Efficient Neural Network Robustness Certification with General Activation Functions

Huan Zhang[1,3,†,*] **Tsui-Wei Weng**[2,†] **Pin-Yu Chen**[3] **Cho-Jui Hsieh**[1] **Luca Daniel**[2]
[1]University of California, Los Angeles, Los Angeles CA 90095
[2]Massachusetts Institute of Technology, Cambridge, MA 02139
[3]MIT-IBM Watson AI Lab, IBM Research
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 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. 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 the 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. To the best of our knowledge, CROWN is the first framework that can efficiently certify non-trivial robustness for general activation functions in neural networks.

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 a NN's robustness against adversarial perturbations has become an important research topic in machine learning.

[*]Work done during internship at IBM Research [†]Equal contribution

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

Given a NN (possibly with a deep and complicated network architecture), we are interested in certifying the (local) robustness of an arbitrary natural example $\mathbf{x}_0$ by ensuring *all* its nearby examples will have the same inference outcome (e.g., consistent top-1 prediction). In this paper, the neighborhood of $\mathbf{x}_0$ is characterized by an ℓ_p ball centered at $\mathbf{x}_0$, for any $p \geq 1$. Geometrically speaking, the minimum distance/distortion of a misclassified nearby example to $\mathbf{x}_0$ is the least adversary strength required to alter the target model's prediction, which is also the largest possible robustness certificate for $\mathbf{x}_0$. Unfortunately, finding the minimum distortion of adversarial examples in NNs with Rectified Linear Unit (ReLU) activations (one of the most widely used activation functions) is known to be an NP-complete problem [15, 16], which makes formal verification techniques such as Reluplex [15] suffer from scalability issues and become computationally demanding even for small-sized NNs.

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 works [18, 20], CROWN allows flexible selections of upper and lower bounds for activation functions, enabling us to design an *adaptive* scheme to choose bounds that reduce the approximation error. Our experiments show up to 26% improvements in certified lower bounds.
- 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 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].

Instead of finding the minimum adversarial distortion, a computationally feasible alternative of robustness certificate is to providing a non-trivial and certified lower bound of minimum distortion. Some analytical lower bounds can be derived from the product of operator norms on the weight matrices [3] or the Jacobian matrix in NNs [17]. But these bounds do not take into account the special property of ReLU and can lead to loose bounds [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, Kolter and Wong [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). Raghunathan et al. [19] apply semidefinite programming for robustness certification in ReLU networks but their approach is limited to NNs with one hidden layer. Weng et al. [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 framework, AI2 [28], uses abstract transformations to zonotopes for proving robustness property for ReLU networks. Nonetheless, there are still some application demands using non-ReLU activations, e.g. RNN and LSTM, thus a profound framework towards efficient computation of non-trivial and certified lower bounds for NNs with general activation functions is of great importance. Our proposed method CROWN aims to fill in this gap by generalizing efficient robustness certification to NNs with different activation functions. Additionally, on ReLU networks the flexibility in our framework can achieve a tighter lower bound. Table 1 summarizes the differences of other approaches and CROWN.

Table 1: Comparison of methods for providing adversarial robustness certification in NNs.

Method	Non-trivial bound	Multi-layer	Scalability	Beyond ReLU
Szegedy et. al. [3]	×	✓	✓	✓
Reluplex [15], Planet [25]	✓	✓	×	×
Hein & Andriushchenko [26]	✓	×	✓	differentiable*
Raghunathan et al. [19]	✓	×	×	×
Kolter and Wong [18]	✓	✓	✓	×
Fast-lin / Fast-lip [20]	✓	✓	✓	×
CROWN (ours)	✓	✓	✓	✓ (general)

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

Some recent works (such as robust optimization based adversarial training [29] or region-based classification [30]) *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 computing certified lower bound of minimum adversarial distortion with general activation functions in NNs. CROWN enables fast computation of *certified* lower bounds of large NNs. We provide principles to derive output bounds of NNs when the inputs are perturbed within an ℓ_p ball and each neuron has different linear approximation bounds on its activation function. As shown in our experiments in Sec. 4, the adaptive selection of CROWN on the approximation bounds can achieve a tighter (larger) certified lower bound. 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. 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 $\mathbf{x} \in \mathbb{R}^{n_0}$, let the number of neurons in each layer be $n_k, \forall k \in [m]$, where $[i]$ denotes set $\{1, 2, \cdots, i\}$. Let the k-th layer weight matrix be $\mathbf{W}^{(k)} \in \mathbb{R}^{n_k \times n_{k-1}}$ and bias vector be $\mathbf{b}^{(k)} \in \mathbb{R}^{n_k}$, and let $\Phi_k : \mathbb{R}^{n_0} \to \mathbb{R}^{n_k}$ be the operator mapping from input to layer k. We have $\Phi_k(\mathbf{x}) = \sigma(\mathbf{W}^{(k)}\Phi_{k-1}(\mathbf{x}) + \mathbf{b}^{(k)}), \forall k \in [m-1]$, where $\sigma(\cdot)$ 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: $\sigma(y) = \max(y, 0)$, hyperbolic tangent: $\sigma(y) = \tanh(y)$, sigmoid: $\sigma(y) = 1/(1+e^{-y})$ and $\sigma(y) = \arctan(y)$. Note that the input $\Phi_0(\mathbf{x}) = \mathbf{x}$, and the vector output of the NN is $f(\mathbf{x}) = \Phi_m(\mathbf{x}) = \mathbf{W}^{(m)}\Phi_{m-1}(\mathbf{x}) + \mathbf{b}^{(m)}$. The j-th output element is denoted as $f_j(\mathbf{x}) = [\Phi_m(\mathbf{x})]_j$.

Input perturbation and pre-activation bounds. Let $\mathbf{x_0} \in \mathbb{R}^{n_0}$ be a given data point, and let the perturbed input vector $\mathbf{x}$ be within an ϵ-bounded ℓ_p-ball centered at $\mathbf{x_0}$, i.e., $\mathbf{x} \in \mathbb{B}_p(\mathbf{x_0}, \epsilon)$, where $\mathbb{B}_p(\mathbf{x_0}, \epsilon) := \{\mathbf{x} \mid \|\mathbf{x} - \mathbf{x_0}\|_p \le \epsilon\}$. For the r-th neuron in k-th layer, let its *pre-activation* input be $\mathbf{y}_r^{(k)}$, where $\mathbf{y}_r^{(k)} = \mathbf{W}_{r,:}^{(k)}\Phi_{k-1}(\mathbf{x}) + \mathbf{b}_r^{(k)}$ and $\mathbf{W}_{r,:}^{(k)}$ denotes the r-th row of matrix $\mathbf{W}^{(k)}$. When $\mathbf{x_0}$ is perturbed within a ϵ-bounded ℓ_p-ball, let $\mathbf{l}_r^{(k)}, \mathbf{u}_r^{(k)} \in \mathbb{R}$ be the pre-activation lower bound and upper bound of $\mathbf{y}_r^{(k)}$, i.e. $\mathbf{l}_r^{(k)} \le \mathbf{y}_r^{(k)} \le \mathbf{u}_r^{(k)}$.

Below, we first define the linear upper bounds and lower bounds of activation functions in Definition 3.1, which are the key to derive explicit output bounds for a 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 $\mathbf{l}_r^{(k)}, \mathbf{u}_r^{(k)}$ and the activation function $\sigma(y)$, define two linear functions $h_{U,r}^{(k)}, h_{L,r}^{(k)} : \mathbb{R} \to \mathbb{R}$, $h_{U,r}^{(k)}(y) = \alpha_{U,r}^{(k)}(y + \beta_{U,r}^{(k)})$, $h_{L,r}^{(k)}(y) = \alpha_{L,r}^{(k)}(y + \beta_{L,r}^{(k)})$, such that $h_{L,r}^{(k)}(y) \le \sigma(y) \le h_{U,r}^{(k)}(y)$, $y \in [\mathbf{l}_r^{(k)}, \mathbf{u}_r^{(k)}]$, $\forall k \in [m-1], r \in [n_k]$ and $\alpha_{U,r}^{(k)}, \alpha_{L,r}^{(k)} \in \mathbb{R}^+, \beta_{U,r}^{(k)}, \beta_{L,r}^{(k)} \in \mathbb{R}$.*

Note that the parameters $\alpha_{U,r}^{(k)}, \alpha_{L,r}^{(k)}, \beta_{U,r}^{(k)}, \beta_{L,r}^{(k)}$ depend on $\mathbf{l}_r^{(k)}$ and $\mathbf{u}_r^{(k)}$, i.e. for different $\mathbf{l}_r^{(k)}$ and $\mathbf{u}_r^{(k)}$ 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 $f_j^L : \mathbb{R}^{n_0} \to \mathbb{R}$ and $f_j^U : \mathbb{R}^{n_0} \to \mathbb{R}$ such that $\forall j \in [n_m],\ \forall \mathbf{x} \in \mathbb{B}_p(\mathbf{x_0}, \epsilon)$, the inequality $f_j^L(\mathbf{x}) \leq f_j(\mathbf{x}) \leq f_j^U(\mathbf{x})$ holds true, where*

$$f_j^U(\mathbf{x}) = \mathbf{\Lambda}_{j,:}^{(0)}\mathbf{x} + \sum_{k=1}^{m} \mathbf{\Lambda}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \mathbf{\Delta}_{:,j}^{(k)}), \quad f_j^L(\mathbf{x}) = \mathbf{\Omega}_{j,:}^{(0)}\mathbf{x} + \sum_{k=1}^{m} \mathbf{\Omega}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \mathbf{\Theta}_{:,j}^{(k)}), \quad (1)$$

$$\mathbf{\Lambda}_{j,:}^{(k-1)} = \begin{cases} \mathbf{e}_j^\top & \text{if } k = m+1; \\ (\mathbf{\Lambda}_{j,:}^{(k)}\mathbf{W}^{(k)}) \odot \lambda_{j,:}^{(k-1)} & \text{if } k \in [m]. \end{cases} \quad \mathbf{\Omega}_{j,:}^{(k-1)} = \begin{cases} \mathbf{e}_j^\top & \text{if } k = m+1; \\ (\mathbf{\Omega}_{j,:}^{(k)}\mathbf{W}^{(k)}) \odot \omega_{j,:}^{(k-1)} & \text{if } k \in [m]. \end{cases}$$

and $\forall i \in [n_k]$, we define four matrices $\lambda^{(k)}, \omega^{(k)}, \mathbf{\Delta}^{(k)}, \mathbf{\Theta}^{(k)} \in \mathbb{R}^{n_m \times n_k}$:

$$\lambda_{j,i}^{(k)} = \begin{cases} \alpha_{U,i}^{(k)} & \text{if } k \neq 0,\ \mathbf{\Lambda}_{j,:}^{(k+1)}\mathbf{W}_{:,i}^{(k+1)} \geq 0; \\ \alpha_{L,i}^{(k)} & \text{if } k \neq 0,\ \mathbf{\Lambda}_{j,:}^{(k+1)}\mathbf{W}_{:,i}^{(k+1)} < 0; \\ 1 & \text{if } k = 0. \end{cases} \quad \omega_{j,i}^{(k)} = \begin{cases} \alpha_{L,i}^{(k)} & \text{if } k \neq 0,\ \mathbf{\Omega}_{j,:}^{(k+1)}\mathbf{W}_{:,i}^{(k+1)} \geq 0; \\ \alpha_{U,i}^{(k)} & \text{if } k \neq 0,\ \mathbf{\Omega}_{j,:}^{(k+1)}\mathbf{W}_{:,i}^{(k+1)} < 0; \\ 1 & \text{if } k = 0. \end{cases}$$

$$\mathbf{\Delta}_{i,j}^{(k)} = \begin{cases} \beta_{U,i}^{(k)} & \text{if } k \neq m,\ \mathbf{\Lambda}_{j,:}^{(k+1)}\mathbf{W}_{:,i}^{(k+1)} \geq 0; \\ \beta_{L,i}^{(k)} & \text{if } k \neq m,\ \mathbf{\Lambda}_{j,:}^{(k+1)}\mathbf{W}_{:,i}^{(k+1)} < 0; \\ 0 & \text{if } k = m. \end{cases} \quad \mathbf{\Theta}_{i,j}^{(k)} = \begin{cases} \beta_{L,i}^{(k)} & \text{if } k \neq m,\ \mathbf{\Omega}_{j,:}^{(k+1)}\mathbf{W}_{:,i}^{(k+1)} \geq 0; \\ \beta_{U,i}^{(k)} & \text{if } k \neq m,\ \mathbf{\Omega}_{j,:}^{(k+1)}\mathbf{W}_{:,i}^{(k+1)} < 0; \\ 0 & \text{if } k = m. \end{cases}$$

and $\odot$ is the Hadamard product and $\mathbf{e}_j \in \mathbb{R}^{n_{m-1}}$ is a vector where all elements are 1.

