Efficient Neural Network Robustness Certification with General Activation Functions

Huan Zhang\textsuperscript{1,} \textsuperscript{*} Tsui-Wei Weng\textsuperscript{2,}† Pin-Yu Chen\textsuperscript{3} Cho-Jui Hsieh\textsuperscript{1} Luca Daniel\textsuperscript{2}

\textsuperscript{1}University of California, Los Angeles, Los Angeles CA 90095
\textsuperscript{2}Massachusetts Institute of Technology, Cambridge, MA 02139
\textsuperscript{3}MIT-IBM Watson AI Lab, IBM Research, Yorktown Heights, NY 10598

huan@huan-zhang.com, twweng@mit.edu
pin-yu.chen@ibm.com, chohsieh@cs.ucla.edu, dluca@mit.edu

Abstract

Finding minimum distortion of adversarial examples and thus certifying robustness in neural network classifiers for given data points is known to be a challenging problem. Nevertheless, recently it has been shown to be possible to give a non-trivial certified lower bound of minimum adversarial distortion, and some recent progress has been made towards this direction by exploiting the piece-wise linear nature of ReLU activations. However, a generic robustness certification for general activation functions still remains largely unexplored. To address this issue, in this paper we introduce CROWN, a general framework to certify robustness of neural networks with general activation functions for given input data points. The novelty in our algorithm consists of bounding a given activation function with linear and quadratic functions, hence allowing it to tackle general activation functions including but not limited to four popular choices: ReLU, tanh, sigmoid and arctan. In addition, we facilitate the search for a tighter certified lower bound by adaptively selecting appropriate surrogates for each neuron activation. Experimental results show that CROWN on ReLU networks can notably improve the certified lower bounds compared to the current state-of-the-art algorithm Fast-Lin, while having comparable computational efficiency. Furthermore, CROWN also demonstrates its effectiveness and flexibility on networks with general activation functions, including tanh, sigmoid and arctan.

1 Introduction

While neural networks (NNs) have achieved remarkable performance and accomplished unprecedented breakthroughs in many machine learning tasks, recent studies have highlighted their lack of robustness against adversarial perturbations [1, 2]. For example, in image learning tasks such as object classification [3, 4, 5, 6] or content captioning [7], visually indistinguishable adversarial examples can be easily crafted from natural images to alter a NN’s prediction result. Beyond the white-box attack setting where the target model is entirely transparent, visually imperceptible adversarial perturbations can also be generated in the black-box setting by only using the prediction results of the target model [8, 9, 10, 11]. In addition, real-life adversarial examples have been made possible through the lens of realizing physical perturbations [12, 13, 14]. As NNs are becoming a core technique deployed in a wide range of applications, including safety-critical tasks, certifying robustness of a NN against adversarial perturbations has become an important research topic in machine learning.

\textsuperscript{*}Work done during internship at IBM Research. \textsuperscript{†}Equal contribution.
Correspondence to Huan Zhang <huan@huan-zhang.com> and Tsui-Wei Weng <twweng@mit.edu>

Given a NN (possibly with a deep and complicated network architecture), we are interested in certifying the (local) robustness of an arbitrary natural example $x_0$ by ensuring all its neighborhood has the same inference outcome (e.g., consistent top-1 prediction). In this paper, the neighborhood of $x_0$ is characterized by an $\ell_p$ ball centered at $x_0$, for any $p \geq 1$. Geometrically speaking, the minimum distance of a misclassified nearby example to $x_0$ is the least adversary strength (a.k.a. minimum adversarial distortion) required to alter the target model’s prediction, which is also the largest possible robustness certificate for $x_0$. Unfortunately, finding the minimum distortion of adversarial examples in NNs with Rectified Linear Unit (ReLU) activations, which is one of the most widely used activation functions, is known to be an NP-complete problem [15, 16]. This makes formal verification techniques such as Reluplex [15] computationally demanding even for small-sized NNs and suffer from scalability issues.

