CNN-Cert: An Efficient Framework for Certifying Robustness of Convolutional Neural Networks

Akhilan Boopathy,[1] Tsui-Wei Weng,[1] Pin-Yu Chen,[2] Sijia Liu,[2] Luca Daniel[1]
[1]Massachusetts Institute of Technology, Cambridge, MA 02139
[2]MIT-IBM Watson AI Lab, IBM Research

Abstract

Verifying robustness of neural network classifiers has attracted great interests and attention due to the success of deep neural networks and their unexpected vulnerability to adversarial perturbations. Although finding minimum adversarial distortion of neural networks (with ReLU activations) has been shown to be an NP-complete problem, obtaining a non-trivial lower bound of minimum distortion as a provable robustness guarantee is possible. However, most previous works only focused on simple fully-connected layers (multilayer perceptrons) and were limited to ReLU activations. This motivates us to propose a general and efficient framework, CNN-Cert, that is capable of certifying robustness on general convolutional neural networks. Our framework is general – we can handle various architectures including convolutional layers, max-pooling layers, batch normalization layer, residual blocks, as well as general activation functions; our approach is efficient – by exploiting the special structure of convolutional layers, we achieve up to 17 and 11 times of speed-up compared to the state-of-the-art certification algorithms (e.g. Fast-Lin, CROWN) and 366 times of speed-up compared to the dual-LP approach while our algorithm obtains similar or even better verification bounds. In addition, CNN-Cert generalizes state-of-the-art algorithms e.g. Fast-Lin and CROWN. We demonstrate by extensive experiments that our method outperforms state-of-the-art lower-bound-based certification algorithms in terms of both bound quality and speed.

Introduction

Recently, studies on adversarial robustness of state-of-the-art machine learning models, particularly neural networks (NNs), have received great attention due to interests in model explainability (Goodfellow, Shlens, and Szegedy 2015) and rapidly growing concerns on security implications (Biggio and Roli 2017). Take image recognition as a motivating example, imperceptible adversarial perturbations of natural images can be easily crafted to manipulate the model predictions, known as prediction-evasive adversarial attacks. One widely-used threat model to quantify the attack strengths is the norm-ball bounded attacks, where the distortion between an original example and the corresponding adversarial example is measured by the ℓ_p norm of their difference in real-valued vector representations (e.g., pixel values for images or embeddings for texts). Popular norm choices are ℓ_1 (Chen et al. 2018), ℓ_2 (Carlini and Wagner 2017b), and ℓ_∞ (Kurakin, Goodfellow, and Bengio 2017).

The methodology of evaluating model robustness against adversarial attacks can be divided into two categories: *game-based* or *verification-based*. Game-based approaches measure the success in mitigating adversarial attacks via mounting empirical validation against a (self-chosen) set of attacks. However, many defense methods have shown to be broken or bypassed by attacks that are adaptive to these defenses under the same threat model (Carlini and Wagner 2017a; Athalye, Carlini, and Wagner 2018), and therefore their robustness claims may not extend to untested attacks. On the other hand, verification-based approaches provide certified defense against any possible attacks under a threat model. In the case of an ℓ_p norm-ball bounded threat model, a verified robustness certificate ϵ means the (top-1) model prediction on the input data cannot be altered if the attack strength (distortion measured by ℓ_p norm) is smaller than ϵ. Different from game-based approaches, verification methods are attack-agnostic and hence can formally certify robustness guarantees, which is crucial to security-sensitive and safety-critical applications.

Although verification-based approaches can provide robustness certification, finding the minimum distortion (i.e., the maximum certifiable robustness) of NNs with ReLU activations has been shown to be an NP-complete problem (Katz et al. 2017). While minimum distortion can be attained in small and shallow networks (Katz et al. 2017; Lomuscio and Maganti 2017; Cheng, Nührenberg, and Ruess 2017; Fischetti and Jo 2017), these approaches are not even scalable to moderate-sized NNs. Recent works aim to circumvent the scalability issue by efficiently solving a non-trivial lower bound on the minimum distortion (Kolter and Wong 2018; Weng et al. 2018a; Dvijotham et al. 2018). However, existing methods may lack generality in supporting different network architectures and activation functions. In addition, current methods often deal with convolutional layers by simply converting back to fully-connected layers, which may lose efficiency if not fully optimized with respect to the NNs, as demonstrated in our experiments. To bridge this gap, we propose CNN-Cert, a general and efficient verification framework for certifying robustness of a broad range

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

Method	Non-trivial bound	Multi-layer	Scalability & Efficiency	Beyond ReLU	Exploit CNN structure	Pooling and other struc.
Reluplex (Katz et al. 2017), Planet (Ehlers 2017)	✓	✓	×	×	×	×
Global Lipschitz constant (Szegedy et al. 2013)	×	✓	✓	✓	×	✓
Local Lipschitz constant (Hein and Andriushchenko 2017)	✓	×	✓	differentiable	×	×
SDP approach (Raghunathan, Steinhardt, and Liang 2018)	✓	×	×	✓	×	×
Dual approach (Kolter and Wong 2018)	✓	✓	✓	×	×	×
Dual approach (Dvijotham et al. 2018)	✓	✓	codes not yet released	✓	×	✓
Fast-lin / Fast-lip (Weng et al. 2018a)	✓	✓	✓	×	×	×
CROWN (Zhang et al. 2018)	✓	✓	✓	✓	×	×
CNN-Cert (This work)	✓	✓	✓	✓	✓	✓

of convolutional neural networks (CNNs). The generality of CNN-Cert enables robustness certification of various architectures, including convolutional layers, max-pooling layers batch normalization layers and residual blocks, and general activation functions. The efficiency of CNN-Cert is optimized by exploiting the convolution operation. A full comparison of verification-based methods is given in Table 1.

We highlight the contributions of this paper as follows.

- CNN-Cert is *general* – it can certify robustness on general CNNs with various building blocks, including convolutional/pooling/batch-norm layers and residual blocks, as well as general activation functions such as ReLU, tanh, sigmoid and arctan. Other variants can easily be incorporated. Moreover, certification algorithms Fast-Lin (Weng et al. 2018a) and CROWN (Zhang et al. 2018) are special cases of CNN-Cert.

- CNN-Cert is *computationally efficient* – the cost is similar to forward-propagation as opposed to NP-completeness in formal verification methods, e.g. Reluplex (Katz et al. 2017). Extensive experiments show that CNN-Cert achieves up to 17 times of speed-up compared to state-of-the-art certification algorithms Fast-Lin and up to 366 times of speed-up compared to dual-LP approaches while CNN-Cert obtains similar or even better verification bounds.