Theorem 3.2 illustrates how a NN function $f_j(\mathbf{x})$ can be bounded by two linear functions $f_j^U(\mathbf{x})$ and $f_j^L(\mathbf{x})$ when the activation function of each neuron is bounded by two linear functions $h_{U,r}^{(k)}$ and $h_{L,r}^{(k)}$ in Definition 3.1. The central idea is to unwrap the activation functions layer by layer by considering the signs of the associated weights of each neuron and apply the two linear bounds $h_{U,r}^{(k)}$ and $h_{L,r}^{(k)}$. As we demonstrate in the proof, when we replace the activation functions with the corresponding linear upper bounds and lower bounds at the layer $m-1$, we can then define equivalent weights and biases based on the parameters of $h_{U,r}^{(m-1)}$ and $h_{L,r}^{(m-1)}$ (e.g. $\mathbf{\Lambda}^{(k)}, \mathbf{\Delta}^{(k)}, \mathbf{\Omega}^{(k)}, \mathbf{\Theta}^{(k)}$ are related to the terms $\alpha_{U,r}^{(k)}, \beta_{U,r}^{(k)}, \alpha_{L,r}^{(k)}, \beta_{L,r}^{(k)}$, respectively) and then repeat the procedure to "back-propagate" to the input layer. This allows us to obtain $f_j^U(\mathbf{x})$ and $f_j^L(\mathbf{x})$ in (1). The formal proof of Theorem 3.2 is in Appendix A. Note that for a neuron r in layer k the slopes of its linear upper and lower bounds $\alpha_{U,r}^{(k)}, \alpha_{L,r}^{(k)}$ of $h_{U,r}^{(k)}$ and $h_{L,r}^{(k)}$ can be different. This implies:

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

Uniform bounds. More importantly, since the input $\mathbf{x} \in \mathbb{B}_p(\mathbf{x_0}, \epsilon)$, we can take the maximum, i.e. $\max_{\mathbf{x} \in \mathbb{B}_p(\mathbf{x_0}, \epsilon)} f_j^U(\mathbf{x})$, and minimum, i.e. $\min_{\mathbf{x} \in \mathbb{B}_p(\mathbf{x_0}, \epsilon)} f_j^L(\mathbf{x})$, as a pair of uniform upper and lower bound of $f_j(\mathbf{x})$ – which in fact has *closed-form* solutions because $f_j^U(\mathbf{x})$ and $f_j^L(\mathbf{x})$ are two linear functions and $\mathbf{x} \in \mathbb{B}_p(\mathbf{x_0}, \epsilon)$ is a convex norm constraint. This result is formally presented below:

Corollary 3.3 (Closed-form uniform bounds)**.** *Given a data point $\mathbf{x_0} \in \mathbb{R}^{n_0}$, ℓ_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 γ_j^L and γ_j^U such that $\forall \mathbf{x} \in \mathbb{B}_p(\mathbf{x_0}, \epsilon)$ and $\forall j \in [n_m]$, $1/q = 1 - 1/p$, the inequality $\gamma_j^L \leq f_j(\mathbf{x}) \leq \gamma_j^U$ holds true, where*

$$\gamma_j^U = \epsilon\|\mathbf{\Lambda}_{j,:}^{(0)}\|_q + \mathbf{\Lambda}_{j,:}^{(0)}\mathbf{x_0} + \sum_{k=1}^{m} \mathbf{\Lambda}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \mathbf{\Delta}_{:,j}^{(k)}),\ \gamma_j^L = -\epsilon\|\mathbf{\Omega}_{j,:}^{(0)}\|_q + \mathbf{\Omega}_{j,:}^{(0)}\mathbf{x_0} + \sum_{k=1}^{m} \mathbf{\Omega}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \mathbf{\Theta}_{:,j}^{(k)}). \quad (2)$$

It can be proved observing that $\mathbf{x}$ is only in term $\mathbf{\Lambda}_{j,:}^{(0)}\mathbf{x}$ or $\mathbf{\Omega}_{j,:}^{(0)}\mathbf{x}$ in (1) and apply Cauchy-Schwartz.

Table 2: Linear upper bound parameters of various activation functions: $h^{(k)}_{U,r}(y) = \alpha^{(k)}_{U,r}(y + \beta^{(k)}_{U,r})$

Upper bound $h^{(k)}_{U,r}$	$r \in \mathcal{S}^+_k$		$r \in \mathcal{S}^-_k$		$r \in \mathcal{S}^\pm_k$	
for activation function	$\alpha^{(k)}_{U,r}$	$\beta^{(k)}_{U,r}$	$\alpha^{(k)}_{U,r}$	$\beta^{(k)}_{U,r}$	$\alpha^{(k)}_{U,r}$	$\beta^{(k)}_{U,r}$
ReLU	1	0	0	0	a ($a \geq \frac{\mathbf{u}^{(k)}_r}{\mathbf{u}^{(k)}_r - \mathbf{l}^{(k)}_r}$, e.g. $a = \frac{\mathbf{u}^{(k)}_r}{\mathbf{u}^{(k)}_r - \mathbf{l}^{(k)}_r}$)	$-\mathbf{l}^{(k)}_r$
Sigmoid, tanh (denoted as $\sigma(y)$)	$\sigma'(d)$ ($\mathbf{l}^{(k)}_r \leq d \leq \mathbf{u}^{(k)}_r$)	$\frac{\sigma(d)}{\alpha^{(k)}_{U,r}} - d$ *	$\frac{\sigma(\mathbf{u}^{(k)}_r) - \sigma(\mathbf{l}^{(k)}_r)}{\mathbf{u}^{(k)}_r - \mathbf{l}^{(k)}_r}$	$\frac{\sigma(\mathbf{l}^{(k)}_r)}{\alpha^{(k)}_{U,r}} - \mathbf{l}^{(k)}_r$	$\sigma'(d)$ ($\frac{\sigma(d) - \sigma(\mathbf{l}^{(k)}_r)}{d - \mathbf{l}^{(k)}_r} - \sigma'(d) = 0, d \geq 0$) ◇	$\frac{\sigma(\mathbf{l}^{(k)}_r)}{\alpha^{(k)}_{U,r}} - \mathbf{l}^{(k)}_r$

* If $\alpha^{(k)}_{U,r}$ is close to 0, we suggest to calculate the intercept directly $\alpha^{(k)}_{U,r} \cdot \beta^{(k)}_{U,r} = \sigma(d) - \alpha^{(k)}_{U,r} d$ to avoid numerical issues in implementation. Same for other similar cases.

◇ Alternatively, if $d \geq \mathbf{u}^{(k)}_r$, then we can set $\alpha^{(k)}_{U,r} = \frac{\sigma(\mathbf{u}^{(k)}_r) - \sigma(\mathbf{l}^{(k)}_r)}{\mathbf{u}^{(k)}_r - \mathbf{l}^{(k)}_r}$

Table 3: Linear lower bound parameters of various activation functions: $h^{(k)}_{L,r}(y) = \alpha^{(k)}_{L,r}(y + \beta^{(k)}_{L,r})$

Lower bound $h^{(k)}_{L,r}$	$r \in \mathcal{S}^+_k$		$r \in \mathcal{S}^-_k$		$r \in \mathcal{S}^\pm_k$	
for activation function	$\alpha^{(k)}_{L,r}$	$\beta^{(k)}_{L,r}$	$\alpha^{(k)}_{L,r}$	$\beta^{(k)}_{L,r}$	$\alpha^{(k)}_{L,r}$	$\beta^{(k)}_{L,r}$
ReLU	1	0	0	0	a ($0 \leq a \leq 1$, e.g. $a = \frac{\mathbf{u}^{(k)}_r}{\mathbf{u}^{(k)}_r - \mathbf{l}^{(k)}_r}, 0, 1$)	0
Sigmoid, tanh (denoted as $\sigma(y)$)	$\frac{\sigma(\mathbf{u}^{(k)}_r) - \sigma(\mathbf{l}^{(k)}_r)}{\mathbf{u}^{(k)}_r - \mathbf{l}^{(k)}_r}$	$\frac{\sigma(\mathbf{l}^{(k)}_r)}{\alpha^{(k)}_{L,r}} - \mathbf{l}^{(k)}_r$	$\sigma'(d)$ ($\mathbf{l}^{(k)}_r \leq d \leq \mathbf{u}^{(k)}_r$)	$\frac{\sigma(d)}{\alpha^{(k)}_{L,r}} - d$	$\sigma'(d)$ ($\frac{\sigma(d) - \sigma(\mathbf{u}^{(k)}_r)}{d - \mathbf{u}^{(k)}_r} - \sigma'(d) = 0, d \leq 0$) †	$\frac{\sigma(\mathbf{u}^{(k)}_r)}{\alpha^{(k)}_{L,r}} - \mathbf{u}^{(k)}_r$

† Alternatively, if $d \leq \mathbf{l}^{(k)}_r$, then we can set $\alpha^{(k)}_{L,r} = \frac{\sigma(\mathbf{u}^{(k)}_r) - \sigma(\mathbf{l}^{(k)}_r)}{\mathbf{u}^{(k)}_r - \mathbf{l}^{(k)}_r}$

Certified lower bound of minimum distortion. Given an input example $\mathbf{x_0}$ and an m-layer NN, let c be the predicted class of $\mathbf{x_0}$ and $t \neq c$ 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:

$$\tilde{\epsilon}_t = \max_{\epsilon} \epsilon \text{ s.t. } \gamma^L_c(\epsilon) - \gamma^U_t(\epsilon) > 0 \text{ and } \tilde{\epsilon} = \min_{t \neq c} \tilde{\epsilon}_t.$$

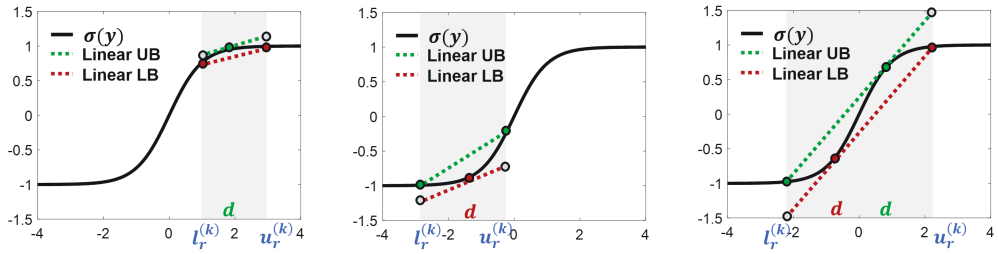
We note that although there is a linear ϵ term in (2), other terms such as $\mathbf{\Lambda}^{(k)}, \mathbf{\Delta}^{(k)}$ and $\mathbf{\Omega}^{(k)}, \mathbf{\Theta}^{(k)}$ also implicitly depend on ϵ. This is because the parameters $\alpha^{(k)}_{U,i}, \beta^{(k)}_{U,i}, \alpha^{(k)}_{L,i}, \beta^{(k)}_{L,i}$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

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

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

[2]The bound can be further improved by considering $g(\mathbf{x}) := f_c(\mathbf{x}) - f_t(\mathbf{x})$ and replacing the last layer's weights by $\mathbf{W}^{(m)}_{c,:} - \mathbf{W}^{(m)}_{t,:}$. This is also used by [20].