Although certifying the largest possible robustness is challenging for ReLU networks, the piece-wise linear nature of ReLUs can be exploited to efficiently compute a non-trivial certified lower bound of the minimum distortion [17, 18, 19, 20]. Beyond ReLU, one fundamental problem that remains largely unexplored is how to generalize the robustness certification technique to other popular activation functions that are not piece-wise linear, such as tanh and sigmoid, and how to motivate and certify the design of other activation functions towards improved robustness. In this paper, we tackle the preceding problem by proposing an efficient robustness certification framework for NNs with general activation functions. Our main contributions in this paper are summarized as follows:

- We propose a generic analysis framework CROWN for certifying NNs using linear or quadratic upper and lower bounds for general activation functions that are not necessarily piece-wise linear.
- Unlike previous work [20], CROWN allows flexible selections of upper/lower bounds for activation functions, enabling an adaptive scheme that helps to reduce approximation errors. Our experiments show that CROWN achieves up to 26% improvements in certified lower bounds compared to [20].
- Our algorithm is efficient and can scale to large NNs with various activation functions. For a NN with over 10,000 neurons, we can give a certified lower bound in about 1 minute on 1 CPU core.

## 2 Background and Related Work

For ReLU networks, finding the minimum adversarial distortion for a given input data point $x_0$ can be cast as a mixed integer linear programming (MILP) problem [21, 22, 23]. Reluplex [15, 24] uses a satisfiable modulo theory (SMT) to encode ReLU activations into linear constraints. Similarly, Planet [25] uses satisfiability (SAT) solvers. However, due to the NP-completeness for solving such a problem [15], these methods can only find minimum distortion for very small networks. It can take Reluplex several hours to find the minimum distortion of an example for a ReLU network with 5 inputs, 5 outputs and 300 neurons[15].

On the other hand, a computationally feasible alternative of robustness certificate is to provide a non-trivial and certified lower bound of minimum distortion. Some analytical lower bounds based on operator norms on the weight matrices [3] or the Jacobian matrix in NNs [17] do not exploit special property of ReLU and thus can be very loose [20]. The bounds in [26, 27] are based on the local Lipschitz constant. [26] assumes a continuous differentiable NN and hence excludes ReLU networks; a closed form lower-bound is also hard to derive for networks beyond 2 layers. [27] applies to ReLU networks and uses Extreme Value Theory to provide an estimated lower bound (CLEVER score). Although the CLEVER score is capable of reflecting the level of robustness in different NNs and is scalable to large networks, it is not a certified lower bound. On the other hand, [18] use the idea of a convex outer adversarial polytope in ReLU networks to compute a certified lower bound by relaxing the MILP certification problem to linear programing (LP). [20] exploit the ReLU property to bound the activation function (or the local Lipschitz constant) and provide efficient algorithms (Fast-Lin and Fast-Lip) for computing a certified lower bound, achieving state-of-the-art performance. A recent work [28] uses abstract transformations to zonotopes for proving robustness property for ReLU networks. Nonetheless, there are still some applications demand non-ReLU activations, e.g. RNN and LSTM, thus a general framework that can efficiently compute non-trivial and certified lower bounds for NNs with general activation functions is of great importance. We aim at filling this gap and propose CROWN that can perform efficient robustness certification to NNs with general activation functions. Table 1 summarizes the differences of other approaches and CROWN. Note that the semidefinite programming approach proposed in [19] and a recent work [29] based on solving Lagrangian dual can both handle general activation functions, but [19] is limited to NNs with one hidden layer and [29] trades off the quality of robustness bound with scalability.
Table 1: Comparison of methods for providing adversarial robustness certification in NNs.

<table>
<thead>
<tr>
<th>Method</th>
<th>Non-trivial bound</th>
<th>Multi-layer</th>
<th>Scalability</th>
<th>Beyond ReLU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Szegedy et. al. [3]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Reluplex [15], Planet [25]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Hein &amp; Andriushchenko [26]</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Raghu et al. et al. [19]</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Koller and Wong [18]</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>CROWN (ours)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

* Continuously differentiable activation function required (soft plus is demonstrated in [26]).

Some recent works (such as robust optimization based adversarial training [30] or region-based classification [31]) empirically exhibit strong robustness against several adversarial attacks, which is beyond the scope of provable robustness certification. In addition, Sinha et al. [16] provide distributional robustness certification based on Wasserstein distance between data distributions, which is different from the local ℓ_p ball robustness model considered in this paper.

