\|_p \leq \epsilon\}$ | $x \in \mathbb{R}^n$ |

### 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 $a^{(k)}$, $W^{(k)}$, $b^{(k)}$, and $z^{(k)}$, respectively. Hence, $z^{(k)} = W^{(k)}a^{(k-1)} + b^{(k)}$, $a^{(k)} = \sigma(z^{(k)})$, where $a^{0} = x_0 \in \mathbb{R}^n$ is the original input and $F(x) = z^{(m)}$ is the network output. Denoting the number of neurons as $n_k$ for the $k$-th layer, implies that $a^{(k)}$, $z^{(k)}$, $b^{(k)} \in \mathbb{R}^{n_k}$ and $W^{(k)} \in \mathbb{R}^{n_k \times n_{k-1}}$, for $k \in [m]$. Furthermore, we use square brackets in the superscripts to group a set of variables (e.g. $a^{[m]}$ denotes the set of variables $\{a^{(1)}, a^{(2)}, ..., a^{(m)}\}$ and $z^{[m]}$ denotes the set of variables $\{z^{(1)}, z^{(2)}, ..., z^{(m)}\}$). 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 $x \in \mathbb{R}^n$ denote the perturbed input of $x_0$ (class $i$) within an $\epsilon$-bounded $l_p$-ball (i.e., $x \in B_p(x_0, \epsilon)$, or $\|x - x_0\|_p \leq \epsilon$), the task of robustness analysis for this network intrinsically involves the comparison between the $i$-th network output $F_i(x)$ and other outputs $F_{\neq i}(x)$. In practice, one can translate the original problem to the problem of deriving a lower bound of $F_i(x)$ and upper bounds of $F_{\neq i}(x)$ 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 $F_i(x)$. The LP problem for the
upper bound of \( F_{j 
eq i}(x)\) can be similarly derived.

**The LP problem.** The optimization problem for solving the lower bound of \( F_i(x) = z_{i}^{(m)} = W_{i}^{(m)} a^{(m-1)} + b_{i}^{(m)} \) reads:

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

(1)

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

\[
l(k) \leq z(k) \leq u(k),
\]

(2)

and therefore every element \( \sigma(z(k)) \) of the nonlinear activation \( \sigma(z(k)) \) in constraint (1) can be bounded by [1]:

\[
\begin{align*}
h_i^{(k)L}(z_i^{(k)}) & \leq \sigma(z_i^{(k)}) \leq h_i^{(k)U}(z_i^{(k)}), \forall z_i^{(k)} \in [l_i^{(k)}, u_i^{(k)}], \\
\end{align*}
\]

(3)

for \( i \in [n_x] \). The existence of linear bounding functions in (3) is guaranteed since \( z_i^{(k)} \) is bounded and compactness is a continuous invariant within any interval. For example, the following are bounding functions:

\[
\begin{align*}
h_i^{(k)L}(z_i^{(k)}) = \min_{z_i^{(k)} \in [l_i^{(k)}, u_i^{(k)}]} (\sigma(z_i^{(k)}), h_i^{(k)L}(z_i^{(k)}), h_i^{(k)U}(z_i^{(k)})), \\
h_i^{(k)L}(z_i^{(k)}) = \max_{z_i^{(k)} \in [l_i^{(k)}, u_i^{(k)}]} (\sigma(z_i^{(k)}), h_i^{(k)L}(z_i^{(k)}), h_i^{(k)U}(z_i^{(k)})).
\end{align*}
\]

(4)

The optimization problem can therefore be relaxed to an LP-alike problem\(^2\):

\[
\begin{align*}
\min_{a^{(0)} \in \mathbb{B}_p(x_0, \epsilon), a^{(m-1)}, a^{(m-1)}} & \quad W_{i}^{(m)} a^{(m-1)} + b_{i}^{(m)} \\
\text{s.t.} & \quad \begin{cases} 
z(k) = W(k) a(k-1) + b(k), \forall k \in [m-1], \\
\end{cases}
\end{align*}
\]

(5)

and therefore every element \( \sigma(z(k)) \) of the nonlinear activation \( \sigma(z(k)) \) in constraint (1) can be bounded by [1]:

\[
\begin{align*}
h_i^{(k)L}(z_i^{(k)}) & \leq \sigma(z_i^{(k)}) \leq h_i^{(k)U}(z_i^{(k)}), \forall z_i^{(k)} \in [l_i^{(k)}, u_i^{(k)}], \\
\end{align*}
\]