Background and Related Work

Adversarial Attacks and Defenses. In the white-box setting where the target model is entirely transparent to an adversary, recent works have demonstrated adversarial attacks on machine learning applications empowered by neural networks, including object recognition (Szegedy et al. 2013), image captioning (Chen et al. 2017a), machine translation (Cheng et al. 2018b), and graph learning (Zügner, Akbarnejad, and Günnemann 2018). Even worse, adversarial attacks are still plausible in the black-box setting, where the adversary is only allowed to access the model output but not the model internals (Chen et al. 2017b; Ilyas et al. 2018; Tu et al. 2018; Cheng et al. 2018a). For improving the robustness of NNs, adversarial training with adversarial attacks is by far one of the most effective strategies that showed strong empirical defense performance (Madry et al. 2018; Sinha, Namkoong, and Duchi 2018). In addition, verification-based methods have validated that NNs with adversarial training can indeed improve robustness (Kolter and Wong 2018; Weng et al. 2018b).

Robustness Verification for Neural Networks. Under the norm-ball bounded threat model, for NNs with ReLU activation functions, although the minimum adversarial distortion gives the best possible certified robustness, solving it is indeed computationally intractable due to its NP-completeness complexity (Katz et al. 2017). Alternatively, solving a non-trivial lower bound of the minimum distortion as a provable robustness certificate is a more promising option but at the cost of obtaining a more conservative robustness certificate. Some analytical lower bounds depending solely on model weights can be derived (Szegedy et al. 2013; Peck et al. 2017; Hein and Andriushchenko 2017; Raghunathan, Steinhardt, and Liang 2018) but they are in general too loose to be useful or limited to 1 or 2 hidden layers. The robustness of NNs can be efficiently certified on ReLU activation (Kolter and Wong 2018; Weng et al. 2018a) and general activation (Zhang et al. 2018) but mostly on models with fully-connected layers. (Dvijotham et al. 2018) can also be applied to different activation functions but their bound quality might decrease a lot as a trade-off between computational efficiency due to its 'any-time' property. This paper falls within this line of research with an aim of providing both a general and efficient certification framework for CNNs (see Table 1 for detailed comparisons).

Threat model, minimum adversarial distortion ρ_{min} and certified lower bound ρ_{cert}. Throughout this paper, we consider the ℓ_p norm-ball bounded threat model with full access to all the model parameters. Given an input image $\mathbf{x_0}$ and a neural network classifier $f(\mathbf{x})$, let $c = \text{argmax}_i f_i(\mathbf{x_0})$ be the class where f predicts for $\mathbf{x_0}$. The minimum distortion ρ_{min} is the smallest perturbation that results in $\text{argmax}_i f_i(\mathbf{x_0}+\delta) \neq c$, and $\rho_{\text{min}} = \|\delta\|_p$. A certified lower bound ρ_{cert} satisfies the following: (i) $\rho_{\text{cert}} < \rho_{\text{min}}$ and (ii) for all $\delta \in \mathbb{R}^d$ and $\|\delta\|_p \leq \rho_{\text{cert}}$, $\text{argmax}_i f_i(\mathbf{x_0}+\delta) = c$. In other words, a certified bound *guarantees* a region (an ℓ_p ball with radius ρ_{cert}) such that the classifier decision can never be altered for all possible perturbations in that region. Note that ρ_{cert} is also known as *un-targeted* robustness, and the *targeted* robustness $\rho_{\text{cert,t}}$ is defined as satisfying (i) but with (ii) slightly modified as $\forall \delta \in \mathbb{R}^d$ and $\|\delta\|_p \leq \rho_{\text{cert}}$, $f_c(\mathbf{x_0}+\delta) > f_t(\mathbf{x_0}+\delta)$ given some targeted class $t \neq c$.

CNN-Cert: A General and Efficient Framework for Robustness Certification

Overview of our results. In this section, we present a general and efficient framework CNN-Cert for computing certified lower bounds of minimum adversarial distortion with general activation functions in CNNs. We derive the range of

Table 2: Expression of $\mathbf{A}_U^r$ and $\mathbf{B}_U^r$. $\mathbf{A}_L^r$ and $\mathbf{B}_L^r$ have exactly the same form as $\mathbf{A}_U^r$ and $\mathbf{B}_U^r$ but with U and L swapped.

Blocks	$\mathbf{A}^r_{U,(\vec{x},z),(\vec{i},k)}$	$\mathbf{B}^r_U$
(i) Act-Conv Block	$\mathbf{W}^{r+}_{(\vec{x},z),(\vec{i},k)}\alpha_{U,(\vec{i}+\vec{x},k)} + \mathbf{W}^{r-}_{(\vec{x},z),(\vec{i},k)}\alpha_{L,(\vec{i}+\vec{x},k)}$	$\mathbf{W}^{r+} * (\alpha_U \odot \beta_U) + \mathbf{W}^{r-} * (\alpha_L \odot \beta_L) + \mathbf{b}^r$
(ii) Residual Block	$[\mathbf{A}^r_{U,\text{act}} * \mathbf{W}^{r-1} + I]_{(\vec{i},k),(\vec{x},z)}$	$\mathbf{A}^r_{U,\text{act}} * \mathbf{b}^{r-1} + \mathbf{B}^r_{U,\text{act}}$
(iv) Pooling Block	$\frac{\mathbf{u}_{(\vec{i}+\vec{x},z)} - \gamma}{\mathbf{u}_{(\vec{i}+\vec{x},z)} - \mathbf{l}_{(\vec{i}+\vec{x},z)}}$ $\gamma = \min\{\max\{\gamma_0, \max \mathbf{l}_S\}, \min \mathbf{u}_S\}$	at location $(\vec{x},z)$: $\sum_{\vec{i}\in S_n} \frac{(\gamma - \mathbf{u}_{(\vec{i}+\vec{x},z)})\mathbf{l}_{(\vec{i}+\vec{x},z)}}{\mathbf{u}_{(\vec{i}+\vec{x},z)} - \mathbf{l}_{(\vec{i}+\vec{x},z)}} + \gamma$ $\gamma_0 = \frac{\sum_S \frac{\mathbf{u}_S}{\mathbf{u}_S - \mathbf{l}_S} - 1}{\sum_S \frac{1}{\mathbf{u}_S - \mathbf{l}_S}}$

Note 1: $(\vec{i},k) = (i,j,k)$ denotes filter coordinate indices and $(\vec{x},z) = (x,y,z)$ denotes output tensor indices.
Note 2: $\mathbf{A}_U^r, \mathbf{B}_U^r, \mathbf{W}, \alpha, \beta, \mathbf{u}, \mathbf{l}$ are all tensors. $\mathbf{W}^{r+}, \mathbf{W}^{r-}$ contains only the positive, negative entries of $\mathbf{W}^r$ with other entries equal 0.
Note 3: $\mathbf{A}_L^r, \mathbf{B}_L^r$ for pooling block are slightly different. Please see Appendix (c) for details.

network output in closed-form by applying a pair of linear upper/lower bound on the neurons (e.g. the activation functions, the pooling functions) when the input of the network is perturbed with noises bounded in ℓ_p norm ($p \geq 1$). Our framework can incorporate general activation functions and various architectures – particularly, we provide results on convolutional layers with activations (a.k.a Act-conv block), max-pooling layers (a.k.a. Pooling block), residual blocks (a.k.a. Residual block) and batch normalization layers (a.k.a. BN block). In addition, we show that the state-of-the-art Fast-Lin algorithm (Weng et al. 2018a) and CROWN (Zhang et al. 2018) are special cases under the CNN-Cert framework.