3 CROWN: A general framework for certifying neural networks

Overview of our results. In this section, we present a general framework CROWN for efficiently computing a certified lower bound of minimum adversarial distortion given any input data point x_0 with general activation functions in larger NNs. We first provide principles in Section 3.1 to derive output bounds of NNs when the inputs are perturbed within an ℓ_p ball and each neuron has different (adaptive) linear approximation bounds on its activation function. In Section 3.2, we demonstrate how to provide robustness certification for four widely-used activation functions (ReLU, tanh, sigmoid and arctan) using CROWN. In particular, we show that the state-of-the-art Fast-Lin algorithm is a special case under the CROWN framework and that the adaptive selections of approximation bounds allow CROWN to achieve a tighter (larger) certified lower bound (see Section 4). In Section 3.3, we further highlight the flexibility of CROWN to incorporate quadratic approximations on the activation functions in addition to the linear approximations described in Section 3.1.

3.1 General framework

Notations. For an m-layer neural network with an input vector x ∈ R^{n_0}, let the number of neurons in each layer be n_k, ∀k ∈ [m], where [·] denotes set {1, 2, ··· , t}. Let the k-th layer weight matrix be W(k) ∈ R^{n_k × n_k−1} and bias vector be b(k) ∈ R^{n_k}, and let Φ_k : R^{n_0} → R^{n_k} be the operator mapping from input to layer k. We have Φ_k(x) = σ(W(k)Φ_{k−1}(x) + b(k)), ∀k ∈ [m−1], where σ(·) is the coordinate-wise activation function. While our methodology is applicable to any activation function of interest, we emphasize on four most widely-used activation functions, namely ReLU: σ(y) = max(y, 0), hyperbolic tangent: σ(y) = tanh(y), sigmoid: σ(y) = 1/(1 + e^{-y}) and arctan: σ(y) = arctan(y). Note that the input Φ_0(x) = x, and the vector output of the NN is f(x) = Φ_m(x) = W^{(m)}Φ_{m−1}(x) + b^{(m)}. The j-th output element is denoted as f_j(x) = [Φ_m(x)]_j.

Input perturbation and pre-activation bounds. Let x_0 ∈ R^{n_0} be a given data point, and let the perturbed input vector x be within an ε-bounded ℓ_p-ball centered at x_0, i.e., x ∈ B_p(x_0, ε), where B_p(x_0, ε) := {x | ∥x − x_0∥_p ≤ ε}. For the r-th neuron in k-th layer, let its pre-activation input be y_r^{(k)}, where y_r^{(k)} = W_{r,k}Φ_{k−1}(x) + b_r^{(k)} and W_{r,k} denotes the r-th row of matrix W^{(k)}. When x_0 is perturbed within an ε-bounded ℓ_p-ball, let l_r^{(k)}, u_r^{(k)} ∈ R be the pre-activation lower bound and upper bound of y_r^{(k)}, i.e., l_r^{(k)} ≤ y_r^{(k)} ≤ u_r^{(k)}.

Below, we first define the linear upper bounds and lower bounds of activation functions in Definitions 3.1, which are the key to derive explicit output bounds for an m-layer neural network with general activation functions. The formal statement of the explicit output bounds is shown in Theorem 3.2.

Definition 3.1 (Linear bounds on activation function). For the r-th neuron in k-th layer with pre-activation bounds l_r^{(k)}, u_r^{(k)} and the activation function σ(y), define two linear functions h_{U,r}^{(k)}(y), h_{L,r}^{(k)}(y) : R → R, h_{U,r}^{(k)}(y) = α_{U,r}^{(k)}(y + β_{U,r}^{(k)}), h_{L,r}^{(k)}(y) = α_{L,r}^{(k)}(y + β_{L,r}^{(k)}), such that h_{U,r}^{(k)}(y) ≤ σ(y) ≤ h_{L,r}^{(k)}(y), y ∈ [l_r^{(k)}, u_r^{(k)}], ∀k ∈ [m−1], r ∈ [n_k] and α_{U,r}, α_{L,r} ∈ R^+, β_{U,r}, β_{L,r} ∈ R.
Note that the parameters \( \alpha_{U,r}^{(k)}, \alpha_{L,r}^{(k)}, \beta_{U,r}^{(k)}, \beta_{L,r}^{(k)} \) depend on \( l^{(k)} \) and \( u^{(k)} \), i.e. for different \( l^{(k)} \) and \( u^{(k)} \) we may choose different parameters. Also, for ease of exposition, in this paper we restrict \( \alpha_{U,r}^{(k)}, \alpha_{L,r}^{(k)} \geq 0 \). However, Theorem 3.2 can be easily generalized to the case of negative \( \alpha_{U,r}^{(k)}, \alpha_{L,r}^{(k)} \).