(a) $r \in \mathcal{S}_k^+$ (b) $r \in \mathcal{S}_k^-$ (c) $r \in \mathcal{S}_k^\pm$

Figure 1: $\sigma(y) = \tanh$. Green lines are the upper bounds $h_{H,r}^{(k)}$; red lines are the lower bounds $h_{H,r}^{(k)}$

to obtain certified lower bounds of minimum distortion. In this section, we illustrate how to find parameters $\alpha_{U,r}^{(k)}, \alpha_{L,r}^{(k)}$ and $\beta_{U,r}^{(k)}, \beta_{L,r}^{(k)}$ 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 $\mathbf{l}_r^{(k)}$ and $\mathbf{u}_r^{(k)}$, we define a partition $\{\mathcal{S}_k^+, \mathcal{S}_k^\pm, \mathcal{S}_k^-\}$ of set $[n_k]$ such that every neuron in k-th layer belongs to exactly one of the three sets. The formal definition of $\mathcal{S}_k^+$, $\mathcal{S}_k^\pm$ and $\mathcal{S}_k^-$ is $\mathcal{S}_k^+ = \{r \in [n_k] \mid 0 \leq \mathbf{l}_r^{(k)} \leq \mathbf{u}_r^{(k)}\}$, $\mathcal{S}_k^\pm = \{r \in [n_k] \mid \mathbf{l}_r^{(k)} < 0 < \mathbf{u}_r^{(k)}\}$, and $\mathcal{S}_k^- = \{r \in [n_k] \mid \mathbf{l}_r^{(k)} \leq \mathbf{u}_r^{(k)} \leq 0\}$. For neurons in each partitioned set, we define corresponding upper bound $h_{U,r}^{(k)}$ and lower bound $h_{L,r}^{(k)}$ in terms of $\mathbf{l}_r^{(k)}$ and $\mathbf{u}_r^{(k)}$. As we will see shortly, segmenting the activation functions based on $\mathbf{l}_r^{(k)}$ and $\mathbf{u}_r^{(k)}$ 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 $\mathbf{l}_r^{(k)}, \mathbf{u}_r^{(k)}$ 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 $\mathcal{S}_k^+$, $\mathcal{S}_k^\pm$ and $\mathcal{S}_k^-$ 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 $h_{U,r}^{(k)}$ and $h_{L,r}^{(k)}$. Below we call $(\mathbf{l}_r^{(k)}, \sigma(\mathbf{l}_r^{(k)}))$ as left end-point and $(\mathbf{u}_r^{(k)}, \sigma(\mathbf{u}_r^{(k)}))$ as right end-point. For $r \in \mathcal{S}_k^+$, since $\sigma(y)$ is concave, we can let $h_{U,r}^{(k)}$ be any tangent line of $\sigma(y)$ at point $d \in [\mathbf{l}_r^{(k)}, \mathbf{u}_r^{(k)}]$, and let $h_{L,r}^{(k)}$ pass the two end-points. Similarly, $\sigma(y)$ is concave for $r \in \mathcal{S}_k^+$, thus we can let $h_{L,r}^{(k)}$ be any tangent line of $\sigma(y)$ at point $d \in [\mathbf{l}_r^{(k)}, \mathbf{u}_r^{(k)}]$ and let $h_{U,r}^{(k)}$ pass the two end-points. Lastly, for $r \in \mathcal{S}_k^\pm$, we can let $h_{U,r}^{(k)}$ be the tangent line that passes the left end-point and $(d, \sigma(d))$ where $d \geq 0$ and $h_{U,r}^{(k)}$ 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 $r \in \mathcal{S}_k^+$, we have $\sigma(y) = y$ and so we can set $h_{U,r}^{(k)} = h_{L,r}^{(k)} = y$; if $r \in \mathcal{S}_k^-$, we have $\sigma(y) = 0$, and thus we can set $h_{U,r}^{(k)} = h_{L,r}^{(k)} = 0$; if $r \in \mathcal{S}_k^\pm$, we can set $h_{U,r}^{(k)} = \frac{\mathbf{u}_r^{(k)}}{\mathbf{u}_r^{(k)} - \mathbf{l}_r^{(k)}}(y - \mathbf{l}_r^{(k)})$ and $h_{L,r}^{(k)} = ay$, $0 \leq a \leq 1$. Setting $a = \frac{\mathbf{u}_r^{(k)}}{\mathbf{u}_r^{(k)} - \mathbf{l}_r^{(k)}}$ 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 $\mathbf{u}_r^{(k)} \geq |\mathbf{l}_r^{(k)}|$ and $a = 0$ when $\mathbf{u}_r^{(k)} < |\mathbf{l}_r^{(k)}|$. In this way, the area between the lower bound $h_{L,r}^{(k)} = ay$ 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 $h_{L,r}^{(k)}$ based on the value of $\mathbf{u}_r^{(k)}$ and $\mathbf{l}_r^{(k)}$ 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 $h_{U,r}^{(k)}$ and $h_{U,r}^{(k)}$ in Table 2 and 3. In general, as long as $h_{U,r}^{(k)}$ and $h_{U,r}^{(k)}$ are identified for the activation functions, we can use Corollary 3.3 to compute certified lower bound for general activation functions. Note that there remain many other choices of $h_{U,r}^{(k)}$ and $h_{L,r}^{(k)}$ 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 $h_{U,r}^{(k)}$ and $h_{L,r}^{(k)}$: $h_{U,r}^{(k)}(y) = \eta_{U,r}^{(k)} y^2 + \alpha_{U,r}^{(k)}(y+\beta_{U,r}^{(k)})$, $h_{L,r}^{(k)}(y) = \eta_{L,r}^{(k)} y^2 + \alpha_{L,r}^{(k)}(y+\beta_{L,r}^{(k)})$, where $\eta_{U,r}^{(k)}, \eta_{L,r}^{(k)} \in \mathbb{R}$. 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:

$$f_j^U(\mathbf{x}) = \Phi_{m-2}(\mathbf{x})^\top \mathbf{Q}_U^{(m-1)} \Phi_{m-2}(\mathbf{x}) + 2\mathbf{p}_U^{(m-1)} \Phi_{m-2}(\mathbf{x}) + s_U^{(m-1)}, \tag{3}$$

$$f_j^L(\mathbf{x}) = \Phi_{m-2}(\mathbf{x})^\top \mathbf{Q}_L^{(m-1)} \Phi_{m-2}(\mathbf{x}) + 2\mathbf{p}_L^{(m-1)} \Phi_{m-2}(\mathbf{x}) + s_L^{(m-1)}, \tag{4}$$

where $\mathbf{Q}_U^{(m-1)} = \mathbf{W}^{(m-1)\top} \mathbf{D}_U^{(m-1)} \mathbf{W}^{(m-1)}$, $\mathbf{Q}_L^{(m-1)} = \mathbf{W}^{(m-1)\top} \mathbf{D}_L^{(m-1)} \mathbf{W}^{(m-1)}$, $\mathbf{p}_U^{(m-1)}$, $\mathbf{p}_L^{(m-1)}$, $s_U^{(m-1)}$, and $s_L^{(m-1)}$ are defined in Appendix D due to page limit. When $m = 2$, $\Phi_{m-2}(\mathbf{x}) = \mathbf{x}$ and we can directly optimize over $\mathbf{x} \in \mathbb{B}_p(\mathbf{x_0}, \epsilon)$; otherwise, we can use the post activation bounds of layer $m-2$ as the constraints. $\mathbf{D}_U^{(m-1)}$ in (3) is a diagonal matrix with i-th entry being $\mathbf{W}_{j,i}^{(m)} \eta_{U,i}^{(m-1)}$, if $\mathbf{W}_{j,i}^{(m)} \geq 0$ or $\mathbf{W}_{j,i}^{(m)} \eta_{L,i}^{(m-1)}$, if $\mathbf{W}_{j,i}^{(m)} < 0$. Thus, in general $\mathbf{Q}_U^{(m-1)}$ 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 $\max_{\mathbf{x} \in \mathbb{B}_p(\mathbf{x_0}, \epsilon)} f_j^U(\mathbf{x})$ into a convex Quadratic Programming problem; for example, we can let $\eta_{U,i}^{(m-1)} = 0$ for all $\mathbf{W}_{j,i}^{(m)} > 0$ and let $\eta_{L,i}^{(m-1)} > 0$ to make $\mathbf{D}_U^{(m-1)}$ 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 $\mathbf{D}_L^{(m-1)}$, we let $\eta_{U,i}^{(m-1)} = 0$ for all $\mathbf{W}_{j,i}^{(m)} < 0$ and let $\eta_{L,i}^{(m-1)} > 0$ to make $\mathbf{D}_L^{(m-1)}$ have non-negative diagonals and hence the problem $\min_{\mathbf{x} \in \mathbb{B}_p(\mathbf{x_0}, \epsilon)} f_j^L(\mathbf{x})$ is convex. We can solve this convex program with projected gradient descent (PGD) for $\mathbf{x} \in \mathbb{B}_p(\mathbf{x_0}, \epsilon)$ and Armijo line search. Empirically we find that PGD usually converges within a few iterations.

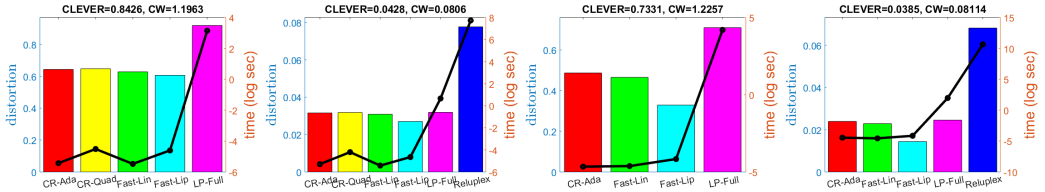
4 Experiments

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 for 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 a Intel Skylake server CPU running at 2.0 GHz on Google Cloud.

Results on Small Networks. Figure 2 shows the certified lower bound for ℓ_2 and ℓ_∞ distortions found by different algorithms on small networks, where Reluplex is feasible and we can observe the



(a) MNIST $2 \times [20]$, ℓ_2 (b) MNIST $2 \times [20]$, ℓ_∞ (c) MNIST $3 \times [20]$, ℓ_2 (d) MNIST $3 \times [20]$, ℓ_∞

Figure 2: Certified lower bounds and min distortion comparisons for ℓ_2 and ℓ_∞ distortions. Left y-axis is distortion and right y-axis (black line) is computation time (seconds, logarithmic scale). On the top of figures are the avg. CLEVER score and the upper bound found by C&W attack [6]. From left to right in (a)-(d): CROWN-Ada, (CROWN-Quad), Fast-Lin, Fast-Lip, LP-Full and (Reluplex).