(6)

and therefore every element \( \sigma(z(k)) \) of the nonlinear activation \( \sigma(z(k)) \) in constraint (1) can be bounded by [1]:

\[
\begin{align*}
h_i^{(k)L}(z_i^{(k)}) & \leq \sigma(z_i^{(k)}) \leq h_i^{(k)U}(z_i^{(k)}), \forall z_i^{(k)} \in [l_i^{(k)}, u_i^{(k)}], \\
\end{align*}
\]

(7)

where \( \text{neg}(x) = x, \) if \( x \leq 0; \) \( \text{neg}(x) = 0, \) otherwise. And

\[
\begin{align*}
W_{i,j}^{(k-1)} & = [\text{relu}(W_{i,j}^{(k)}) \circ (s(k-1)L)^T + \text{neg}(W_{i,j}^{(k)}) \circ (s(k-1)U)^T]W_{i,j}^{(k-1)}, \\
b_{i,j}^{(k-1)} & = \text{relu}(W_{i,j}^{(k)}) \circ (s(k-1)L)^T + \text{neg}(W_{i,j}^{(k)}) \circ (s(k-1)U)^T) b_{i,j}^{(k-1)} + \\
\text{relu}(W_{i,j}^{(k)}) (l(k-1)L + \text{neg}(W_{i,j}^{(k)}) (l(k-1)U + b_{i,j}^{(k)}), \end{align*}
\]

where \( \circ \) 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 \( z_i^{(k)} \) is upper bounded by

\[
\begin{align*}
\sum_{j=1}^{n_x} W_{i,j}^{(1)} x_j + b_{i}^{(1)}, \end{align*}
\]

where \( W_{i,j}^{(1)} \) and \( b_{i}^{(1)} \) are similarly defined as shown above. Taking the dual form of the bound then yields the closed-form bound \( \gamma_i^{(k)} \) that satisfies

\[
\begin{align*}
z_i^{(k)} & \geq \gamma_i^{(k)} := W_{i}^{(1)} x_0 - \epsilon \| W_{i}^{(1)} \|_q + b_{i}^{(1)}, \\
\forall x & \in \mathbb{B}_p(x_0, \epsilon), \end{align*}
\]

(8)

where \( \| \cdot \|_q \) is the \( \ell_q \) norm for \( q = 1, \ldots, m \). The above steps are used to derive the closed-form lower bound, the closed-form upper bound \( \gamma_i^{(k)} \) 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 \( z_i^{(k)} \), 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 \( \gamma_i^{(k)} \) are implicitly reliant to slopes of bounding lines in previous layers \( \frac{1}{m}(k-1)U = \frac{1}{m}(k-1)L \).

\(^2\)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 Equation (2).

\(^3\)\( z_i^{(1)} \) is deterministically computed by the input and \( z_i^{(m)} = F(x) \) is the output bound.
\{s^{(1)L}, \ldots, s^{(k-1)L}\}, s^{(k=L)} = \{s^{(1)L}, \ldots, s^{(k-1)L}\} and their intercepts \{l^{(1)L}, \ldots, l^{(k-1)L}\}, l^{(k=L)} = \{l^{(1)L}, \ldots, l^{(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 \(\{\hat{s}^{(v=1)L}, \hat{s}^{(v=L)}\}, \hat{t}^{(v=1)L}, \hat{t}^{(v=L)}\} are used to calculate \(\gamma(v)^{L}\) and \(\gamma(v)^{U}\), \(\{\hat{s}^{(k-1)L}, \hat{s}^{(k=L)}\}, \hat{t}^{(k-1)L}, \hat{t}^{(k=L)}\} are used to calculate \(\gamma(k)_{L}\) and \(\gamma(k)_{U}\), then the following condition holds,

\[
\begin{align*}
\hat{s}^{(v=1)L} & = \hat{s}^{(v=L)}, \hat{s}^{(v=L)} = \hat{s}^{(v=1)L}, \\
\hat{t}^{(v=1)L} & = \hat{t}^{(v=L)}, \hat{t}^{(v=L)} = \hat{t}^{(v=1)L},
\end{align*}
\]

for \(\forall i \in [n_u], \forall j \in [n_k], 2 \leq v \leq k \leq m \) 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 \(h_{(k)}^{(k)}(z_{(k)})\) and \(h_{(k=L)}^{(k)}(z_{(k)})\) in Problem (5) is chosen to be one linear function 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.