**Theorem 3.2 (Explicit output bounds of neural network \( f \)).** Given an \( m \)-layer neural network function \( f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_m} \), there exists two explicit functions \( f_U^L : \mathbb{R}^{n_0} \to \mathbb{R} \) and \( f_U^L : \mathbb{R}^{n_0} \to \mathbb{R} \) such that \( \forall j \in [n_m] \), \( \forall x \in B_{p}(x_0, \epsilon) \), the inequality \( f_U^L(x) \leq f_j(x) \leq f_L^U(x) \) holds true, where

\[
f_U^L(x) = \alpha_{U}^{(0)} x + \sum_{k=1}^{m} \Lambda_j^{(k)}(b^{(k)} + \Delta_{:,j}), \quad f_L^U(x) = \Omega_{j}^{(0)} x + \sum_{k=1}^{m} \Omega_j^{(k)}(b^{(k)} + \Theta_{:,j}),
\]

### 1. Fast-Lin [20] is a special case of our framework as they require the slopes \( \alpha_{U,r}^{(k)}, \beta_{r}^{(k)} \) to be the same; and it only applies to ReLU networks (cf. Sec. 3.2). In Fast-Lin, \( \Lambda^{(0)} \) and \( \Omega^{(0)} \) are identical.

### 2. Our CROWN framework allows adaptive selections on the linear approximation when computing certified lower bounds of minimum adversarial distortion, which is the main contributor to improve the certified lower bound as demonstrated in the experiments in Section 4.

**Global bounds.** More importantly, since the input \( x \in B_{p}(x_0, \epsilon) \), we can take the maximum, i.e. \( \max_{x \in B_{p}(x_0, \epsilon)} f_U^L(x) \), and minimum, i.e. \( \min_{x \in B_{p}(x_0, \epsilon)} f_U^L(x) \), as a pair of global upper and lower bound of \( f_j(x) \) – which in fact has closed-form solutions because \( f_U^L(x) \) and \( f_L^U(x) \) are two linear functions and \( x \in B_{p}(x_0, \epsilon) \) is a convex norm constraint. This result is formally presented below and its proof is given in Appendix B.

**Corollary 3.3 (Closed-form global bounds).** Given a data point \( x_0 \in \mathbb{R}^{n_0}, \ell_p \) ball parameters \( p \geq 1 \) and \( \epsilon > 0 \). For an \( m \)-layer neural network function \( f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_m} \), there exists two fixed values \( \gamma_U^{L,j} \) and \( \gamma_L^{U,j} \) such that \( \forall x \in B_{p}(x_0, \epsilon) \) and \( \forall j \in [n_m] \), \( 1/q = 1 - 1/p \), the inequality \( \gamma_U^{L,j} \leq f_j(x) \leq \gamma_L^{U,j} \) holds true, where

\[
\gamma_U^{L,j} = \epsilon \| \Lambda_{j}^{(0)} \|_q + \sum_{k=1}^{m} \Lambda_j^{(k)}(b^{(k)} + \Delta_{:,j}), \quad \gamma_L^{U,j} = -\epsilon \| \Omega_{j}^{(0)} \|_q + \sum_{k=1}^{m} \Omega_j^{(k)}(b^{(k)} + \Theta_{:,j}).
\]
Table 2: Linear upper bound parameters of various activation functions: $h^{(k)}_{U,r}(y) = \alpha^{(k)}_{U,r}(y + \beta^{(k)}_{U,r})$