Table 4: Comparison of certified lower bounds on large ReLU networks. Bounds are the average over 100 images (skipped misclassified images) with random attack targets. Percentage improvements are calculated against Fast-Lin as Fast-Lip is worse than Fast-Lin.

Network	Certified Bounds				Improvement (%)	Average Computation Time (sec)		
	ℓ_p norm	Fast-Lin	Fast-Lip	CROWN-Ada	CROWN-Ada vs Fast-Lin	Fast-Lin	Fast-Lip	CROWN-Ada
MNIST $4 \times [1024]$	ℓ_1	1.57649	0.72800	**1.88217**	+19%	1.80	2.04	3.54
	ℓ_2	0.18891	0.06487	**0.22811**	+21%	1.78	1.96	3.79
	ℓ_∞	0.00823	0.00264	**0.00997**	+21%	1.53	2.17	3.57
CIFAR-10 $7 \times [1024]$	ℓ_1	0.86468	0.09239	**1.09067**	+26%	13.21	19.76	22.43
	ℓ_2	0.05937	0.00407	**0.07496**	+26%	12.57	18.71	21.82
	ℓ_∞	0.00134	0.00008	**0.00169**	+26%	8.98	20.34	16.66

Table 5: Comparison of certified lower bounds by CROWN-Ada on ReLU networks and CROWN-general on networks with tanh, sigmoid and arctan activations. CIFAR models with sigmoid activations achieve much worse accuracy than other networks and are thus excluded.

Network	Certified Bounds by CROWN-Ada and CROWN-general					Average Computation Time (sec)			
	ℓ_p norm	ReLU	tanh	sigmoid	arctan	ReLU	tanh	sigmoid	arctan
MNIST $3 \times [1024]$	ℓ_1	3.00231	2.48407	2.94239	2.33246	1.25	1.61	1.68	1.70
	ℓ_2	0.50841	0.27287	0.44471	0.30345	1.26	1.76	1.61	1.75
	ℓ_∞	0.02576	0.01182	0.02122	0.01363	1.37	1.78	1.76	1.77
CIFAR-10 $6 \times [2048]$	ℓ_1	0.91201	0.44059	-	0.46198	71.62	89.77	-	83.80
	ℓ_2	0.05245	0.02538	-	0.02515	71.51	84.22	-	83.12
	ℓ_∞	0.00114	0.00055	-	0.00055	49.28	59.72	-	58.04

gap between different certified lower bounds and the true minimum adversarial distortion. Reluplex and LP-Full are orders of magnitudes slower than other methods (note the logarithmic scale on right y-axis), and CROWN-Quad (for 2-layer) and CROWN-Ada achieve the largest lower bounds. Improvements of CROWN-Ada over Fast-Lin are more significant in larger NNs, as we show below.

Results on Large ReLU Networks. Table 4 demonstrates the lower bounds found by different algorithms for all common ℓ_p norms. CROWN-Ada significantly outperforms Fast-Lin and Fast-Lip, while the computation time increased by less than 2X over Fast-Lin, and is comparable with Fast-Lip. See Appendix for results on more networks.

Results on Different Activations. Table 7 compares the certified lower bound computed by CROWN-general for four activation functions and different ℓ_p norm on large networks. CROWN-general is able to certify non-trivial lower bounds for all four activation functions efficiently. Comparing to CROWN-Ada on ReLU networks, certifying general activations that are not piece-wise linear only incurs a about 20% computational overhead.

5 Conclusion

We propose a general framework CROWN to efficiently compute a certified lower bound of minimum distortion in neural networks. 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.

References

[1] A. Fawzi, S.-M. Moosavi-Dezfooli, and P. Frossard, "The robustness of deep networks: A geometrical perspective," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 50–62, 2017.

[2] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *arXiv preprint arXiv:1712.03141*, 2017.

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

[4] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *ICLR, arXiv preprint arXiv:1412.6572*, 2015.

[5] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2574–2582.

[6] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 39–57.

[7] H. Chen, H. Zhang, P.-Y. Chen, J. Yi, and C.-J. Hsieh, "Show-and-fool: Crafting adversarial examples for neural image captioning," *arXiv preprint arXiv:1712.02051*, 2017.

[8] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *ACM Asia Conference on Computer and Communications Security*, 2017, pp. 506–519.

[9] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," *ICLR, arXiv preprint arXiv:1611.02770*, 2017.

[10] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *ACM Workshop on Artificial Intelligence and Security*, 2017, pp. 15–26.

[11] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," *ICLR, arXiv preprint arXiv:1712.04248*, 2018.

[12] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[13] I. Evtimov, K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, A. Rahmati, and D. Song, "Robust physical-world attacks on machine learning models," *arXiv preprint arXiv:1707.08945*, 2017.

[14] A. Athalye and I. Sutskever, "Synthesizing robust adversarial examples," *arXiv preprint arXiv:1707.07397*, 2017.

[15] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," in *International Conference on Computer Aided Verification*. Springer, 2017, pp. 97–117.

[16] A. Sinha, H. Namkoong, and J. Duchi, "Certifiable distributional robustness with principled adversarial training," *ICLR, arXiv preprint arXiv:1710.10571*, 2018.

[17] J. Peck, J. Roels, B. Goossens, and Y. Saeys, "Lower bounds on the robustness to adversarial perturbations," in *Advances in Neural Information Processing Systems*, 2017, pp. 804–813.

[18] J. Z. Kolter and E. Wong, "Provable defenses against adversarial examples via the convex outer adversarial polytope," *arXiv preprint arXiv:1711.00851*, 2017.

[19] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," *ICLR, arXiv preprint arXiv:1801.09344*, 2018.

[20] T.-W. Weng, H. Zhang, H. Chen, Z. Song, C.-J. Hsieh, D. Boning, I. S. Dhillon, and L. Daniel, "Towards fast computation of certified robustness for relu networks," *ICML, arXiv preprint arXiv:1804.09699*, 2018.

[21] A. Lomuscio and L. Maganti, "An approach to reachability analysis for feed-forward relu neural networks," *arXiv preprint arXiv:1706.07351*, 2017.

[22] C.-H. Cheng, G. Nührenberg, and H. Ruess, "Maximum resilience of artificial neural networks," in *International Symposium on Automated Technology for Verification and Analysis*. Springer, 2017, pp. 251–268.

[23] M. Fischetti and J. Jo, "Deep neural networks as 0-1 mixed integer linear programs: A feasibility study," *arXiv preprint arXiv:1712.06174*, 2017.

[24] N. Carlini, G. Katz, C. Barrett, and D. L. Dill, "Provably minimally-distorted adversarial examples," *arXiv preprint arXiv:1709.10207*, 2017.

[25] R. Ehlers, "Formal verification of piece-wise linear feed-forward neural networks," in *International Symposium on Automated Technology for Verification and Analysis*. Springer, 2017, pp. 269–286.

[26] M. Hein and M. Andriushchenko, "Formal guarantees on the robustness of a classifier against adversarial manipulation," in *Advances in Neural Information Processing Systems*, 2017, pp. 2263–2273.

[27] T.-W. Weng, H. Zhang, P.-Y. Chen, J. Yi, D. Su, Y. Gao, C.-J. Hsieh, and L. Daniel, "Evaluating the robustness of neural networks: An extreme value theory approach," *ICLR, arXiv preprint arXiv:1801.10578*, 2018.

[28] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev, "Ai2: Safety and robustness certification of neural networks with abstract interpretation," in *IEEE Symposium on Security and Privacy (SP)*, vol. 00, 2018, pp. 948–963.

[29] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *ICLR, arXiv preprint arXiv:1706.06083*, 2018.

[30] X. Cao and N. Z. Gong, "Mitigating evasion attacks to deep neural networks via region-based classification," in *ACM Annual Computer Security Applications Conference*, 2017, pp. 278–287.

[31] P.-Y. Chen, Y. Sharma, H. Zhang, J. Yi, and C.-J. Hsieh, "Ead: elastic-net attacks to deep neural networks via adversarial examples," *arXiv preprint arXiv:1709.04114*, 2017.

A Proof of Theorem 3.2

Given an m-layer neural network function $f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_m}$ with pre-activation bounds $\mathbf{l}^{(k)}$ and $\mathbf{u}^{(k)}$ for $\mathbf{x} \in \mathbb{B}_p(\mathbf{x_0}, \epsilon)$ and $\forall k \in [m-1]$, let the pre-activation inputs for the i-th neuron at layer $m-1$ be $\mathbf{y}_i^{(m-1)} := \mathbf{W}_{i,:}^{(m-1)} \Phi_{m-2}(\mathbf{x}) + \mathbf{b}_i^{(m-1)}$. The j-th output of the neural network is the following:

$$
\begin{aligned}
f_j(\mathbf{x}) &= \sum_{i=1}^{n_{m-1}} \mathbf{W}_{j,i}^{(m)} [\Phi_{m-1}(\mathbf{x})]_i + \mathbf{b}_j^{(m)}, && (5)\\
&= \sum_{i=1}^{n_{m-1}} \mathbf{W}_{j,i}^{(m)} \sigma(\mathbf{y}_i^{(m-1)}) + \mathbf{b}_j^{(m)}, \\
&= \underbrace{\sum_{\mathbf{W}_{j,i}^{(m)} \geq 0} \mathbf{W}_{j,i}^{(m)} \sigma(\mathbf{y}_i^{(m-1)})}_{S_1} + \underbrace{\sum_{\mathbf{W}_{j,i}^{(m)} < 0} \mathbf{W}_{j,i}^{(m)} \sigma(\mathbf{y}_i^{(m-1)})}_{S_2} + \mathbf{b}_j^{(m)}. && (6)
\end{aligned}
$$

Assume the activation function $\sigma(y)$ is bounded by two linear functions $h_{U,i}^{(m-1)}, h_{L,i}^{(m-1)}$ in Definition 3.1, we have

$$
h_{L,i}^{(m-1)}(\mathbf{y}_i^{(m-1)}) \leq \sigma(\mathbf{y}_i^{(m-1)}) \leq h_{U,i}^{(m-1)}(\mathbf{y}_i^{(m-1)}).
$$

Thus, if the associated weight $\mathbf{W}_{j,i}^{(m)}$ to the i-th neuron is non-negative (terms in S_1), we have

$$
\mathbf{W}_{j,i}^{(m)} \cdot h_{L,i}^{(m-1)}(\mathbf{y}_i^{(m-1)}) \leq \mathbf{W}_{j,i}^{(m)} \sigma(\mathbf{y}_i^{(m-1)}) \leq \mathbf{W}_{j,i}^{(m)} \cdot h_{U,i}^{(m-1)}(\mathbf{y}_i^{(m-1)}); \tag{7}
$$

otherwise for terms in S_2, we have

$$
\mathbf{W}_{j,i}^{(m)} \cdot h_{U,i}^{(m-1)}(\mathbf{y}_i^{(m-1)}) \leq \mathbf{W}_{j,i}^{(m)} \sigma(\mathbf{y}_i^{(m-1)}) \leq \mathbf{W}_{j,i}^{(m)} \cdot h_{L,i}^{(m-1)}(\mathbf{y}_i^{(m-1)}). \tag{8}
$$