General framework

When an input data point is perturbed within an ℓ_p ball with radius ϵ, we are interested in the change of network output because this information can be used to find a certified lower bound of minimum adversarial distortion (as discussed in the section **Computing certified lower bound ρ_{cert}**). Toward this goal, the first step is to derive explicit output bounds for the neural network classifiers with various popular building blocks, as shown in Figure 1, Table 2 and Table 9 (with general strides and padding). The fundamental idea of our method is to apply *linear* bounding techniques separately on the *non-linear* operations in the neural networks, e.g. the non-linear activation functions, residual blocks and pooling operations. Our proposed techniques are general and allow efficient computations of certified lower bounds. We begin the formal introduction to CNN-Cert by giving notations and intuitions of deriving explicit bounds for each building block followed by the descriptions of utilizing such explicit bounds to compute certified lower bounds ρ_{cert} in our proposed framework.

Notations. Let $f(\mathbf{x})$ be a neural network classifier function and $\mathbf{x_0}$ be an input data point. We use $\sigma(\cdot)$ to denote the coordinate-wise activation function in the neural networks. Some popular choices of σ include ReLU: $\sigma(y) = \max(y, 0)$, hyperbolic tangent: $\sigma(y) = \tanh(y)$, sigmoid: $\sigma(y) = 1/(1+e^{-y})$ and arctan: $\sigma(y) = \tan^{-1}(y)$. The symbol $*$ denotes the convolution operation and $\Phi^r(\mathbf{x})$ denotes the output of r-th layer building block, which is a function of an input $\mathbf{x}$. We use superscripts to denote index of layers and subscripts to denote upper bound (U), lower bound (L) and its corresponding building blocks (e.g. act is short for activation, conv is short for convolution, res is short for residual block, bn is short for batch normalization and pool is short for pooling). Sometimes subscripts are also used to indicate the element index in a vector/tensor, which is self-content. We will often write $\Phi^r(\mathbf{x})$ as Φ^r for simplicity and we will sometimes use $\Phi^m(\mathbf{x})$ to denote the output of the classifier, i.e. $\Phi^m = f(\mathbf{x})$. Note that the weights $\mathbf{W}$, bias $\mathbf{b}$, input $\mathbf{x}$ and the output Φ^m of each layer are tensors since we consider a general CNN in this paper.

(i) Tackling the non-linear activation functions and convolutional layer. For the convolutional layer with an activation function $\sigma(\cdot)$, let Φ^{r-1} be the input of activation layer and Φ^r be the output of convolutional layer. The input/output relation is as follows:

$$\Phi^r = \mathbf{W}^r * \sigma(\Phi^{r-1}) + \mathbf{b}^r. \tag{1}$$

Given the range of Φ^{r-1}, we can bound the range of Φ^r by applying two linear bounds on each activation function $\sigma(y)$:

$$\alpha_L(y + \beta_L) \leq \sigma(y) \leq \alpha_U(y + \beta_U). \tag{2}$$

When the input y is in the range of $[l, u]$, the parameters $\alpha_L, \alpha_U, \beta_L, \beta_U$ can be chosen appropriately based on y's lower bound l and upper bound u. If we use (2) and consider the signs of the weights associated with the activation functions, it is possible to show that the output Φ^r in (1) can be bounded as follows:

$$\Phi^r \leq \mathbf{A}^r_{U,\text{act}} * \Phi^{r-1} + \mathbf{B}^r_{U,\text{act}}, \tag{3}$$

$$\Phi^r \geq \mathbf{A}^r_{L,\text{act}} * \Phi^{r-1} + \mathbf{B}^r_{L,\text{act}}, \tag{4}$$

where $\mathbf{A}^r_{U,\text{act}}, \mathbf{A}^r_{L,\text{act}}, \mathbf{B}^r_{U,\text{act}}, \mathbf{B}^r_{L,\text{act}}$ are constant tensors related to weights $\mathbf{W}^r$ and bias $\mathbf{b}^r$ as well as the corresponding parameters $\alpha_L, \alpha_U, \beta_L, \beta_U$ in the linear bounds of each neuron. See Table 2 for full results. Note the bounds in (3) and (4) are element-wise inequalities and we leave the derivations in the Appendix (a). On the other hand, if Φ^{r-1} is also the output of convolutional layer, i.e.

$$\Phi^{r-1} = \mathbf{W}^{r-1} * \sigma(\Phi^{r-2}) + \mathbf{b}^{r-1},$$

thus the bounds in (3) and (4) can be rewritten as follows:

$$\begin{aligned}\Phi^r &\leq \mathbf{A}^r_{U,\text{act}} * \Phi^{r-1} + \mathbf{B}^r_{U,\text{act}} \\ &= \mathbf{A}^r_{U,\text{act}} * (\mathbf{W}^{r-1} * \sigma(\Phi^{r-2}) + \mathbf{b}^{r-1}) + \mathbf{B}^r_{U,\text{act}} \\ &= \mathbf{A}^{r-1}_{U,\text{conv}} * \sigma(\Phi^{r-2}) + \mathbf{B}^{r-1}_{U,\text{conv}} + \mathbf{B}^r_{U,\text{act}} \end{aligned} \tag{5}$$

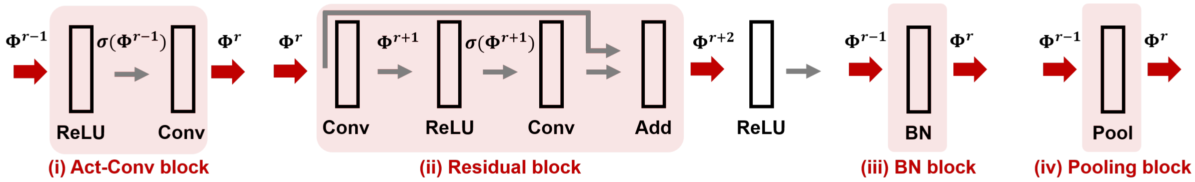

Figure 1: Cartoon graph of commonly-used building blocks (i)-(iv) considered in our CNN-Cert framework. The key step in deriving explicit network output bound is to consider the input/output relations of each building block, marked as red arrows. The activation layer can be general activations but here is denoted as ReLU.