<table>
<thead>
<tr>
<th>Upper bound $h^{(k)}_{U,r}$</th>
<th>$r \in S^{+}$</th>
<th>$r \in S^{k}$</th>
<th>$r \in S^{k+}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReLU</td>
<td>$\alpha^{(k)}_{U,r}$</td>
<td>$\beta^{(k)}_{U,r}$</td>
<td>$\alpha^{(k)}_{U,r}$</td>
</tr>
<tr>
<td>Sigmoid, tanh</td>
<td>$\sigma'(d) - \sigma(d) - d$</td>
<td>$\frac{\sigma(u^{(k)}<em>{U,r}) - \sigma(u^{(k)}</em>{U,r} - \beta^{(k)}<em>{U,r})}{u^{(k)}</em>{U,r}} - 1$</td>
<td>$\sigma'(d)$</td>
</tr>
<tr>
<td>(denoted as $\sigma(y)$)</td>
<td>$(l^{(k)} \leq d \leq u^{(k)}_{(k)})$</td>
<td>$(l^{(k)} \leq d \leq u^{(k)}_{(k)})$</td>
<td>$(l^{(k)} \leq d \leq u^{(k)}_{(k)})$</td>
</tr>
</tbody>
</table>

Certified lower bound of minimum distortion. Given an input example $x_0$ and an $m$-layer NN, let $c$ be the predicted class of $x_0$ and $\ell \neq c$ be the targeted attack class. We aim to use the uniform bounds established in Corollary 3.3 to obtain the largest possible lower bound $\tilde{\epsilon}_t$ and $\tilde{\epsilon}$ of targeted and untargeted attacks respectively, which can be formulated as follows:

$$\tilde{\epsilon}_t = \max \epsilon \text{ s.t. } \gamma_c^L(\epsilon) - \gamma_c^U(\epsilon) > 0 \text{ and } \tilde{\epsilon} = \min_{\ell \neq c} \tilde{\epsilon}_t.$$

We note that although there is a linear $\epsilon$ term in (2), other terms such as $\Lambda^{(k)}$, $\Delta^{(k)}$, and $\Theta^{(k)}$ also implicitly depend on $\epsilon$. This is because the parameters $\alpha^{(k)}_{L,i}, \beta^{(k)}_{L,i}$, $\alpha^{(k)}_{L,i}, \beta^{(k)}_{L,i}$ depend on $I^{(k)}_k, u^{(k)}_k$, which may vary with $\epsilon$; thus the values in $\Lambda^{(k)}$, $\Delta^{(k)}$, $\Omega^{(k)}$, $\Theta^{(k)}$ depend on $\epsilon$. It is therefore difficult to obtain an explicit expression of $\gamma_c^L(\epsilon) - \gamma_c^U(\epsilon)$ in terms of $\epsilon$. Fortunately, we can still perform a binary search to obtain $\tilde{\epsilon}_t$ with Corollary 3.3. More precisely, we first initialize $\tilde{\epsilon}$ to some fixed positive value and apply Corollary 3.3 repeatedly to obtain $I^{(k)}_k$ and $u^{(k)}_k$ from $k = 1$ to $m$ and $r \in [l_k]$. We then check if the condition $\gamma_c^L(\epsilon) - \gamma_c^U(\epsilon) > 0$ is satisfied. If so, we increase $\epsilon$; otherwise, we decrease $\epsilon$; and we repeat the procedure until a given tolerance level is met.

**Time Complexity.** With Corollary 3.3, we can compute analytic output bounds efficiently without resorting to any optimization solvers for general $\ell_p$ distortion, and the time complexity for an $m$-layer ReLU network is polynomial in contrast to ReLuplex or Mixed-Integer Optimization-based approach [22, 23] where SMT and MIO solvers are exponential-time. For an $m$ layer network with $n$ neurons per layer and $n$ outputs, time complexity of CROWN is $O(m^2 n^3)$. Forming $\Lambda^{(0)}$ and $\Omega^{(0)}$ for the $m$-th layer involves multiplications of layer weights in a similar cost of forward propagation in $O(m n^3)$ time. Also, the bounds for all previous $k \in [m - 1]$ layers need to be computed beforehand in $O(k n^3)$ time; thus the total time complexity is $O(m^2 n^3)$.