Upper bound. Let $f_j^{U,m-1}(\mathbf{x})$ be an upper bound of $f_j(\mathbf{x})$. To compute $f_j^{U,m-1}(\mathbf{x})$, (6), (7) and (8) are the key equations. Precisely, for the $\mathbf{W}_{j,i}^{(m)} \geq 0$ terms in (6), the upper bound is the right-hand-side (RHS) in (7); and for the $\mathbf{W}_{j,i}^{(m)} < 0$ terms in (6), the upper bound is the RHS in (8). Thus, we obtain:

$$
\begin{aligned}
& f_j^{U,m-1}(\mathbf{x}) \\
&= \sum_{\mathbf{W}_{j,i}^{(m)} \geq 0} \mathbf{W}_{j,i}^{(m)} \cdot h_{U,i}^{(m-1)}(\mathbf{y}_i^{(m-1)}) + \sum_{\mathbf{W}_{j,i}^{(m)} < 0} \mathbf{W}_{j,i}^{(m)} \cdot h_{L,i}^{(m-1)}(\mathbf{y}_i^{(m-1)}) + \mathbf{b}_j^{(m)}, && (9)\\
&= \sum_{\mathbf{W}_{j,i}^{(m)} \geq 0} \mathbf{W}_{j,i}^{(m)} \alpha_{U,i}^{(m-1)}(\mathbf{y}_i^{(m-1)} + \beta_{U,i}^{(m-1)}) + \sum_{\mathbf{W}_{j,i}^{(m)} < 0} \mathbf{W}_{j,i}^{(m)} \alpha_{L,i}^{(m-1)}(\mathbf{y}_i^{(m-1)} + \beta_{L,i}^{(m-1)}) + \mathbf{b}_j^{(m)}, && (10)\\
&= \sum_{i=1}^{n_{m-1}} \mathbf{W}_{j,i}^{(m)} \lambda_{j,i}^{(m-1)}(\mathbf{y}_i^{(m-1)} + \mathbf{\Delta}_{i,j}^{(m-1)}) + \mathbf{b}_j^{(m)}, && (11)\\
&= \sum_{i=1}^{n_{m-1}} \mathbf{\Lambda}_{j,i}^{(m-1)}(\sum_{r=1}^{n_{m-2}} \mathbf{W}_{i,r}^{(m-1)} [\Phi_{m-2}(\mathbf{x})]_r + \mathbf{b}_i^{(m-1)} + \mathbf{\Delta}_{i,j}^{(m-1)}) + \mathbf{b}_j^{(m)}, && (12)\\
&= \sum_{i=1}^{n_{m-1}} \mathbf{\Lambda}_{j,i}^{(m-1)}(\sum_{r=1}^{n_{m-2}} \mathbf{W}_{i,r}^{(m-1)} [\Phi_{m-2}(\mathbf{x})]_r) + \sum_{i=1}^{n_{m-1}} \mathbf{\Lambda}_{j,i}^{(m-1)}(\mathbf{b}_i^{(m-1)} + \mathbf{\Delta}_{i,j}^{(m-1)}) + \mathbf{b}_j^{(m)}, && (13)\\
&= \sum_{r=1}^{n_{m-2}} \left(\sum_{i=1}^{n_{m-1}} \mathbf{\Lambda}_{j,i}^{(m-1)} \mathbf{W}_{i,r}^{(m-1)} \right) [\Phi_{m-2}(\mathbf{x})]_r + \left(\sum_{i=1}^{n_{m-1}} \mathbf{\Lambda}_{j,i}^{(m-1)}(\mathbf{b}_i^{(m-1)} + \mathbf{\Delta}_{i,j}^{(m-1)}) + \mathbf{b}_j^{(m)} \right), && (14)\\
&= \sum_{r=1}^{n_{m-2}} \tilde{\mathbf{W}}_{j,r}^{(m-1)} [\Phi_{m-2}(\mathbf{x})]_r + \tilde{\mathbf{b}}_j^{(m-1)}. && (15)
\end{aligned}
$$

From (9) to (10), we replace $h_{U,i}^{(m-1)}(\mathbf{y}_i^{(m-1)})$ and $h_{L,i}^{(m-1)}(\mathbf{y}_i^{(m-1)})$ by their definitions; from (10) to (11), we use variables $\lambda_{j,i}^{(m-1)}$ and $\mathbf{\Delta}_{j,i}^{(m-1)}$ to denote the slopes in front of $\mathbf{y}_i^{(m-1)}$ and the intercepts in the parentheses:

$$\lambda_{j,i}^{(m-1)} = \begin{cases} \alpha_{U,i}^{(m-1)} & \text{if } \mathbf{W}_{j,i}^{(m)} \geq 0 \quad (\iff \mathbf{\Lambda}_{j,:}^{(m)}\mathbf{W}_{:,i}^{(m)} \geq 0); \\ \alpha_{L,i}^{(m-1)} & \text{if } \mathbf{W}_{j,i}^{(m)} < 0 \quad (\iff \mathbf{\Lambda}_{j,:}^{(m)}\mathbf{W}_{:,i}^{(m)} < 0); \end{cases} \tag{16}$$

$$\mathbf{\Delta}_{i,j}^{(m-1)} = \begin{cases} \beta_{U,i}^{(m-1)} & \text{if } \mathbf{W}_{j,i}^{(m)} \geq 0 \quad (\iff \mathbf{\Lambda}_{j,:}^{(m)}\mathbf{W}_{:,i}^{(m)} \geq 0); \\ \beta_{L,i}^{(m-1)} & \text{if } \mathbf{W}_{j,i}^{(m)} < 0 \quad (\iff \mathbf{\Lambda}_{j,:}^{(m)}\mathbf{W}_{:,i}^{(m)} < 0). \end{cases} \tag{17}$$

From (11) to (12), we replace $\mathbf{y}_i^{(m-1)}$ with its definition and let $\mathbf{\Lambda}_{j,i}^{(m)} \coloneqq 1$. From (12) to (13), we collect the constant terms that are not related to $\mathbf{x}$. From (13) to (14), we swap the summation order of i and r, and the coefficients in front of $[\Phi_{m-2}(x)]_r$ can be combined into a new equivalent weight $\tilde{\mathbf{W}}_{j,r}^{(m-1)}$ and the constant term can combined into a new equivalent bias $\tilde{\mathbf{b}}_j^{(m-1)}$ in (15):

$$\tilde{\mathbf{W}}_{j,r}^{(m-1)} = \sum_{i=1}^{n_{m-1}} \mathbf{\Lambda}_{j,i}^{(m-1)}\mathbf{W}_{i,r}^{(m-1)} = \mathbf{\Lambda}_{j,:}^{(m-1)}\mathbf{W}_{:,r}^{(m-1)},$$

$$\tilde{\mathbf{b}}_j^{(m-1)} = \sum_{i=1}^{n_{m-1}} \mathbf{\Lambda}_{j,i}^{(m-1)}(\mathbf{b}_i^{(m-1)} + \mathbf{\Delta}_{i,j}^{(m-1)}) + \mathbf{b}_j^{(m)} = \mathbf{\Lambda}_{j,:}^{(m-1)}(\mathbf{b}^{(m-1)} + \mathbf{\Delta}_{:,j}^{(m-1)}) + \mathbf{b}_j^{(m)}.$$

Notice that after defining the new equivalent weight $\tilde{\mathbf{W}}_{j,r}^{(m-1)}$ and equivalent bias $\tilde{\mathbf{b}}_j^{(m-1)}$, $f_j^{U,m-1}(\mathbf{x})$ in (15) and $f_j(\mathbf{x})$ in (5) are in the same form. Thus, we can repeat the above procedure again to obtain an upper bound of $f_j^{U,m-1}(\mathbf{x})$, i.e. $f_j^{U,m-2}(\mathbf{x})$:

$$\begin{aligned}
\mathbf{\Lambda}_{j,i}^{(m-2)} &= \tilde{\mathbf{W}}_{j,i}^{(m-1)}\lambda_{j,i}^{(m-2)} \\
&= \mathbf{\Lambda}_{j,:}^{(m-1)}\mathbf{W}_{:,i}^{(m-1)}\lambda_{j,i}^{(m-2)} \\
\tilde{\mathbf{W}}_{j,r}^{(m-2)} &= \mathbf{\Lambda}_{j,:}^{(m-2)}\mathbf{W}_{:,r}^{(m-2)} \\
\tilde{\mathbf{b}}_j^{(m-2)} &= \mathbf{\Lambda}_{j,:}^{(m-2)}(\mathbf{b}^{(m-2)} + \mathbf{\Delta}_{:,j}^{(m-2)}) + \tilde{\mathbf{b}}_j^{(m-1)} \\
\lambda_{j,i}^{(m-2)} &= \begin{cases} \alpha_{U,i}^{(m-2)} & \text{if } \tilde{\mathbf{W}}_{j,i}^{(m-1)} \geq 0 \quad (\iff \mathbf{\Lambda}_{j,:}^{(m-1)}\mathbf{W}_{:,i}^{(m-1)} \geq 0); \\ \alpha_{L,i}^{(m-2)} & \text{if } \tilde{\mathbf{W}}_{j,i}^{(m-1)} < 0 \quad (\iff \mathbf{\Lambda}_{j,:}^{(m-1)}\mathbf{W}_{:,i}^{(m-1)} < 0); \end{cases} \\
\mathbf{\Delta}_{i,j}^{(m-2)} &= \begin{cases} \beta_{U,i}^{(m-2)} & \text{if } \tilde{\mathbf{W}}_{j,i}^{(m-1)} \geq 0 \quad (\iff \mathbf{\Lambda}_{j,:}^{(m-1)}\mathbf{W}_{:,i}^{(m-1)} \geq 0); \\ \beta_{L,i}^{(m-2)} & \text{if } \tilde{\mathbf{W}}_{j,i}^{(m-1)} < 0 \quad (\iff \mathbf{\Lambda}_{j,:}^{(m-1)}\mathbf{W}_{:,i}^{(m-1)} < 0). \end{cases}
\end{aligned}$$

and repeat again iteratively until obtain the final upper bound $f_j^{U,1}(\mathbf{x})$, where $f_j(\mathbf{x}) \leq f_j^{U,m-1}(\mathbf{x}) \leq f_j^{U,m-2}(\mathbf{x}) \leq \ldots \leq f_j^{U,1}(\mathbf{x})$. We let $f_j(\mathbf{x})$ denote the final upper bound $f_j^{U,1}(\mathbf{x})$, and we have

$$f_j^U(\mathbf{x}) = \mathbf{\Lambda}_{j,:}^{(0)}\mathbf{x} + \sum_{k=1}^{m} \mathbf{\Lambda}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \mathbf{\Delta}_{:,j}^{(k)})$$

and ($\odot$ is the Hadamard product)

$$\mathbf{\Lambda}_{j,:}^{(k-1)} = \begin{cases} \mathbf{e}_j^\top & \text{if } k = m+1; \\ (\mathbf{\Lambda}_{j,:}^{(k)}\mathbf{W}^{(k)}) \odot \lambda_{j,:}^{(k-1)} & \text{if } k \in [m]. \end{cases}$$

and $\forall i \in [n_k]$,

$$\lambda_{j,i}^{(k)} = \begin{cases} \alpha_{U,i}^{(k)} & \text{if } k \in [m-1],\ \mathbf{\Lambda}_{j,:}^{(k+1)}\mathbf{W}_{:,i}^{(k+1)} \geq 0; \\ \alpha_{L,i}^{(k)} & \text{if } k \in [m-1],\ \mathbf{\Lambda}_{j,:}^{(k+1)}\mathbf{W}_{:,i}^{(k+1)} < 0; \\ 1 & \text{if } k = 0. \end{cases}$$

$$\mathbf{\Delta}_{i,j}^{(k)} = \begin{cases} \beta_{U,i}^{(k)} & \text{if } k \in [m-1],\ \mathbf{\Lambda}_{j,:}^{(k+1)}\mathbf{W}_{:,i}^{(k+1)} \geq 0; \\ \beta_{L,i}^{(k)} & \text{if } k \in [m-1],\ \mathbf{\Lambda}_{j,:}^{(k+1)}\mathbf{W}_{:,i}^{(k+1)} < 0; \\ 0 & \text{if } k = m. \end{cases}$$