and similarly

$$\begin{aligned}\Phi^r &\geq \mathbf{A}^r_{L,\text{act}} * \Phi^{r-1} + \mathbf{B}^r_{L,\text{act}} \\ &= \mathbf{A}^{r-1}_{L,\text{conv}} * \sigma(\Phi^{r-2}) + \mathbf{B}^{r-1}_{L,\text{conv}} + \mathbf{B}^r_{L,\text{act}} \qquad (6)\end{aligned}$$

by letting $\mathbf{A}^{r-1}_{U,\text{conv}} = \mathbf{A}^r_{U,\text{act}} * \mathbf{W}^{r-1}$, $\mathbf{B}^{r-1}_{U,\text{conv}} = \mathbf{A}^r_{U,\text{act}} * \mathbf{b}^{r-1}$, and $\mathbf{A}^{r-1}_{L,\text{conv}} = \mathbf{A}^r_{L,\text{act}} * \mathbf{W}^{r-1}$, $\mathbf{B}^{r-1}_{L,\text{conv}} = \mathbf{A}^r_{L,\text{act}} * \mathbf{b}^{r-1}$. Observe that the form of the upper bound in (5) and lower bound in (6) becomes the same convolution form again as (1). Therefore, for a neural network consists of *convolutional* layers and *activation* layers, the above technique can be used iteratively to obtain the final upper and lower bounds of the output Φ^r in terms of the input of neural network $\Phi^0(\mathbf{x}) = \mathbf{x}$ in the following convolutional form:

$$\mathbf{A}^0_{L,\text{conv}} * \mathbf{x} + \mathbf{B}^0_L \leq \Phi^r(\mathbf{x}) \leq \mathbf{A}^0_{U,\text{conv}} * \mathbf{x} + \mathbf{B}^0_U.$$

In fact, the above framework is very general and is not limited to the *convolution-activation* building blocks. The framework can also incorporate popular residual blocks, pooling layers and batch normalization layers, etc. The key idea is to derive linear upper bounds and lower bounds for each building block in the form of (3) and (4), and then plug in the corresponding bounds and *back-propagate* to the previous layer.

(ii) Tackling the residual blocks operations. For the residual block, let Φ^{r+2} denote the output of residual block (before activation) and Φ^{r+1} be the output of first convolutional layer and Φ^r be the input of residual block. The input/output relation is as follows:

$$\begin{aligned}\Phi^{r+1} &= \mathbf{W}^{r+1} * \Phi^r + \mathbf{b}^{r+1}, \\ \Phi^{r+2} &= \mathbf{W}^{r+2} * \sigma(\Phi^{r+1}) + \mathbf{b}^{r+2} + \Phi^r.\end{aligned}$$

Similar to the linear bounding techniques for up-wrapping the non-linear activation functions, the output of residual block can be bounded as:

$$\begin{aligned}\Phi^{r+2} &\leq \mathbf{A}^{r+2}_{U,\text{res}} * \Phi^r + \mathbf{B}^{r+2}_{U,\text{res}}, \\ \Phi^{r+2} &\geq \mathbf{A}^{r+2}_{L,\text{res}} * \Phi^r + \mathbf{B}^{r+2}_{L,\text{res}},\end{aligned}$$

where $\mathbf{A}^{r+2}_{U,\text{res}}, \mathbf{A}^{r+2}_{L,\text{res}}, \mathbf{B}^{r+2}_{U,\text{res}}, \mathbf{B}^{r+2}_{L,\text{res}}$ are constant tensors related to weights $\mathbf{W}^{r+2}$, $\mathbf{W}^{r+1}$, bias $\mathbf{b}^{r+2}$, $\mathbf{b}^{r+1}$, and the corresponding parameters $\alpha_L, \alpha_U, \beta_L, \beta_U$ in the linear bounds of each neuron; see Table 2 for details. Note that in Table 2, all indices are shifted from $r+2$ to r. The full derivations are provided in the Appendix (b).

(iii) Tackling the batch normalization. The batch normalization layer performs operations of scaling and shifting during inference time. Let Φ^r be the output and Φ^{r-1} be the input, the input/output relation is the following:

$$\Phi^r = \gamma_{\text{bn}} \frac{\Phi^{r-1} - \mu_{\text{bn}}}{\sqrt{\sigma^2_{\text{bn}} + \epsilon_{\text{bn}}}} + \beta_{\text{bn}},$$

where γ_{bn}, β_{bn} are the learned training parameters and μ_{bn}, σ^2_{bn} are the running average of the batch mean and variance during training. Thus, it is simply scaling and shifting on both upper bounds and lower bounds:

$$\mathbf{A}^r_{L,\text{bn}} * \Phi^{r-1} + \mathbf{B}^r_{L,\text{bn}} \leq \Phi^r \leq \mathbf{A}^r_{U,\text{bn}} * \Phi^{r-1} + \mathbf{B}^r_{U,\text{bn}}$$

where $\mathbf{A}^r_{U,\text{bn}} = \mathbf{A}^r_{L,\text{bn}} = \frac{\gamma_{\text{bn}}}{\sqrt{\sigma^2_{\text{bn}} + \epsilon_{\text{bn}}}}$ and $\mathbf{B}^r_{U,\text{bn}} = \mathbf{B}^r_{L,\text{bn}} = -\gamma_{\text{bn}} \frac{\mu_{\text{bn}}}{\sqrt{\sigma^2_{\text{bn}} + \epsilon_{\text{bn}}}} + \beta_{\text{bn}}$.

(iv) Tackling the pooling operations. Let Φ^r and Φ^{r-1} be the output and input of the pooling layer. For max-pooling operations, the input/output relation is the following:

$$\Phi^r_n = \max_{S_n} \Phi^{r-1}_{S_n},$$

where S_n denotes the pooled input index set associated with the n-th output. When the input Φ^{r-1} is bounded in the range $[\mathbf{l}^r, \mathbf{u}^r]$, it is possible to bound the output Φ^r by linear functions as follows:

$$\begin{aligned}\Phi^r &\leq \mathbf{A}^r_{U,\text{pool}} * \Phi^{r-1} + \mathbf{B}^r_{U,\text{pool}}, \\ \Phi^r &\geq \mathbf{A}^r_{L,\text{pool}} * \Phi^{r-1} + \mathbf{B}^r_{L,\text{pool}},\end{aligned}$$

where $\mathbf{A}^r_{U,\text{pool}}, \mathbf{A}^r_{L,\text{pool}}, \mathbf{B}^r_{U,\text{pool}}, \mathbf{B}^r_{L,\text{pool}}$ are constant tensors related to $\mathbf{l}^r$ and $\mathbf{u}^r$. For average pooling operation, the range of the output Φ^r is simply the the average of $\mathbf{l}^r$ and $\mathbf{u}^r$ on the corresponding pooling indices. See Table 2 and derivation details in Appendix (c).