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 (Fastened CROWN) 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:

\[
\begin{align*}
\max_{s^{(k-1)L}, s^{(k=L)}}, i \in [l_{(k-1)}^{(v)}, l_{(k-1)}^{(u)}], j \in [l_{(k)}^{(v)}, l_{(k)}^{(u)}]} \gamma_{(k)}^{(k)_L} \\
\text{s.t. } s_{i}^{(v)} z_{i}^{(v)} + t_{i}^{(v)} \leq \sigma(z_{i}^{(v)}) \leq s_{i}^{(v)} z_{i}^{(v)} + t_{i}^{(v)} U, \\
\forall z_{i}^{(v)} \in [l_{i}^{(v)}, u_{i}^{(v)}], i \in [n_v], v \in [k-1],
\end{align*}
\]

and

\[
\begin{align*}
\min_{s^{(k-1)L}, s^{(k=L)}}, i \in [l_{(k-1)}^{(v)}, l_{(k-1)}^{(u)}], j \in [l_{(k)}^{(v)}, l_{(k)}^{(u)}]} \gamma_{(k)}^{(k)_U} \\
\text{s.t. } s_{i}^{(v)} z_{i}^{(v)} + t_{i}^{(v)} \leq \sigma(z_{i}^{(v)}) \leq s_{i}^{(v)} z_{i}^{(v)} + t_{i}^{(v)} U, \\
\forall z_{i}^{(v)} \in [l_{i}^{(v)}, u_{i}^{(v)}], i \in [n_v], v \in [k-1].
\end{align*}
\]

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 $l(s_0, l)$, respectively.

| Nonlinearity | ReLU (Lower bnd.) | Sigmoid & Tanh (Upper bnd.) | Sigmoid & Tanh (Lower bnd.) |
|--------------|-------------------|----------------------------|-----------------------------|
| Pre-activation Bounds | $l < u \leq 0$ | $l < u \leq 0$ | $l < u \leq 0$ |
| $l < u \leq 0$ | $l < u \leq 0$ | $l < u \leq 0$ | $l < u \leq 0$ |
| Variable | $s_0$ | $s_0$ | $s_0$ |
| Range | $[0, l]$ | $[\hat{t}_l, u]$ | $[\hat{t}_l, u]$ |
| Slope | $d_1$ | $d_1$ | $d_2$ |
| Intercept | $l(s_0, l)$ | $l(s_0, l)$ | $l(s_0, l)$ |

Notes: Case 1 refers to $\sigma(u) + t(\sigma(v), u) = a(1)$; and case $2$, otherwise. Case 3 refers to $\sigma(u) + t(\sigma(v), l) = a(1)$; and case $4$, otherwise.

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

smaller ones. We present our solutions by first introducing the idea of “tighter” bounding lines.

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

\[
\begin{aligned}
\hat{h}_i^{(k)L}(Z_i^{(k)}) &> \hat{h}_i^{(k)L}(Z_i^{(k)}) , \forall Z_i^{(k)} \in \left[ l_1^{(k)}(u_i^{(k)}) \right] \\
\sigma(Z_i^{(k)}) &\geq \hat{h}_i^{(k)L}(Z_i^{(k)}) , \forall Z_i^{(k)} \in \left[ l_1^{(k)}(u_i^{(k)}) \right]
\end{aligned}
\]

then we say $\hat{h}_i^{(k)L}(Z_i^{(k)}) = \hat{s}_i^{(k)L}Z_i^{(k)} + \hat{t}_i^{(k)L}$ is a tighter lower-bounding line than $\tilde{h}_i^{(k)L}(Z_i^{(k)}) = \tilde{s}_i^{(k)L}Z_i^{(k)} + \tilde{t}_i^{(k)L}$ for the nonlinear activation $\sigma$ in the interval $[l_1^{(k)}, u_i^{(k)}]$.

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 ($l_1^{(k)}, u_i^{(k)} < 0 < u_i^{(k)}$).