### 3.2 Case studies: CROWN for ReLU, tanh, sigmoid and arctan activations

In Section 3.1 we showed that as long as one can identify two linear functions $h_U(y), h_L(y)$ to bound a general activation function $\sigma(y)$ for each neuron, we can use Corollary 3.3 with a binary search

---

3 The bound can be further improved by considering $g(x) := f_c(x) - f_l(x)$ and replacing the last layer’s weights by $W^{(m)}_{c_l} - W^{(m)}_{l_l}$. This is also used by [20].
to obtain certified lower bounds of minimum distortion. In this section, we illustrate how to find parameters $\alpha_{U,r}, \alpha_{L,r}$ and $\beta_{U,r}, \beta_{L,r}$ of $h_U(y), h_L(y)$ for four most widely used activation functions: ReLU, tanh, sigmoid and arctan. Other activations, including but not limited to leaky ReLU, ELU and softplus, can be easily incorporated into our CROWN framework following a similar procedure.

**Segmenting activation functions.** Based on the signs of $I^{(k)}_r$ and $u^{(k)}_r$, we define a partition \{${S^+_k, S^+_k, S^-_k}$\} of set $[n_k]$ such that every neuron in $k$-th layer belongs to exactly one of the three sets. The formal definition of $S^+_k$, $S^-_k$ and $S^0_k$ is $S^+_k = \{r \in [n_k] \mid 0 \leq I^{(k)}_r \leq u^{(k)}_r\}$, $S^-_k = \{r \in [n_k] \mid 1^{(k)}_r < u^{(k)}_r\}$, and $S^0_k = \{r \in [n_k] \mid 1^{(k)}_r \leq u^{(k)}_r \leq 0\}$. For neurons in each partitioned set, we define corresponding upper bound $h^{(k)}_{U,r}$ and lower bound $h^{(k)}_{L,r}$ in terms of $I^{(k)}_r$ and $u^{(k)}_r$. As we will see shortly, segmenting the activation functions based on $I^{(k)}_r$ and $u^{(k)}_r$ is useful to bound a given activation function. We note there are multiple ways of segmenting the activation functions and defining the partitioned sets (e.g. based on the values of $I^{(k)}_r$, $u^{(k)}_r$ rather than their signs), and we can easily incorporate this into our framework to provide the corresponding explicit output bounds for the new partition sets. In the case study, we consider $S^+_k, S^-_k$ and $S^0_k$ for the four activations, as this partition reflects the curvature of tanh, sigmoid and arctan functions and activation states of ReLU.

**Bounding tanh/sigmoid/arctan.** For tanh activation, $\sigma(y) = \frac{1-e^{-2y}}{1+e^{-2y}}$; for sigmoid activation, $\sigma(y) = \frac{1}{1+e^{-y}}$; for arctan activation, $\sigma(y) = \arctan(y)$. All functions are convex on one side ($y < 0$) and concave on the other side ($y > 0$), thus the same rules can be used to find $h^{(k)}_{U,r}$ and $h^{(k)}_{L,r}$. Below we call $(I^{(k)}_r, \sigma(I^{(k)}_r))$ as left end-point and $(u^{(k)}_r, \sigma(u^{(k)}_r))$ as right end-point. For $r \in S^+_k$, since $\sigma(y)$ is concave, we can let $h^{(k)}_{U,r}$ be any tangent line of $\sigma(y)$ at point $d \in [I^{(k)}_r, u^{(k)}_r]$, and let $h^{(k)}_{L,r}$ pass the two end-points. Similarly, $\sigma(y)$ is concave for $r \in S^-_k$, thus we can let $h^{(k)}_{L,r}$ be any tangent line of $\sigma(y)$ at point $d \in [I^{(k)}_r, u^{(k)}_r]$ and let $h^{(k)}_{U,r}$ pass the two end-points. Lastly, for $r \in S^0_k$, we can let $h^{(k)}_{U,r}$ be the tangent line that passes the left end-point and $(d, \sigma(d))$ where $d \geq 0$ and $h^{(k)}_{U,r}$ be the tangent line that passes the right end-point and $(d, \sigma(d))$ where $d \leq 0$. The value of $d$ for transcendental functions can be found using a binary search. The plots of upper and lower bounds for tanh and sigmoid are in Figure 1 and 3 (in Appendix). Plots for arctan are similar and so omitted.