Lower bound. The above derivations of upper bound can be applied similarly to deriving lower bounds of $f_j(\mathbf{x})$, and the only difference is now we need to use the LHS of (7) and (8) (rather than RHS when deriving upper bound) to bound the two terms in (6). Thus, following the same procedure in deriving the upper bounds, we can iteratively unwrap the activation functions and obtain a final lower bound $f_j^{L,1}(\mathbf{x})$, where $f_j(\mathbf{x}) \geq f_j^{L,m-1}(\mathbf{x}) \geq f_j^{L,m-2}(\mathbf{x}) \geq \ldots \geq f_j^{L,1}(\mathbf{x})$. Let $f_j^L(\mathbf{x}) = f_j^{L,1}(\mathbf{x})$, we have:

$$f_j^L(\mathbf{x}) = \mathbf{\Omega}_{j,:}^{(0)}\mathbf{x} + \sum_{k=1}^{m} \mathbf{\Omega}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \mathbf{\Theta}_{:,j}^{(k)})$$

$$\mathbf{\Omega}_{j,:}^{(k-1)} = \begin{cases} \mathbf{e}_j^\top & \text{if } k = m+1; \\ (\mathbf{\Omega}_{j,:}^{(k)}\mathbf{W}^{(k)}) \odot \omega_{j,:}^{(k-1)} & \text{if } k \in [m]. \end{cases}$$

and $\forall i \in [n_k]$,

$$\omega_{j,i}^{(k)} = \begin{cases} \alpha_{L,i}^{(k)} & \text{if } k \in [m-1],\ \mathbf{\Omega}_{j,:}^{(k+1)}\mathbf{W}_{:,i}^{(k+1)} \geq 0; \\ \alpha_{U,i}^{(k)} & \text{if } k \in [m-1],\ \mathbf{\Omega}_{j,:}^{(k+1)}\mathbf{W}_{:,i}^{(k+1)} < 0; \\ 1 & \text{if } k = 0. \end{cases}$$

$$\mathbf{\Theta}_{i,j}^{(k)} = \begin{cases} \beta_{L,i}^{(k)} & \text{if } k \in [m-1],\ \mathbf{\Omega}_{j,:}^{(k+1)}\mathbf{W}_{:,i}^{(k+1)} \geq 0; \\ \beta_{U,i}^{(k)} & \text{if } k \in [m-1],\ \mathbf{\Omega}_{j,:}^{(k+1)}\mathbf{W}_{:,i}^{(k+1)} < 0; \\ 0 & \text{if } k = m. \end{cases}$$

Indeed, $\lambda_{j,i}^{(k)}$ and $\omega_{j,i}^{(k)}$ only differs in the conditions of selecting $\alpha_{U,i}^{(k)}$ or $\alpha_{L,i}^{(k)}$; similarly for $\mathbf{\Delta}_{i,j}^{(k)}$ and $\mathbf{\Theta}_{i,j}^{(k)}$.

B Proof of Corollary 3.3

Definition B.1 (Dual norm)**.** *Let $\|\cdot\|$ be a norm on $\mathbb{R}^n$. The associated dual norm, denoted as $\|\cdot\|_*$, is defined as*

$$\|\mathbf{a}\|_* = \{\sup_{\mathbf{y}} \mathbf{a}^\top\mathbf{y} \mid \|\mathbf{y}\| \leq 1\}.$$

Global upper bound. Our goal is to find a *global* upper and lower bound for the m-th layer network output $f_j(\mathbf{x}), \forall \mathbf{x} \in \mathbb{B}_p(\mathbf{x_0}, \epsilon)$. By Theorem 3.2, for $\mathbf{x} \in \mathbb{B}_p(\mathbf{x_0}, \epsilon)$, we have $f_j^L(\mathbf{x}) \leq f_j(\mathbf{x}) \leq f_j^U(\mathbf{x})$ and $f_j^U(\mathbf{x}) = \mathbf{\Lambda}_{j,:}^{(0)}\mathbf{x} + \sum_{k=1}^m \mathbf{\Lambda}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \mathbf{\Delta}_{:,j}^{(k)})$. Thus define $\gamma_j^U := \max_{\mathbf{x}\in\mathbb{B}_p(\mathbf{x_0},\epsilon)} f_j^U(\mathbf{x})$, and we have

$$f_j(\mathbf{x}) \leq f_j^U(\mathbf{x}) \leq \max_{\mathbf{x}\in\mathbb{B}_p(\mathbf{x_0},\epsilon)} f_j^U(\mathbf{x}) = \gamma_j^U,$$

since $\forall \mathbf{x} \in \mathbb{B}_p(\mathbf{x_0}, \epsilon)$. In particular,

$$\begin{aligned}
\max_{\mathbf{x}\in\mathbb{B}_p(\mathbf{x_0},\epsilon)} f_j^U(\mathbf{x}) &= \max_{\mathbf{x}\in\mathbb{B}_p(\mathbf{x_0},\epsilon)} \left[\mathbf{\Lambda}_{j,:}^{(0)}\mathbf{x} + \sum_{k=1}^m \mathbf{\Lambda}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \mathbf{\Delta}_{:,j}^{(k)})\right] \\
&= \left[\max_{\mathbf{x}\in\mathbb{B}_p(\mathbf{x_0},\epsilon)} \mathbf{\Lambda}_{j,:}^{(0)}\mathbf{x}\right] + \sum_{k=1}^m \mathbf{\Lambda}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \mathbf{\Delta}_{:,j}^{(k)}) \qquad (18) \\
&= \epsilon\left[\max_{\mathbf{y}\in\mathbb{B}_p(\mathbf{0},1)} \mathbf{\Lambda}_{j,:}^{(0)}\mathbf{y}\right] + \mathbf{\Lambda}_{j,:}^{(0)}\mathbf{x_0} + \sum_{k=1}^m \mathbf{\Lambda}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \mathbf{\Delta}_{:,j}^{(k)}) \qquad (19) \\
&= \epsilon\|\mathbf{\Lambda}_{j,:}^{(0)}\|_q + \mathbf{\Lambda}_{j,:}^{(0)}\mathbf{x_0} + \sum_{k=1}^m \mathbf{\Lambda}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \mathbf{\Delta}_{:,j}^{(k)}). \qquad (20)
\end{aligned}$$

From (18) to (19), let $\mathbf{y} := \frac{\mathbf{x}-\mathbf{x_0}}{\epsilon}$, and thus $\mathbf{y} \in \mathbb{B}_p(\mathbf{0}, 1)$. From (19) to (20), apply Definition B.1 and use the fact that ℓ_q norm is dual of ℓ_p norm for $p, q \in [1, \infty]$.

Global lower bound. Similarly, let $\gamma_j^L := \min_{\mathbf{x}\in\mathbb{B}_p(\mathbf{x_0},\epsilon)} f_j^L(\mathbf{x})$, we have

$$f_j(\mathbf{x}) \geq f_j^L(\mathbf{x}) \geq \min_{\mathbf{x}\in\mathbb{B}_p(\mathbf{x_0},\epsilon)} f_j^L(\mathbf{x}) = \gamma_j^L.$$

Since $f_j^L(\mathbf{x}) = \mathbf{\Omega}_{j,:}^{(0)}\mathbf{x} + \sum_{k=1}^m \mathbf{\Omega}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \mathbf{\Theta}_{:,j}^{(k)})$, we can derive γ_j^L (similar to the derivation of γ_j^U) below:

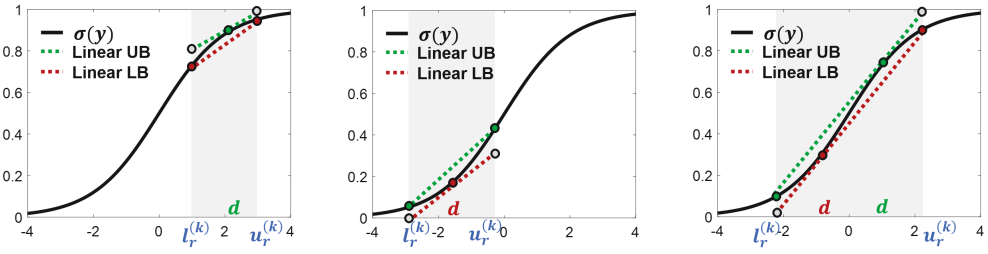
$$\begin{aligned}
\min_{\mathbf{x}\in\mathbb{B}_p(\mathbf{x_0},\epsilon)} f_j^L(\mathbf{x}) &= \min_{\mathbf{x}\in\mathbb{B}_p(\mathbf{x_0},\epsilon)} \left[\mathbf{\Omega}_{j,:}^{(0)}\mathbf{x} + \sum_{k=1}^m \mathbf{\Omega}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \mathbf{\Theta}_{:,j}^{(k)})\right] \\
&= \left[\min_{\mathbf{x}\in\mathbb{B}_p(\mathbf{x_0},\epsilon)} \mathbf{\Omega}_{j,:}^{(0)}\mathbf{x}\right] + \sum_{k=1}^m \mathbf{\Omega}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \mathbf{\Theta}_{:,j}^{(k)}) \\
&= -\epsilon\left[\max_{\mathbf{y}\in\mathbb{B}_p(\mathbf{0},1)} -\mathbf{\Omega}_{j,:}^{(0)}\mathbf{y}\right] + \mathbf{\Omega}_{j,:}^{(0)}\mathbf{x_0} + \sum_{k=1}^m \mathbf{\Omega}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \mathbf{\Theta}_{:,j}^{(k)}) \\
&= -\epsilon\|\mathbf{\Omega}_{j,:}^{(0)}\|_q + \mathbf{\Omega}_{j,:}^{(0)}\mathbf{x_0} + \sum_{k=1}^m \mathbf{\Omega}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \mathbf{\Theta}_{:,j}^{(k)}).
\end{aligned}$$

Thus, we have

$$\text{(global upper bound)} \quad \gamma_j^U = \epsilon\|\mathbf{\Lambda}_{j,:}^{(0)}\|_q + \mathbf{\Lambda}_{j,:}^{(0)}\mathbf{x_0} + \sum_{k=1}^m \mathbf{\Lambda}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \mathbf{\Delta}_{:,j}^{(k)}),$$

$$\text{(global lower bound)} \quad \gamma_j^L = -\epsilon\|\mathbf{\Omega}_{j,:}^{(0)}\|_q + \mathbf{\Omega}_{j,:}^{(0)}\mathbf{x_0} + \sum_{k=1}^m \mathbf{\Omega}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \mathbf{\Theta}_{:,j}^{(k)}),$$

C Illustration of linear upper and lower bounds Sigmoid activation function.