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 $\{\hat{g}^{[k]-1}L, \hat{g}^{[k]-1}L, \hat{g}^{[k]-1}L, \hat{g}^{[k]-1}L\}$ and $\{\hat{g}^{[k]-1}U, \hat{g}^{[k]-1}U, \hat{g}^{[k]-1}U, \hat{g}^{[k]-1}U\}$, and in both of which the self-consistency condition is met, then the closed-form bounds obtained via CROWN satisfy

\[
\begin{aligned}
\gamma^{(k)}_i(\hat{g}^{[k]-1}U, \hat{g}^{[k]-1}L, \hat{g}^{[k]-1}U, \hat{g}^{[k]-1}L) &\geq \\
\gamma^{(k)}_i(\hat{g}^{[k]-1}U, \hat{g}^{[k]-1}L, \hat{g}^{[k]-1}U, \hat{g}^{[k]-1}L) &\leq \\
\gamma^{(k)}_i(\hat{g}^{[k]-1}U, \hat{g}^{[k]-1}L, \hat{g}^{[k]-1}U, \hat{g}^{[k]-1}L) &\leq \\
\gamma^{(k)}_i(\hat{g}^{[k]-1}U, \hat{g}^{[k]-1}L, \hat{g}^{[k]-1}U, \hat{g}^{[k]-1}L) &\leq \\
\end{aligned}
\]

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

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 \( \{ s[k-1]U, s[k-1]L, \hat{t}[k-1]U, \hat{t}[k-1]L \} \) and \( \{ s[k-1]U, s[k-1]L, \tilde{t}[k-1]U, \tilde{t}[k-1]L \} \), then the closed-form bounds obtained via CROWN satisfy

\[
\begin{align*}
\gamma_i^{(k)L}(s[k-1]U, s[k-1]L, \hat{t}[k-1]U, \hat{t}[k-1]L) & \geq \gamma_i^{(k)L}(s[k-1]U, s[k-1]L, \tilde{t}[k-1]U, \tilde{t}[k-1]L), \\
\gamma_i^{(k)U}(s[k-1]U, s[k-1]L, \hat{t}[k-1]U, \hat{t}[k-1]L) & \leq \gamma_i^{(k)U}(s[k-1]U, s[k-1]L, \tilde{t}[k-1]U, \tilde{t}[k-1]L),
\end{align*}
\]

for all \( i \in [n_k] \), when \( \hat{t}^{(v-1)U} \leq \hat{t}^{(v-1)L} \) and \( \hat{t}^{(v-1)U} \leq \hat{t}^{(v-1)L} \).

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 \( x_0 \in \mathbb{R}^n \), \( l_p \) ball parameters \( p \geq 1 \), and \( \epsilon \geq 0 \), for all \( j \in [n_m] \), \( \frac{1}{q} = 1 - \frac{1}{p} \), we can compute two fixed values \( \gamma^L_j \) and \( \gamma^U_j \) such that \( \forall x \in B_p(x_0, \epsilon) \), the inequality \( \gamma^L_j \leq F_j(x) \leq \gamma^U_j \) holds. Suppose the label of the input sequence is \( i \), the largest possible lower bound \( \epsilon_i \) of untargeted and targeted (target class \( j \)) attacks is found by solving:

\[
\begin{align*}
\text{Untargeted:} & \quad \epsilon_i = \max_{\epsilon} \epsilon, \ s.t. \ \gamma^L_j(\epsilon) \geq \gamma^U_j(\epsilon), \ \forall j \neq i. \\
\text{Targeted:} & \quad \hat{\epsilon}(i, j) = \max_{\epsilon} \epsilon, \ s.t. \ \gamma^L_j(\epsilon) \geq \gamma^U_j(\epsilon).
\end{align*}
\]

We conduct binary search procedures to compute the largest possible \( \epsilon_i \) (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 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 our 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.