**Bounding ReLU.** For ReLU activation, $\sigma(y) = \max(0, y)$. If $r \in S^+_k$, we have $\sigma(y) = y$ and so we can set $h^{(k)}_{U,r} = I^{(k)}_r = y$; if $r \in S^-_k$, we have $\sigma(y) = 0$, and thus we can set $h^{(k)}_{U,r} = I^{(k)}_r = 0$; if $r \in S^0_k$, we can set $h^{(k)}_{U,r} = \frac{u^{(k)}_r}{u^{(k)}_r - I^{(k)}_r}(y - u^{(k)}_r)$ and $h^{(k)}_{L,r} = ay, 0 \leq a \leq 1$. Setting $a = \frac{u^{(k)}_r}{u^{(k)}_r - I^{(k)}_r}$ leads to the linear lower bound used in Fast-Lin [20]. Thus, Fast-Lin is a special case under our framework. We propose to **adaptively** choose $a$, where we set $a = 1$ when $u^{(k)}_r \geq |I^{(k)}_r|$ and $a = 0$ when $u^{(k)}_r < |I^{(k)}_r|$. In this way, the area between the lower bound $h^{(k)}_{L,r} = ay$ and $\sigma(y)$ (which reflects the gap between the lower bound and the ReLU function) is always minimized. As shown in our experiments, the adaptive selection of $h^{(k)}_{L,r}$ based on the value of $u^{(k)}_r$ and $I^{(k)}_r$ helps to achieve a tighter certified lower bound. Figure 4 (in Appendix) illustrates the idea discussed here.
Summary. We summarized the above analysis on choosing valid linear functions \( h^{(k)}_{U,r} \) and \( h^{(k)}_{L,r} \) in Table 2 and 3. In general, as long as \( h^{(k)}_{U,r} \) and \( h^{(k)}_{L,r} \) are identified for the activation functions, we can use Corollary 3.3 to compute certified lower bounds for general activation functions. Note that there remain many other choices of \( h^{(k)}_{U,r} \) and \( h^{(k)}_{L,r} \) as valid upper/lower bounds of \( \sigma(y) \), but ideally, we would like them to be close to \( \sigma(y) \) in order to achieve a tighter lower bound of minimum distortion.

3.3 Extension to quadratic bounds

In addition to the linear bounds on activation functions, the proposed CROWN framework can also incorporate quadratic bounds by adding a quadratic term to \( h^{(k)}_{U,r} \) and \( h^{(k)}_{L,r} \): \( \eta^{(k)}_{U,r}(y) = \eta^{(k)}_{U,r} y^2 + \alpha^{(k)}_{U,r}(y + \beta^{(k)}_{U,r}), h^{(k)}_{L,r}(y) = \eta^{(k)}_{L,r} y^2 + \alpha^{(k)}_{L,r}(y + \beta^{(k)}_{L,r}) \), where \( \eta^{(k)}_{U,r}, \eta^{(k)}_{L,r} \in \mathbb{R} \). Following the procedure of unwrapping the activation functions at the layer \( m - 1 \), we show in Appendix D that the output upper bound and lower bound with quadratic approximations are:

\[
\begin{align*}
    f^U_j(x) &= \Phi_{m-2}(x) \top Q^{(m-1)}_U \Phi_{m-2}(x) + 2p^{(m-1)}_U \Phi_{m-2}(x) + s^{(m-1)}_U, \\
    f^L_j(x) &= \Phi_{m-2}(x) \top Q^{(m-1)}_L \Phi_{m-2}(x) + 2p^{(m-1)}_L \Phi_{m-2}(x) + s^{(m-1)}_L,
\end{align*}
\]