Computing global bounds $\eta_{j,U}$ and $\eta_{j,L}$ of network output $\Phi^m(\mathbf{x})$. Let $\Phi^m(\mathbf{x})$ be the output of a m-th layer neural network classifier. We have shown that when the input of each building block is bounded and lies in the range of some $[\mathbf{l}, \mathbf{u}]$, then the output of the building block can be bounded by two linear functions in the form of input convolution. Since a neural network can be regarded as a cascade of building blocks – the input of current building block is the output of previous building block – we can propagate the bounds from the last building block that relates the network output *backward* to the first building block that relates the network input $\mathbf{x}$. A final upper bound and lower bound connect the network output and input are in the following linear relationship:

$$\mathbf{A}_L^0 * \mathbf{x} + \mathbf{B}_L^0 \leq \Phi^m(\mathbf{x}) \leq \mathbf{A}_U^0 * \mathbf{x} + \mathbf{B}_U^0. \quad (7)$$

Recall that the input $\mathbf{x}$ is constrained within an ℓ_p ball $\mathbb{B}_p(\mathbf{x_0}, \epsilon)$ centered at input data point $\mathbf{x_0}$ and with radius ϵ. Thus, maximizing (minimizing) the right-hand side (left-hand side) of (7) over $\mathbf{x} \in \mathbb{B}_p(\mathbf{x_0}, \epsilon)$ leads to a global upper (lower) bound of j-th output $\Phi_j^m(\mathbf{x})$:

$$\eta_{j,U} = \epsilon \|\text{vec}(\mathbf{A}_U^0)\|_q + \mathbf{A}_U^0 * \mathbf{x_0} + \mathbf{B}_U^0, \quad (8)$$

$$\eta_{j,L} = -\epsilon \|\text{vec}(\mathbf{A}_L^0)\|_q + \mathbf{A}_L^0 * \mathbf{x_0} + \mathbf{B}_L^0, \quad (9)$$

where $\|\cdot\|_q$ is ℓ_q norm and $1/p + 1/q = 1$ with $p, q \geq 1$.

Computing certified lower bound ρ_{cert}. Recall that the predicted class of input data $\mathbf{x_0}$ is c and let t be a targeted class. Given the magnitude of largest input perturbation ϵ, we can check if the output $\Phi_c^m(\mathbf{x}) - \Phi_t^m(\mathbf{x}) > 0$ by applying the global bounds derived in (8) and (9). In other words, given an ϵ, we will check the condition if $\eta_{c,L} - \eta_{t,U} > 0$. If the condition is true, we can increase ϵ; otherwise decrease ϵ. Thus, the largest certified lower bound can be attained by a bisection on ϵ. Note that although there is an explicit ϵ term in (8) and (9), they are *not* a linear function in ϵ because all the intermediate bounds of Φ^r depend on ϵ. Fortunately, we can still find ρ_{cert} numerically via the aforementioned bisection method. On the other hand, also note that the derivation of output bounds Φ^r in each building block depend on the range $[\mathbf{l}^{r-1}, \mathbf{u}^{r-1}]$ of the building block input (say Φ^{r-1}), which we call the intermediate bounds. The value of intermediate bounds can be computed similarly by treating Φ^{r-1} as the final output of the sub-network which consists of all building blocks before layer $r-1$ and deriving the corresponding $\mathbf{A}_U^0, \mathbf{A}_L^0, \mathbf{B}_U^0, \mathbf{B}_L^0$ in (7). Thus, all the intermediate bounds also have the same explicit forms as (8) and (9) but substituted by its corresponding $\mathbf{A}_U^0, \mathbf{A}_L^0, \mathbf{B}_U^0, \mathbf{B}_L^0$.

Discussions: Fast-Lin and CROWN are special cases of CNN-Cert. Fast-Lin (Weng et al. 2018a) and CROWN (Zhang et al. 2018) are special cases of CNN-Cert. In Fast-Lin, two linear bounds with the same slope (i.e. $\alpha_U = \alpha_L$ in (2)) are applied on the ReLU activation while in CROWN and CNN-Cert different slopes are possible (α_U and α_L can be different). However, both Fast-Lin and CROWN only consider fully-connected layers (MLP) while CNN-Cert can handle various building blocks and architectures such as residual blocks, pooling blocks and batch normalization blocks and is hence a more general framework. We show in Table 13 (appendix) that when using the same linear bounds in ReLU activations, CNN-Cert obtains the same robustness certificate as CROWN; meanwhile, for the general activations, CNN-Cert uses more accurate linear bounds and thus achieves better certificate quality up to 260% compared with CROWN (if we use exactly the same linear bounds, then CNN-Cert and CROWN indeed get the same certificate). Note that in all cases, CNN-Cert is much faster than CROWN (2.5-11.4× speed-up) due to the advantage of explicit convolutional bounds in CNN-Cert.

Discussion: CNN-Cert is computationally efficient. CNN-Cert has a similar cost to forward-propagation for general convolutional neural networks – it takes polynomial time, unlike algorithms that find the exact minimum adversarial distortion such as Reluplex (Katz et al. 2017) which is NP-complete. As shown in the experiment sections, CNN-Cert demonstrates an empirical speedup as compared to (a) the original versions of Fast-Lin (b) an optimized sparse matrix versions of Fast-Lin (by us) and (c) Dual-LP approaches while maintaining similar or better certified bounds (the improvement is around 8-20 %). For a pure CNN network with m layers, k-by-k filter size, n filters per layer, input size r-by-r, and stride 1-by-1, the time complexity of CNN-Cert is $O(r^2m^6k^4n^3)$. The equivalent fully connected network requires $O(r^6m^2n^3)$ time to certify.

Discussion: Training-time operations are independent of CNN-Cert. Since CNN-Cert is certifying the robustness of a fixed classifier f at the testing time, techniques that only apply to the training phase, such as dropout, will not affect the operation of CNN-Cert (though the given model to be certified might vary if model weights differ).

Experiments

We conduct extensive experiments comparing CNN-Cert with other lower-bound based verification methods on 5 classes of networks: (I) pure CNNs; (II) general CNNs (ReLU) with pooling and batch normalization; (III) residual networks (ReLU); (IV) general CNNs and residual networks with non-ReLU activation functions; (V) small MLP models. Due to page constraints, we refer readers to the appendix for additional results. Our codes are available at https://github.com/AkhilanB/CNN-Cert.

Comparative Methods.