---

\(^5\)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 $l_\infty$ 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". "$m \times [N] \sigma$" means an $m$-layer network with $N$ neurons and $\sigma$ activation.

| Network | $p$ | Certified Bounds | Improvement | Avg. Time per Image (s) | Speedups of FROWN over LP |
|---------|-----|------------------|-------------|-----------------------|--------------------------|
| CROWN | FROWN | LP | FROWN | FROWN | LP | FROWN | FROWN | LP |
| 4 × [20] ReLU | $\infty$ | 0.2019 | 0.2317 | 0.2209 | 11.27% | 12.38% | 0.35 | 0.36 | 2.8X |
| 8 × [20] ReLU | $\infty$ | 0.2094 | 0.2365 | 0.2262 | 12.95% | 20.66% | 1.81 | 4.19 | 2.3X |
| 12 × [20] ReLU | $\infty$ | 0.1996 | 0.2406 | 0.2740 | 25.65% | 37.28% | 4.39 | 9.46 | 2.2X |
| 4 × [20] Sigmoid | $\infty$ | 0.1019 | 0.1418 | 0.1388 | 39.08% | 36.21% | 0.74 | 2.11 | 2.8X |
| 8 × [20] Sigmoid | $\infty$ | 0.0858 | 0.1618 | 0.1626 | 88.62% | 89.59% | 4.15 | 32.15 | 7.7X |
| 12 × [20] Sigmoid | $\infty$ | 0.0782 | 0.1510 | 0.1081 | 93.06% | 38.22% | 8.71 | 152.91 | 17.6X |

Table 4: (Experiment II) Averaged certified $l_\infty$ bounds of different classifiers on CIFAR10 Networks.

| Network | $p$ | Certified Bounds | Improvement | Network | $p$ | Certified Bounds | Improvement |
|---------|-----|------------------|-------------|---------|-----|------------------|-------------|
| CROWN | FROWN | FROWN | CROWN | FROWN | FROWN | CROWN | FROWN |
| 4 × [2048] ReLU | 1 | 7.5460 | 7.5897 | 0.70% | 5 | 4.5990 | 4.7211 | 2.72% |
| 2 | 0.6175 | 0.6303 | 2.01% | 2 | 0.0745 | 0.3729 | -60.60% |
| 3 | 0.1351 | 0.1357 | 0.04% | 3 | 0.0095 | 0.0931 | -95.05% |
| 4 × [2048] Sigmoid | 1 | 3.6568 | 4.7216 | 22.34% | 1 | 1.0984 | 1.8430 | 22.11% |
| 2 | 0.2847 | 0.3353 | 17.77% | 2 | 0.1180 | 0.1448 | 22.71% |
| $\infty$ | 0.0069 | 0.0034 | 18.33% | $\infty$ | 0.0029 | 0.0034 | 21.03% |

In 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, the more significant improvement can be made by FROWN in tightening CROWN solutions. Notably, an up to 35% improvement in certified bounds is achieved when considering $l_\infty$-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 a feasible solution and ii) the LP-based approach 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. Further-
more, 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 et al. 2019] Boopathy, A.; Weng, T.-W.; Chen, P.-Y.; Liu, S.; and Daniel, L. 2019. Cnncert: An efficient framework for certifying robustness of convolutional neural networks. In AAAI.

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

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

[Cisse et al. 2017] 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 et al. 2018] Dvijotham, K.; Stanforth, R.; Gowal, S.; Mann, T.; and Kohli, P. 2018. A dual approach to scalable verification of deep networks. UAI.

[Gao et al. 2019a] 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 et al. 2019b] 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 and Poellabauer 2017] Gong, Y., and Poellabauer, C. 2017. Crafting adversarial examples for speech paralinguistics applications. CoRR abs/1711.03280.

[Goodfellow et al. 2015] Goodfellow, I.; Shlens, J.; and Szegedy, C. 2015. Explaining and harnessing adversarial examples. In ICLR.

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

[Ko et al. 2019] 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, Goodfellow, and Bengio 2017] Kurakin, A.; Goodfellow, I.; and Bengio, S. 2017. Adversarial examples in the physical world. ICLR Workshop.

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

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

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

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

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

[Salman et al. 2019] 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 et al. 2016] 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 et al. 2018] Singh, G.; Gehr, T.; Mirman, M.; Plüschel, M.; and Vechev, M. 2018. Fast and effective robustness certification. In NeurIPS, 10825–10836.

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

[Weng et al. 2018a] 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 et al. 2018b] 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 et al. 2018c] 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 et al. 2019a] 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 and Kolter 2018] Wong, E., and Kolter, Z. 2018. Provably secure defenses against adversarial examples via the convex outer adversarial polytope. In ICML, volume 80, 5286–5295.

[Zeng et al. 2019] 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 et al. 2018] 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.