where \( Q^{(m-1)}_U = W^{(m-1)} \top D^{(m-1)}_U W^{(m-1)}, Q^{(m-1)}_L = W^{(m-1)} \top D^{(m-1)}_L W^{(m-1)}, p^{(m-1)}_U, p^{(m-1)}_L, s^{(m-1)}_U, s^{(m-1)}_L \) are defined in Appendix D due to page limit. When \( m = 2 \), \( \Phi_{m-2}(x) = x \) and we can directly optimize over \( x \in \mathbb{B}_p(x_0, \epsilon) \); otherwise, we can use the post activation bounds of layer \( m - 2 \) as the constraints. \( D^{(m-1)}_U \) in (3) is a diagonal matrix with \( i \)-th entry being \( W_{j,i}^{(m)} \eta^{(m-1)}_{U,i} \), if \( W_{j,i}^{(m)} \geq 0 \) or \( W_{j,i}^{(m)} \eta^{(m)}_{U,i} \), if \( W_{j,i}^{(m)} < 0 \). Thus, in general \( Q^{(m-1)}_U \) is indefinite, resulting in a non-convex optimization when finding the global bounds as in Corollary 3.3. Fortunately, by properly choosing the quadratic bounds, we can make the problem \( \max_{x \in \mathbb{B}_p(x_0, \epsilon)} f^U_j(x) \) into a convex Quadratic Programming problem; for example, we can let \( \eta^{(m-1)}_{U,i} = 0 \) for all \( W_{j,i}^{(m)} > 0 \) and let \( \eta^{(m-1)}_{L,i} > 0 \) to make \( D^{(m-1)}_U \) have only negative and zero diagonals for the maximization problem – this is equivalent to applying a linear upper bound and a quadratic lower bound to bound the activation function. Similarly, for \( D^{(m-1)}_L \), we let \( \eta^{(m-1)}_{L,i} = 0 \) for all \( W_{j,i}^{(m)} < 0 \) and let \( \eta^{(m-1)}_{L,i} > 0 \) to make \( D^{(m-1)}_L \) have non-negative diagonals and hence the problem \( \min_{x \in \mathbb{B}_p(x_0, \epsilon)} f^L_j(x) \) is convex. We can solve this convex program with projected gradient descent (PGD) for \( x \in \mathbb{B}_p(x_0, \epsilon) \) and Armijo line search. Empirically, we find that PGD usually converges within a few iterations.

4 Experiments

Methods. For ReLU networks, CROWN-Ada is CROWN with adaptive linear bounds (Sec. 3.2), CROWN-Quad is CROWN with quadratic bounds (Sec. 3.3). Fast-Lin and Fast-Lip are state-of-the-art fast certified lower bound proposed in [20]. Reluplex can solve the exact minimum adversarial distortion but is only computationally feasible for very small networks. LP-Full is based on the LP formulation in [18] and we solve LPs for each neuron exactly to achieve the best possible bound. For networks with other activation functions, CROWN-general is our proposed method.

Model and Dataset. We evaluate CROWN and other baselines on multi-layer perceptron (MLP) models trained on MNIST and CIFAR-10 datasets. We denote a feed-forward network with \( m \) layers and \( n \) neurons per layer as \( m \times n \). For models with ReLU activation, we use pretrained models provided by [20] and also evaluate the same set of 100 random test images and random attack targets as in [20] (according to their released code) to make our results comparable. For training NN models with other activation functions, we search for best learning rate and weight decay parameters to achieve a similar level of accuracy as ReLU models.

Implementation and Setup. We implement our algorithm using Python (numpy with numba). Most computations in our method are matrix operations that can be automatically parallelized by the BLAS library; however, we set the number of BLAS threads to 1 for a fair comparison to other methods. Experiments were conducted on an Intel Skylake server CPU running at 2.0 GHz on Google Cloud. Our code is available at https://github.com/CROWN-Robustness/Crown.
1 Conclusion

We have presented a general framework CROWN to efficiently compute a certified lower bound of minimum distortion in neural networks for any given data point \( x_0 \). CROWN features adaptive bounds for improved robustness certification and applies to general activation functions. Moreover, experiments show that (1) CROWN outperforms state-of-the-art baselines on ReLU networks and (2) CROWN can efficiently certify non-trivial lower bounds for large networks with over 10K neurons and with different activation functions.
Acknowledgement

This work was supported in part by NSF IIS-1719097, Intel faculty award, Google Cloud Credits for Research Program and GPUs donated by NVIDIA. Tsui-Wei Weng and Luca Daniel are partially supported by MIT-IBM Watson AI Lab and MIT-Skoltech program.

References