- Certification algorithms: (i) Fast-Lin provides certificate on ReLU networks (Weng et al. 2018a); (ii) Global-Lips provides certificate using global Lipschitz constant (Szegedy et al. 2013); (iii) Dual-LP solves dual problems of the LP formulation in (Kolter and Wong 2018), and is the best result that (Dvijotham et al. 2018) can achieve, although it might not be attainable due to the any-time property; (iv) Reluplex (Katz et al. 2017) obtains exact minimum distortion but is computationally expensive.
- Robustness estimation, Attack methods: (i) CLEVER (Weng et al. 2018b) is a robustness estimation score without certification; (ii) CW/EAD are attack methods (Carlini and Wagner 2017b; Chen et al. 2018).
- Our methods: CNN-Cert-Relu is CNN-Cert with the same linear bounds on ReLU used in Fast-Lin, while CNN-Cert-Ada uses adaptive bounds all activation functions. CNNs are converted into equivalent MLP networks before evaluation for methods that only support MLP networks.

Implementations, Models and Dataset. CNN-Cert is implemented with Python (numpy with numba) and we also implement a version of Fast-Lin using sparse matrix multiplication for comparison with CNN-Cert since convolutional layers correspond to sparse weight matrices. Experiments

Table 3: Averaged bounds of CNN-Cert and other methods on **(I) pure CNN networks with ReLU activations** , see **Comparative Methods** section for methods descriptions. '-' indicates the method is computationally infeasible.

Network	Certified lower bounds					CNN-Cert-Ada Improvement (%)		Attack	Uncertified
	ℓ_p norm	CNN-Cert-Ada	Fast-Lin	Global-Lips	Dual-LP	vs. Fast-Lin	vs. Dual-LP	CW/EAD	CLEVER
MNIST, 4 layer	ℓ_∞	0.0491	0.0406	0.0002	0.0456	+21%	+8%	0.1488	0.0542
5 filters	ℓ_2	0.1793	0.1453	0.0491	0.1653	+23%	+8%	3.1407	1.0355
8680 hidden nodes	ℓ_1	0.3363	0.2764	0.0269	0.3121	+22%	+8%	14.4516	4.2955
MNIST, 4 layer	ℓ_∞	0.0340	0.0291	0.0000	-	+17%	-	0.1494	0.0368
20 filters	ℓ_2	0.1242	0.1039	0.0221	-	+20%	-	3.0159	0.7067
34720 hidden nodes	ℓ_1	0.2404	0.1993	0.0032	-	+21%	-	13.7950	3.4716
MNIST, 5 layer	ℓ_∞	0.0305	0.0248	0.0000	-	+23%	-	0.1041	0.0576
5 filters	ℓ_2	0.1262	0.1007	0.0235	-	+25%	-	1.8443	0.9011
10680 hidden nodes	ℓ_1	0.2482	0.2013	0.0049	-	+23%	-	11.6711	3.5369
CIFAR, 7 layer	ℓ_∞	0.0042	0.0036	0.0000	-	+17%	-	0.0229	0.0110
5 filters	ℓ_2	0.0340	0.0287	0.0023	-	+18%	-	0.6612	0.3503
19100 hidden nodes	ℓ_1	0.1009	0.0843	0.0001	-	+20%	-	12.5444	1.2138
CIFAR, 5 layer	ℓ_∞	0.0042	0.0037	0.0000	-	+14%	-	0.0172	0.0075
10 filters	ℓ_2	0.0324	0.0277	0.0042	-	+17%	-	0.4177	0.2390
29360 hidden nodes	ℓ_1	0.0953	0.0806	0.0005	-	+18%	-	11.6536	1.5539

Table 4: Averaged runtime of CNN-Cert and other methods on **(I) pure CNN networks with ReLU activations**

Network	Average Computation Time (sec)					CNN-Cert-Ada Speed-up			
	ℓ_p norm	CNN-Cert-Ada	Fast-Lin	Global-Lips	Dual-LP	vs. Fast-Lin, sparse	vs. Fast-Lin	vs. Dual-LP	vs. CLEVER
MNIST, 4 layer	ℓ_∞	2.33	9.03	0.0001	853.20	1.9	3.9	366.1	31.4
5 filters	ℓ_2	0.88	9.19	0.0001	236.30	5.0	10.5	270.1	83.0
8680 hidden nodes	ℓ_1	0.86	8.98	0.0001	227.69	5.2	10.5	265.2	87.1
MNIST, 4 layer	ℓ_∞	17.27	173.43	0.0001	-	2.0	10.0	-	11.2
20 filters	ℓ_2	17.19	180.10	0.0002	-	2.1	10.5	-	11.4
34720 hidden nodes	ℓ_1	17.35	179.63	0.0001	-	2.1	10.4	-	11.0
MNIST, 5 layer	ℓ_∞	4.96	16.89	0.0001	-	1.4	3.4	-	19.0
5 filters	ℓ_2	2.25	18.47	0.0001	-	3.0	8.2	-	46.8
10680 hidden nodes	ℓ_1	2.32	16.70	0.0001	-	3.0	7.2	-	43.6
CIFAR, 7 layer	ℓ_∞	15.11	78.04	0.0001	-	1.5	5.2	-	12.3
5 filters	ℓ_2	16.11	73.08	0.0001	-	1.4	4.5	-	11.8
19100 hidden nodes	ℓ_1	14.93	76.89	0.0001	-	1.5	5.1	-	12.9
CIFAR, 5 layer	ℓ_∞	20.87	169.29	0.0001	-	1.5	8.1	-	8.0
10 filters	ℓ_2	16.93	170.42	0.0002	-	2.0	10.1	-	9.2
29360 hidden nodes	ℓ_1	17.07	168.30	0.0001	-	1.9	9.9	-	9.3

Table 5: Averaged bounds and runtimes on **(II) general CNN networks with ReLU activations**.