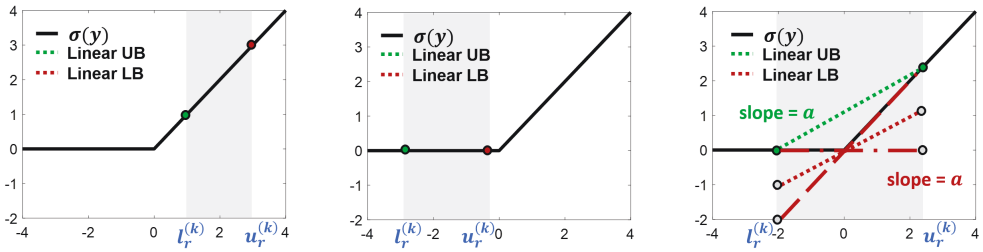
(a) $r \in \mathcal{S}_k^+$ (b) $r \in \mathcal{S}_k^-$ (c) $r \in \mathcal{S}_k^\pm$

Figure 3: The linear upper and lower bounds for $\sigma(y) =$ sigmoid

D $f_j^U(\mathbf{x})$ and $f_j^L(\mathbf{x})$ using Quadratic approximation

Upper bound. Let $f_j^U(\mathbf{x})$ be an upper bound of $f_j(\mathbf{x})$. To compute $f_j^U(\mathbf{x})$ with quadratic approximations, we can still apply (7) and (8) except that $h_{U,r}^{(k)}(y)$ and $h_{L,r}^{(k)}(y)$ are replaced by the following quadratic functions:

$$h_{U,r}^{(k)}(y) = \eta_{U,r}^{(k)}y^2 + \alpha_{U,r}^{(k)}(y + \beta_{U,r}^{(k)}),\ h_{L,r}^{(k)}(y) = \eta_{L,r}^{(k)}y^2 + \alpha_{L,r}^{(k)}(y + \beta_{L,r}^{(k)}).$$



(a) $r \in \mathcal{S}_k^+$ (b) $r \in \mathcal{S}_k^-$ (c) $r \in \mathcal{S}_k^\pm$

Figure 4: The linear upper and lower bounds for $\sigma(y) = \text{ReLU}$. For the cases (a) and (b), the linear upper bound and lower bound are exactly the function $\sigma(y)$ in the region (grey-shaded). For (c), we plot three out of many choices of lower bound, and they are $h_{L,r}^{(k)}(y) = 0$ (dashed-dotted), $h_{L,r}^{(k)}(y) = y$ (dashed), and $h_{L,r}^{(k)}(y) = \frac{\mathbf{u}_r^{(k)}}{\mathbf{u}_r^{(k)} - \mathbf{l}_r^{(k)}} y$ (dotted).

Therefore,

$$f_j^U(\mathbf{x}) = \sum_{\mathbf{W}_{j,i}^{(m)} \geq 0} \mathbf{W}_{j,i}^{(m)} \cdot h_{U,i}^{(m-1)}(\mathbf{y}_i^{(m-1)}) + \sum_{\mathbf{W}_{j,i}^{(m)} < 0} \mathbf{W}_{j,i}^{(m)} \cdot h_{L,i}^{(m-1)}(\mathbf{y}_i^{(m-1)}) + \mathbf{b}_j^{(m)}, \tag{21}$$

$$= \sum_{i=1}^{n_{m-1}} \mathbf{W}_{j,i}^{(m)} \left(\tau_{j,i}^{(m-1)} \mathbf{y}_i^{(m-1)2} + \lambda_{j,i}^{(m-1)} (\mathbf{y}_i^{(m-1)} + \mathbf{\Delta}_{i,j}^{(m-1)}) \right) + \mathbf{b}_j^{(m)}, \tag{22}$$

$$= \mathbf{y}^{(m-1)\top} \text{diag}(\mathbf{q}_{U,j}^{(m-1)}) \mathbf{y}^{(m-1)} + \mathbf{\Lambda}_{j,:}^{(m-1)} \mathbf{y}^{(m-1)} + \mathbf{W}_{j,:}^{(m)} \mathbf{\Delta}_{:,j}^{(m-1)}, \tag{23}$$

$$= \Phi_{m-2}(\mathbf{x})^\top \mathbf{Q}_U^{(m-1)} \Phi_{m-2}(\mathbf{x}) + 2\mathbf{p}_U^{(m-1)} \Phi_{m-2}(\mathbf{x}) + s_U^{(m-1)}. \tag{24}$$

From (21) to (22), we replace $h_{U,i}^{(m-1)}(\mathbf{y}_i^{(m-1)})$ and $h_{L,i}^{(m-1)}(\mathbf{y}_i^{(m-1)})$ by their definitions and let

$$(\tau_{j,i}^{(m-1)}, \lambda_{j,i}^{(m-1)}, \mathbf{\Delta}_{i,j}^{(m-1)}) = \begin{cases} (\eta_{U,i}^{(m-1)}, \alpha_{U,i}^{(m-1)}, \beta_{U,i}^{(m-1)}) & \text{if } \mathbf{W}_{j,i}^{(m)} \geq 0; \\ (\eta_{L,i}^{(m-1)}, \alpha_{L,i}^{(m-1)}, \beta_{L,i}^{(m-1)}) & \text{if } \mathbf{W}_{j,i}^{(m)} < 0. \end{cases}$$

From (22) to (23), we let $\mathbf{q}_{U,j}^{(m-1)} = \mathbf{W}_{j,:}^{(m)} \odot \tau_{j,i}^{(m-1)}$, and write in the matrix form. From (23) to (24), we substitute $\mathbf{y}^{(m-1)}$ by its definition: $\mathbf{y}^{(m-1)} = \mathbf{W}^{(m-1)} \Phi_{(m-2)}(\mathbf{x}) + \mathbf{b}^{(m-1)}$ and then collect the quadratic terms, linear terms and constant terms of $\Phi_{(m-2)}(\mathbf{x})$, where

$$\begin{aligned} \mathbf{Q}_U^{(m-1)} &= \mathbf{W}^{(m-1)\top} \text{diag}(\mathbf{q}_{U,j}^{(m-1)}) \mathbf{W}^{(m-1)}, \\ \mathbf{p}_U^{(m-1)} &= \mathbf{b}^{(m-1)\top} \odot \mathbf{q}_{U,j}^{(m-1)} + \mathbf{\Lambda}_{j,:}^{(m-1)}, \\ s_U^{(m-1)} &= \mathbf{p}_U^{(m-1)} \mathbf{b}^{(m-1)} + \mathbf{W}_{j,:}^{(m)} \mathbf{\Delta}_{:,j}^{(m-1)}. \end{aligned}$$

Lower bound. Similar to the above derivation, we can simply swap $h_{U,r}^{(k)}$ and $h_{L,r}^{(k)}$ and obtain lower bound $f_j^L(\mathbf{x})$:

$$\begin{aligned} f_j^L(\mathbf{x}) &= \sum_{\mathbf{W}_{j,i}^{(m)} < 0} \mathbf{W}_{j,i}^{(m)} \cdot h_{U,i}^{(m-1)}(\mathbf{y}_i^{(m-1)}) + \sum_{\mathbf{W}_{j,i}^{(m)} \geq 0} \mathbf{W}_{j,i}^{(m)} \cdot h_{L,i}^{(m-1)}(\mathbf{y}_i^{(m-1)}) + \mathbf{b}_j^{(m)}, \\ &= \Phi_{m-2}(\mathbf{x})^\top \mathbf{Q}_L^{(m-1)} \Phi_{m-2}(\mathbf{x}) + 2\mathbf{p}_L^{(m-1)} \Phi_{m-2}(\mathbf{x}) + s_L^{(m-1)}, \end{aligned}$$

where

$$\mathbf{Q}_L^{(m-1)} = \mathbf{W}^{(m-1)\top} \text{diag}(\mathbf{q}_{L,j}^{(m-1)}) \mathbf{W}^{(m-1)}, \ \mathbf{q}_{L,j}^{(m-1)} = \mathbf{W}_{j,:}^{(m)} \odot \nu_{j,i}^{(m-1)}; \tag{25}$$

$$\mathbf{p}_U^{(m-1)} = \mathbf{b}^{(m-1)\top} \odot \mathbf{q}_{U,j}^{(m-1)} + \mathbf{\Lambda}_{j,:}^{(m-1)}, \quad \mathbf{p}_L^{(m-1)} = \mathbf{b}^{(m-1)\top} \odot \mathbf{q}_{L,j}^{(m-1)} + \mathbf{\Omega}_{j,:}^{(m-1)}; \tag{26}$$

$$s_U^{(m-1)} = \mathbf{p}_U^{(m-1)} \mathbf{b}^{(m-1)} + \mathbf{W}_{j,:}^{(m)} \mathbf{\Delta}_{:,j}^{(m-1)}, \quad s_L^{(m-1)} = \mathbf{p}_L^{(m-1)} \mathbf{b}^{(m-1)} + \mathbf{W}_{j,:}^{(m)} \mathbf{\Theta}_{:,j}^{(m-1)}, \tag{27}$$

and

$$(\nu_{j,i}^{(m-1)}, \omega_{j,i}^{(m-1)}, \mathbf{\Theta}_{i,j}^{(m-1)}) = \begin{cases} (\eta_{L,i}^{(m-1)}, \alpha_{L,i}^{(m-1)}, \beta_{L,i}^{(m-1)}) & \text{if } \mathbf{W}_{j,i}^{(m)} \geq 0; \\ (\eta_{U,i}^{(m-1)}, \alpha_{U,i}^{(m-1)}, \beta_{U,i}^{(m-1)}) & \text{if } \mathbf{W}_{j,i}^{(m)} < 0. \end{cases} \tag{28}$$

E Additional Experimental Results

E.1 Results on CROWN-Ada

Table 6: Comparison of our proposed certified lower bounds for ReLU with adaptive lower bounds (CROWN-Ada), Fast-Lin and Fast-Lip and Op-nrom. LP-full and Reluplex cannot finish within a reasonable amount of time for all the networks reported here. We also include Op-norm, where we directly compute the operator norm (for example, for $p = 2$ it is the spectral norm) for each layer and use their products as a global Lipschitz constant and then compute the robustness lower bound. CLEVER is an estimated robustness lower bound, and attacking algorithms (including CW [6] and EAD [31]) provide upper bounds of the minimum adversarial distortion. For each norm, we consider the robustness against three targeted attack classes: the runner-up class (with the second largest probability), a random class and the least likely class. It is clear that CROWN-Ada notably improves the lower bound comparing to Fast-Lin, especially for larger and deeper networks, the improvements can be up to 28%.