Network	Certified Bounds				CNN-Cert-Ada Imp. (%)	Attack	Uncertified	Average Computation Time (sec)		
	ℓ_p norm	CNN-Cert-Relu	CNN-Cert-Ada	Global-Lips	vs. CNN-Cert-Relu	CW/EAD	CLEVER	CNN-Cert-Ada	Global-Lips	CW/EAD
MNIST, LeNet	ℓ_∞	0.0113	0.0120	0.0002	+6%	0.1705	0.0714	9.54	0.0001	20.50
	ℓ_2	0.0617	0.0654	0.0600	+6%	5.1327	1.2580	9.46	0.0001	5.56
	ℓ_1	0.1688	0.1794	0.0023	+6%	21.6101	5.5241	9.45	0.0001	3.75
MNIST, 7 layer	ℓ_∞	0.0068	0.0079	0.0000	+16%	0.1902	0.1156	191.81	0.0001	41.13
	ℓ_2	0.0277	0.0324	0.0073	+17%	4.9397	1.7703	194.82	0.0007	10.83
	ℓ_1	0.0542	0.0638	0.0000	+18%	19.6854	6.8565	188.84	0.0001	6.31
MNIST, LeNet	ℓ_∞	0.0234	0.0273	0.0001	+17%	0.1240	0.1261	10.05	0.0001	36.08
No Pooling	ℓ_2	0.1680	0.2051	0.0658	+22%	3.7831	2.4130	10.76	0.0003	8.17
	ℓ_1	0.5425	0.6655	0.0184	+23%	22.2273	10.6149	11.63	0.0001	5.34
MNIST, 4 layer	ℓ_∞	0.0083	0.0105	0.0011	+26%	0.0785	0.0318	2.35	0.0001	30.49
5 filters	ℓ_2	0.0270	0.0333	0.3023	+23%	0.8678	0.6284	2.42	0.0002	8.26
Batch Norm	ℓ_1	0.0485	0.0604	0.1053	+25%	6.1088	2.4622	2.39	0.0001	5.46
MNIST, 4 layer	ℓ_∞	0.0406	0.0492	0.0002	+21%	0.1488	0.0536	1.66	0.0001	22.23
5 filters	ℓ_2	0.1454	0.1794	0.0491	+23%	3.1407	1.0283	1.31	0.0001	5.78
	ℓ_1	0.2764	0.3363	0.0269	+22%	14.4516	4.4930	1.49	0.0001	3.98
Tiny ImageNet	ℓ_∞	0.0002	0.0003	-	+24%	0.4773	0.0056	5492.35	-	257.06
7 layer	ℓ_2	0.0012	0.0016	-	+29%	-	0.4329	5344.49	-	-
	ℓ_1	0.0038	0.0048	-	+28%	-	7.1665	5346.08	-	-
MNIST, LeNet	ℓ_∞	0.0117	0.0124	0.0003	+6%	0.1737	0.0804	6.89	0.0001	38.76
	ℓ_2	0.0638	0.0678	0.0672	+6%	5.1441	1.4599	6.85	0.0001	9.22
100 images	ℓ_1	0.1750	0.1864	0.0027	+7%	22.7232	5.7677	6.91	0.0001	5.57
MNIST, 4 layer	ℓ_∞	0.0416	0.0500	0.0002	+20%	0.1515	0.0572	0.98	0.0001	40.02
5 filters	ℓ_2	0.1483	0.1819	0.0516	+23%	3.2258	1.0834	0.85	0.0001	8.93
100 images	ℓ_1	0.2814	0.3409	0.0291	+21%	14.7665	4.2765	0.83	0.0001	6.25

are conducted on a AMD Zen server CPU. We evaluate CNN-Cert and other methods on CNN models trained on the MNIST, CIFAR-10 and tiny Imagenet datasets. All pure convolutional networks use 3-by-3 convolutions. The general 7-layer CNNs use two max pooling layers and uses 32 and 64 filters for two convolution layers each. LeNet uses a similar architecture to LeNet-5 (LeCun et al. 1998), with the no-pooling version applying the same convolutions over larger inputs. The residual networks (ResNet) evaluated use simple residual blocks with two convolutions per block and

Table 6: Averaged bounds and runtimes on **(III) ResNet with ReLU activations** .

Network	Certified Bounds			CNN-Cert-Ada Imp. (%)	Attack	Uncertified	Average Computation Time (sec)		
	ℓ_p norm	CNN-Cert-Relu	CNN-Cert-Ada	vs. CNN-Cert-Relu	CW/EAD	CLEVER	CNN-Cert-Relu	CNN-Cert-Ada	CW/EAD
MNIST, ResNet-2	ℓ_∞	0.0183	0.0197	+8%	0.0348	0.0385	2.26	2.25	24.96
	ℓ_2	0.0653	0.0739	+13%	0.2892	0.7046	2.21	2.25	5.59
	ℓ_1	0.1188	0.1333	+12%	4.8225	2.2088	2.19	2.22	3.00
MNIST, ResNet-3	ℓ_∞	0.0179	0.0202	+13%	0.0423	0.0501	10.39	10.04	32.82
	ℓ_2	0.0767	0.0809	+5%	0.3884	1.0704	10.13	10.11	6.89
	ℓ_1	0.1461	0.1514	+4%	5.9454	3.8978	10.20	10.15	4.22
MNIST, ResNet-4	ℓ_∞	0.0153	0.0166	+8%	0.0676	0.0455	28.66	28.18	35.13
	ℓ_2	0.0614	0.0683	+11%	1.0094	0.9621	28.43	28.20	7.89
	ℓ_1	0.1012	0.1241	+23%	9.1925	3.7999	27.81	28.53	5.34

Table 7: Averaged bounds and runtimes on **(IV) general CNNs and ResNet with general activation functions**. 7-layer sigmoid network results are omitted due to poor test accuracy.

Network	Certified lower bounds						Uncertified	Average Computation Time (sec)				
	ℓ_p norm	CNN-Cert-Relu	CNN-Cert-Ada	Sigmoid	Tanh	Arctan	CLEVER	CNN-Cert-Relu	CNN-Cert-Ada	Sigmoid	Tanh	Arctan
MNIST, Pure CNN	ℓ_∞	0.0203	0.0237	0.0841	0.0124	0.0109	0.0354	18.34	18.27	18.81	20.31	19.03
8-layer	ℓ_2	0.0735	0.0877	0.3441	0.0735	0.0677	0.4268	18.25	18.22	18.83	19.70	19.05
5 filters	ℓ_1	0.1284	0.1541	0.7319	0.1719	0.1692	1.2190	18.35	18.51	19.40	20.00	19.36
MNIST, General CNN	ℓ_∞	0.0113	0.0120	0.0124	0.0170	0.0153	0.0714	9.71	9.54	9.55	9.66	9.37
LeNet	ℓ_2	0.0617	0.0654	0.0616	0.1012	0.0912	1.2580	9.45	9.46	9.42	9.49	9.50
	ℓ_1	0.1688	0.1794	0.1666	0.2744	0.2522	5.5241	9.44	9.45	9.59	9.69	9.86
MNIST, General CNN	ℓ_∞	0.0068	0.0079	-	0.0085	0.0079	0.1156	193.68	191.81	-	191.26	195.08
7-layer	ℓ_2	0.0277	0.0324	-	0.0429	0.0386	1.7703	194.21	194.82	-	193.85	194.81
	ℓ_1	0.0542	0.0638	-	0.0955	0.0845	6.8565	187.88	188.84	-	188.83	188.79
MNIST, ResNet-3	ℓ_∞	0.0179	0.0202	0.0042	0.0058	0.0048	0.0501	10.39	10.04	10.08	10.39	10.26
	ℓ_2	0.0767	0.0809	0.0147	0.0223	0.0156	1.0704	10.13	10.11	10.14	10.43	10.27
	ℓ_1	0.1461	0.1514	0.0252	0.0399	0.0277	3.8978	10.20	10.15	10.40	10.84	10.69

Table 8: Averaged bounds and runtimes on **(V) small MLP networks**.

Network	Certified Bounds					CNN-Cert-Ada Improvement (%)		Exact	Attack	Uncertified
	ℓ_p norm	Fast-Lin	CNN-Cert-Relu	CNN-Cert-Ada	Dual-LP	vs. Fast-Lin	vs. Dual-LP	Reluplex	CW/EAD	CLEVER
MNIST, 2 layer	ℓ_∞	0.0365	0.0365	0.0371	0.0372	+2%	0%	0.0830	0.0871	0.0526
20 nodes	ℓ_2	0.7754	0.7754	0.7892	0.9312	+2%	-15%	-	1.9008	1.1282
Fully Connected	ℓ_1	5.3296	5.3252	5.4452	5.7583	+2%	-5%	-	15.8649	7.8207
MNIST, 3 layer	ℓ_∞	0.0297	0.0297	0.0305	0.0308	+3%	-1%	-	0.0835	0.0489
20 nodes	ℓ_2	0.6286	0.6289	0.6464	0.7179	+3%	-10%	-	2.3083	1.0214
Fully Connected	ℓ_1	4.2631	4.2599	4.4258	4.5230	+4%	-2%	-	15.9909	6.9988

ResNet with k residual blocks is denoted as ResNet-k. We evaluate all methods on 10 random test images and attack targets (in order to accommodate slow verification methods) and also 100 images results for some networks in Table 5. It shows that the results of average 100 images are similar to average 10 imagess. We train all models for 10 epochs and tune hyperparameters to optimize validation accuracy.

Results (I): pure CNNs with ReLU activation. Table 3 demonstrates that CNN-Cert bounds consistently improve on Fast-Lin over network size. CNN-Cert also improves on Dual-LP. Attack results show that all certified methods leave a significant gap on the attack-based distortion bounds (i.e. upper bounds on the minimum distortions). Table 4 gives the runtimes of various methods and shows that CNN-Cert is faster than Fast-Lin, with over an order of magnitude speedup for the smallest network. CNN-Cert is also faster than the sparse version of Fast-Lin. The runtime improvement of CNN-Cert decreases with network size. Notably, CNN-Cert is multiple orders of magnitude faster than the Dual-LP method. Global-Lips is an analytical bound, but it provides very loose lower bounds by merely using the product of layer weights as the Lipschitz constant. In contrast, CNN-Cert takes into account the network output at the neuron level and thus can certify significantly larger lower bounds, and is around 8-20 % larger compared to Fast-Lin and Dual-LP approaches.

Results (II), (III): general CNNs and ResNet with ReLU activation. Table 5 gives certified lower bounds for various general CNNs including networks with pooling layers and batch normalization. CNN-Cert improves upon Fast-Lin style ReLU bounds (CNN-Cert-Relu). Interestingly, the LeNet style network without pooling layers has certified bounds much larger than the pooling version while the network with batch normalization has smaller certified bounds. These findings provide some new insights on uncovering the relation between certified robustness and network architecture, and CNN-Cert could potentially be leveraged to search for more robust networks. Table 6 gives ResNet results and shows CNN-Cert improves upon Fast-Lin.

Results (IV): general CNNs and ResNet with general activations. Table 7 computes certified lower bounds for networks with 4 different activation functions. Some sigmoid network results are omitted due to poor test set accuracy. We conclude that CNN-Cert can indeed efficiently find nontrivial lower bounds for all the tested activation functions and that computing certified lower bounds for general activation functions incurs no significant computational penalty.

Results (V): Small MLP networks. Table 8 shows results on small MNIST MLP with 20 nodes per layer. For the small 2-layer network, we are able to run Reluplex and compute minimum adversarial distortion. It can be seen that the gap between the certified lower bounds method here are all around 2 times while CLEVER and attack methods are close to Reluplex though without guarantees.

Conclusion and Future Work

In this paper, we propose CNN-Cert, a general and efficient verification framework for certifying robustness of CNNs. By applying our proposed linear bounding technique on each building block, CNN-Cert can handle a wide variety of network architectures including convolution, pooling, batch normalization, residual blocks, as well as general activation functions. Extensive experimental results under four different classes of CNNs consistently validate the superiority of CNN-Cert over other methods in terms of its effectiveness in solving tighter non-trivial certified bounds and its run time efficiency.

Acknowledgement

Akhilan Boopathy, Tsui-Wei Weng and Luca Daniel are partially supported by MIT-IBM Watson AI Lab.

References

Athalye, A.; Carlini, N.; and Wagner, D. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *ICML*.

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

Carlini, N., and Wagner, D. 2017a. Adversarial examples are not easily detected: Bypassing ten detection methods. *arXiv preprint arXiv:1705.07263*.

Carlini, N., and Wagner, D. 2017b. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (SP)*, 39–57.

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

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

Chen, P.-Y.; Sharma, Y.; Zhang, H.; Yi, J.; and Hsieh, C.-J. 2018. Ead: elastic-net attacks to deep neural networks via adversarial examples. *AAAI*.

Cheng, M.; Le, T.; Chen, P.-Y.; Yi, J.; Zhang, H.; and Hsieh, C.-J. 2018a. Query-efficient hard-label black-box attack: An optimization-based approach. *arXiv preprint arXiv:1807.04457*.

Cheng, M.; Yi, J.; Zhang, H.; Chen, P.-Y.; and Hsieh, C.-J. 2018b. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. *arXiv preprint arXiv:1803.01128*.

Cheng, C.-H.; Nührenberg, G.; and Ruess, H. 2017. Maximum resilience of artificial neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, 251–268. Springer.

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

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

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

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

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

Ilyas, A.; Engstrom, L.; Athalye, A.; and Lin, J. 2018. Black-box adversarial attacks with limited queries and information. *arXiv preprint arXiv:1804.08598*.

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

Kolter, J. Z., and Wong, E. 2018. Provable defenses against adversarial examples via the convex outer adversarial polytope. *ICML*.

Kurakin, A.; Goodfellow, I.; and Bengio, S. 2017. Adversarial machine learning at scale. *ICLR*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

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

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards deep learning models resistant to adversarial attacks. *ICLR*.

Peck, J.; Roels, J.; Goossens, B.; and Saeys, Y. 2017. Lower bounds on the robustness to adversarial perturbations. In *NIPS*.

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

Sinha, A.; Namkoong, H.; and Duchi, J. 2018. Certifiable distributional robustness with principled adversarial training. *ICLR*.

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

Tu, C.-C.; Ting, P.; Chen, P.-Y.; Liu, S.; Zhang, H.; Yi, J.; Hsieh, C.-J.; and Cheng, S.-M. 2018. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. *arXiv preprint arXiv:1805.11770*.

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

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

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

Zügner, D.; Akbarnejad, A.; and Günnemann, S. 2018. Adversarial attacks on neural networks for graph data. In *KDD*.