Networks			Lower bounds and upper bounds (Avg.)							Time per Image (Avg.)		
			Lower Bounds (certified)				improvements	Uncertified		Lower Bounds		
Config	p	Target	[20]		[3]	Our algorithm	over	[27]	Attacks	[20]		Our bound
			Fast-Lin	Fast-Lip	Op norm	CROWN-Ada	Fast-Lin	CLEVER	CW/EAD	Fast-Lin	Fast-Lip	CROWN-Ada
MNIST $2 \times [1024]$	∞	runner-up	0.02256	0.01802	0.00159	0.02467	+9.4%	0.0447	0.0856	163 ms	179 ms	128 ms
		rand	0.03083	0.02512	0.00263	0.03353	+8.8%	0.0708	0.1291	176 ms	213 ms	166 ms
		least	0.03854	0.03128	0.00369	0.04221	+9.5%	0.0925	0.1731	176 ms	251 ms	143 ms
	2	runner-up	0.46034	0.42027	0.24327	0.50110	+8.9%	0.8104	1.1874	154 ms	184 ms	110 ms
		rand	0.63299	0.59033	0.40201	0.68506	+8.2%	1.2841	1.8779	141 ms	212 ms	133 ms
		least	0.79263	0.73133	0.56509	0.86377	+9.0%	1.6716	2.4556	152 ms	291 ms	116 ms
	1	runner-up	2.78786	3.46500	0.20601	3.01633	+8.2%	4.5970	9.5295	159 ms	989 ms	136 ms
		rand	3.88241	5.10000	0.35957	4.17760	+7.6%	7.4186	17.259	168 ms	1.15 s	157 ms
		least	4.90809	6.36600	0.48774	5.33261	+8.6%	9.9847	23.933	179 ms	1.37 s	144 ms
MNIST $3 \times [1024]$	∞	runner-up	0.01830	0.01021	0.00004	0.02114	+15.5%	0.0509	0.1037	805 ms	1.28 s	1.33 s
		rand	0.02216	0.01236	0.00007	0.02576	+16.2%	0.0717	0.1484	782 ms	859 ms	1.37 s
		least	0.02432	0.01384	0.00009	0.02835	+16.6%	0.0825	0.1777	792 ms	684 ms	1.37 s
	2	runner-up	0.35867	0.22120	0.06626	0.41295	+15.1%	0.8402	1.3513	732 ms	1.06 s	1.26 s
		rand	0.43892	0.26980	0.10233	0.50841	+15.8%	1.2441	2.0387	711 ms	696 ms	1.26 s
		least	0.48361	0.30147	0.13256	0.56167	+16.1%	1.4401	2.4916	723 ms	655 ms	1.25 s
	1	runner-up	2.08887	1.80150	0.00734	2.39443	+14.6%	4.8370	10.159	685 ms	2.36 s	1.15 s
		rand	2.59898	2.25950	0.01133	3.00231	+15.5%	7.2177	17.796	743 ms	2.69 s	1.25 s
		least	2.87560	2.50000	0.01499	3.33231	+15.9%	8.3523	22.395	729 ms	3.08 s	1.31 s
MNIST $4 \times [1024]$	∞	runner-up	0.00715	0.00219	0.00001	0.00861	+20.4%	0.0485	0.08635	1.54 s	3.42 s	3.23 s
		rand	0.00823	0.00264	0.00001	0.00997	+21.1%	0.0793	0.1303	1.53 s	2.17 s	3.57 s
		least	0.00899	0.00304	0.00001	0.01096	+21.9%	0.1028	0.1680	1.74 s	2.00 s	3.87 s
	2	runner-up	0.16338	0.05244	0.11015	0.19594	+19.9%	0.8689	1.2422	1.79 s	2.58 s	3.52 s
		rand	0.18891	0.06487	0.17734	0.22811	+20.8%	1.4231	1.8921	1.78 s	1.96 s	3.79 s
		least	0.20671	0.07440	0.23710	0.25119	+21.5%	1.8864	2.4451	1.98 s	2.01 s	4.01 s
	1	runner-up	1.33794	0.58480	0.00114	1.58151	+18.2%	5.2685	10.079	1.87 s	1.93 s	3.34 s
		rand	1.57649	0.72800	0.00183	1.88217	+19.4%	8.9764	17.200	1.80 s	2.04 s	3.54 s
		least	1.73874	0.82800	0.00244	2.09157	+20.3%	11.867	23.910	1.94 s	2.40 s	3.72 s
CIFAR $5 \times [2048]$	∞	runner-up	0.00137	0.00020	0.00000	0.00167	+21.9%	0.0062	0.00950	18.2 s	38.2 s	33.1 s
		rand	0.00170	0.00030	0.00000	0.00212	+24.7%	0.0147	0.02351	19.6 s	48.2 s	36.7 s
		least	0.00188	0.00036	0.00000	0.00236	+25.5%	0.0208	0.03416	20.4 s	50.5 s	38.6 s
	2	runner-up	0.06122	0.00948	0.00156	0.07466	+22.0%	0.2712	0.3778	24.2 s	39.4 s	41.0 s
		rand	0.07654	0.01417	0.00333	0.09527	+24.5%	0.6399	0.9497	26.0 s	31.2 s	42.5 s
		least	0.08456	0.01778	0.00489	0.10588	+25.2%	0.9169	1.4379	25.0 s	33.2 s	44.4 s
	1	runner-up	0.93836	0.22632	0.00000	1.13799	+21.3%	4.0755	7.6529	24.7 s	45.1 s	40.5 s
		rand	1.18928	0.31984	0.00000	1.47393	+23.9%	9.7145	21.643	25.7 s	36.2 s	44.0 s
		least	1.31904	0.38887	0.00001	1.64452	+24.7%	12.793	34.497	26.0 s	31.7 s	44.9 s
CIFAR $6 \times [2048]$	∞	runner-up	0.00075	0.00005	0.00000	0.00094	+25.3%	0.0054	0.00770	27.6 s	64.7 s	47.3 s
		rand	0.00090	0.00007	0.00000	0.00114	+26.7%	0.0131	0.01866	28.1 s	72.3 s	49.3 s
		least	0.00095	0.00008	0.00000	0.00122	+28.4%	0.0199	0.02868	28.1 s	76.3 s	49.4 s
	2	runner-up	0.03462	0.00228	0.00476	0.04314	+24.6%	0.2394	0.2979	37.0 s	60.7 s	65.8 s
		rand	0.04129	0.00331	0.01079	0.05245	+27.0%	0.5860	0.7635	40.0 s	56.8 s	71.5 s
		least	0.04387	0.00385	0.01574	0.05615	+28.0%	0.8756	1.2111	40.0 s	56.3 s	72.5 s
	1	runner-up	0.59636	0.05647	0.00000	0.73727	+23.6%	3.3569	6.0112	37.2 s	65.6 s	66.8 s
		rand	0.72178	0.08212	0.00000	0.91201	+26.4%	8.2507	17.160	39.5 s	53.5 s	71.6 s
		least	0.77179	0.09397	0.00000	0.98331	+27.4%	12.603	28.958	40.7 s	42.1 s	72.5 s
CIFAR $7 \times [1024]$	∞	runner-up	0.00119	0.00006	0.00000	0.00148	+24.4%	0.0062	0.0102	8.98 s	20.1 s	16.2 s
		rand	0.00134	0.00008	0.00000	0.00169	+26.1%	0.0112	0.0218	8.98 s	20.3 s	16.7 s
		least	0.00141	0.00010	0.00000	0.00179	+27.0%	0.0148	0.0333	8.81 s	22.1 s	17.4 s
	2	runner-up	0.05279	0.00308	0.00020	0.06569	+24.4%	0.2661	0.3943	12.7 s	20.9 s	20.7 s
		rand	0.05937	0.00407	0.00029	0.07496	+26.3%	0.5145	0.9730	12.6 s	18.7 s	21.8 s
		least	0.06249	0.00474	0.00038	0.07943	+27.1%	0.6253	1.3709	12.9 s	20.7 s	22.2 s
	1	runner-up	0.76648	0.07028	0.00000	0.95204	+24.2%	4.815	7.9987	12.8 s	21.0 s	21.9 s
		rand	0.86468	0.09239	0.00000	1.09067	+26.1%	8.630	22.180	13.2 s	19.8 s	22.4 s
		least	0.91127	0.10639	0.00000	1.15687	+27.0%	11.44	31.529	13.3 s	17.6 s	22.9 s

E.2 Results on CROWN-general

Table 7: Comparison of certified lower bounds by CROWN-Ada on ReLU networks and CROWN-general on networks with tanh, sigmoid and arctan activations. CIFAR models with sigmoid activations achieve much worse accuracy than other networks and are thus excluded. For each norm, we consider the robustness against three targeted attack classes: the runner-up class (with the second largest probability), a random class and the least likely class.

Network			Certified Bounds by CROWN-general			Average Computation Time (sec)		
	ℓ_p norm	target	tanh	sigmoid	arctan	tanh	sigmoid	arctan
MNIST $3 \times [1024]$	ℓ_∞	runner-up	0.0164	0.0225	0.0169	0.3374	0.3213	0.3148
		random	0.0230	0.0325	0.0240	0.3185	0.3388	0.3128
		least	0.0306	0.0424	0.0314	0.3129	0.3586	0.3156
	ℓ_2	runner-up	0.3546	0.4515	0.3616	0.3139	0.3110	0.3005
		random	0.5023	0.6552	0.5178	0.3044	0.3183	0.2931
		least	0.6696	0.8576	0.6769	0.3869	0.3495	0.2676
	ℓ_1	runner-up	2.4600	2.7953	2.4299	0.2940	0.3339	0.3053
		random	3.5550	4.0854	3.5995	0.3277	0.3306	0.3109
		least	4.8215	5.4528	4.7548	0.3201	0.3915	0.3254
MNIST $4 \times [1024]$	ℓ_∞	runner-up	0.0091	0.0162	0.0107	1.6794	1.7902	1.7099
		random	0.0118	0.0212	0.0136	1.7783	1.7597	1.7667
		least	0.0147	0.0243	0.0165	1.8908	1.8483	1.7930
	ℓ_2	runner-up	0.2086	0.3389	0.2348	1.6416	1.7606	1.8267
		random	0.2729	0.4447	0.3034	1.7589	1.7518	1.6945
		least	0.3399	0.5064	0.3690	1.8206	1.7929	1.8264
	ℓ_1	runner-up	1.8296	2.2397	1.7481	1.5506	1.6052	1.6704
		random	2.4841	2.9424	2.3325	1.6149	1.7015	1.6847
		least	3.1261	3.3486	2.8881	1.7762	1.7902	1.8345
MNIST $5 \times [1024]$	ℓ_∞	runner-up	0.0060	0.0150	0.0062	3.9916	4.4614	3.7635
		random	0.0073	0.0202	0.0077	3.5068	4.4069	3.7387
		least	0.0084	0.0230	0.0091	3.9076	4.6283	3.9730
	ℓ_2	runner-up	0.1369	0.3153	0.1426	4.1634	4.3311	4.1039
		random	0.1660	0.4254	0.1774	4.1468	4.1797	4.0898
		least	0.1909	0.4849	0.2096	4.5045	4.4773	4.5497
	ℓ_1	runner-up	1.1242	2.0616	1.2388	4.4911	3.9944	4.4436
		random	1.3952	2.8082	1.5842	4.4543	4.0839	4.2609
		least	1.6231	3.2201	1.9026	4.4674	4.5508	4.5154
CIFAR-10 $5 \times [2048]$	ℓ_∞	runner-up	0.0005	-	0.0006	37.3918	-	37.1383
		random	0.0008	-	0.0009	38.0841	-	37.9199
		least	0.0010	-	0.0011	39.1638	-	39.4041
	ℓ_2	runner-up	0.0219	-	0.0256	47.4896	-	48.3390
		random	0.0368	-	0.0406	54.0104	-	52.7471
		least	0.0460	-	0.0497	55.8924	-	56.3877
	ℓ_1	runner-up	0.3744	-	0.4491	46.4041	-	47.1640
		random	0.6384	-	0.7264	54.2138	-	51.6295
		least	0.8051	-	0.8955	56.2512	-	55.6069
CIFAR-10 $6 \times [2048]$	ℓ_∞	runner-up	0.0004	-	0.0003	59.5020	-	58.2473
		random	0.0006	-	0.0006	59.7220	-	58.0388
		least	0.0006	-	0.0007	60.8031	-	60.9790
	ℓ_2	runner-up	0.0177	-	0.0163	78.8801	-	72.1884
		random	0.0254	-	0.0251	84.2228	-	83.1202
		least	0.0294	-	0.0306	86.2997	-	86.9320
	ℓ_1	runner-up	0.3043	-	0.2925	78.7486	-	70.2496
		random	0.4406	-	0.4620	89.7717	-	83.7972
		least	0.5129	-	0.5665	87.2094	-	86.6502
CIFAR-10 $7 \times [1024]$	ℓ_∞	runner-up	0.0006	-	0.0005	20.8612	-	20.5169
		random	0.0008	-	0.0007	21.4550	-	21.2134
		least	0.0008	-	0.0008	21.3406	-	21.1804
	ℓ_2	runner-up	0.0260	-	0.0225	27.9442	-	27.0240
		random	0.0344	-	0.0317	30.3782	-	29.8086
		least	0.0376	-	0.0371	30.7492	-	30.7321
	ℓ_1	runner-up	0.3826	-	0.3648	28.1898	-	27.1238
		random	0.5087	-	0.5244	29.6373	-	30.5106
		least	0.5595	-	0.6171	31.3457	-	30.6481