Fast Training of Provably Robust Neural Networks by *SingleProp*

Akhilan Boopathy[1], **Lily Weng** [1], **Sijia Liu**[2], **Pin-Yu Chen**[2], **Gaoyuan Zhang**[2], **Luca Daniel**[1]
[1] Massachusetts Institute of Technology
[2] MIT-IBM Watson AI Lab, IBM Research
akhilan@mit.edu

Abstract

Recent works have developed several methods of defending neural networks against adversarial attacks with certified guarantees. However, these techniques can be computationally costly due to the use of certification during training. We develop a new regularizer that is both more efficient than existing certified defenses, requiring only one additional forward propagation through a network, and can be used to train networks with similar certified accuracy. Through experiments on MNIST and CIFAR-10 we demonstrate improvements in training speed and comparable certified accuracy compared to state-of-the-art certified defenses.

1 Introduction

Although deep neural networks (DNNs) have achieved tremendous success in various applications, it has become widely-known that they are vulnerable to adversarial examples (also known as adversarial attacks), namely, crafted examples with human-imperceptible perturbations to cause misclassification (Goodfellow, Shlens, and Szegedy 2015; Szegedy et al. 2013). Many attack generation methods have been proposed in order to find the possible minimum adversarial perturbation, commonly evaluated by its ℓ_p norm for $p \in \{0, 1, 2, \infty\}$ (Papernot et al. 2016; Carlini and Wagner 2017; Athalye and Sutskever 2017; Su, Vargas, and Sakurai 2019; Xu et al. 2018; Chen et al. 2018). Meanwhile, various defense methods were proposed to enhance the robustness of DNNs against adversarial attacks. However, many of them are built on heuristic strategies, which are thus easily bypassed by stronger adversaries (Athalye, Carlini, and Wagner 2018). The work (Madry et al. 2018) proposed a stronger defense method, adversarial training, which minimizes the *worst-case* training loss under adversarial perturbations.

Motivated by the limitation of heuristic defense, another line of research (known as verified/certified robustness) aims to provide *provable* robustness guarantees of DNNs against an input with *arbitrary perturbation* within a certain ℓ_p ball region (Katz et al. 2017; Cheng, Nührenberg, and Ruess 2017; Carlini et al. 2017; Kolter and Wong 2018; Raghunathan, Steinhardt, and Liang 2018; Weng et al. 2018; Zhang et al. 2018; Boopathy et al. 2019; Dvijotham et al. 2018b; Wong et al. 2018; Xiao et al. 2019; Gowal et al. 2019; Mirman, Gehr, and Vechev 2018; Dvijotham et al. 2018a). The recent progress on verification spans from the exact verification method (Katz et al. 2017; Cheng, Nührenberg, and Ruess 2017; Carlini et al. 2017) to the relaxed verification method (Kolter and Wong 2018; Raghunathan, Steinhardt, and Liang 2018; Weng et al. 2018; Singh et al. 2019; Zhang et al. 2018; Boopathy et al. 2019; Dvijotham et al. 2018b). Here the former uses expensive computation methods, e.g., mixed-integer programming (MIP), to find the exact minimum adversarial perturbation, and the latter considers a relaxed verification problem by convexifying the adversarial polytope which significantly improves the computation efficiency compared to the exact method at the cost of tightness of robustness certificate.

Another research direction that has attracted a lot of interest in this field is robust training, i.e. to make a neural network classifier more robust to adversarial attacks. Recent work (Xiao et al. 2019) proposed the principle of co-design between training and verification, and showed that the exact verification method (Tjeng and Tedrake 2019) can be accelerated by imposing weight sparsity and activation stability (so-called ReLU stability) on trainable neural network models. On the other hand, there are several works (Kolter and Wong 2018; Raghunathan, Steinhardt, and Liang 2018; Gowal et al. 2019; Mirman, Gehr, and Vechev 2018) aiming to train a more *verifiable* model targeted on different verifiers mentioned earlier (Kolter and Wong 2018; Raghunathan, Steinhardt, and Liang 2018; Weng et al. 2018; Singh et al. 2019; Zhang et al. 2018; Boopathy et al. 2019; Wong et al. 2018) and this line of research is known as *certified robust training*. The idea is to incorporate the robustness verification bounds into the training process and thus the learnt model yields strengthened robustness with certificate. Nevertheless, current verification-based training methods are multiple times slower than standard (non-robust) training per training step. In particular, using convex outer bounds-based methods (Kolter and Wong 2018; Weng et al. 2018; Zhang et al. 2018) empirically require more than $100\times$ standard training cost (Kolter and Wong 2018). The fastest method to date, interval bound propagation (IBP) (Gowal et al. 2019), requires 2 additional forward propagations compared to standard training, for a total training time of $3\times$ standard training. We highlight that IBP is significantly faster than adversarial

training, which typically requires much greater than 2 adversarial steps to achieve high robustness (Madry et al. 2018). As such, IBP is currently the fastest effective certified robust training method.

While IBP empirically achieves high certified accuracies, i.e. higher percentage of the test images that are guaranteed to be classified correctly under any possible ℓ_p perturbations with magnitude ϵ, it still requires $2\times$ additional computation overhead of standard training. This raises the question of what minimum computational overhead is required to achieve certified robustness for a neural network model, which is the main motivation of this work. Our goal is to develop a robust training method that achieves high certified accuracy while achieving the minimum overhead of *only one additional* forward pass than standard training. We summarize our contributions as follows:

- We propose an efficient robust training algorithm, named as SingleProp, based on a novel regularizer which can be derived as an approximation of linear bounding verifiers (Kolter and Wong 2018; Weng et al. 2018; Zhang et al. 2018; Boopathy et al. 2019). Our proposed regularizer requires only 1 additional forward pass per training step (hence the name SingleProp) relative to standard training, resulting in only $1\times$ additional training time and memory overhead relative to standard training. To our best knowledge, **SingleProp** is the fastest and most efficient among SOTA certified training algorithms.
- Extensive experiments demonstrate SingleProp achieves superior computational efficiency and comparable certified accuracies compared to the current fastest certified robust training method IBP (Gowal et al. 2019). In particular, we show on both MNIST and CIFAR datasets that the certified accuracies only decrease slightly while enjoying $1.5\times$-$2\times$ faster to train as well as $1.5\times$ reduction in memory usage. While this drop in accuracy is expected due to the approximation used in our method, we observe that SingleProp outperforms IBP training on the Fast-Lin verifier (Weng et al. 2018).

2 Background and Related Work

2.1 Verifications

Assuming a norm-bounded threat model, finding the minimum adversarial distortion exactly is an NP-complete problem, making it computationally infeasible (Katz et al. 2017). Fortunately, finding lower bounds on the minimum adversarial distortion is computationally tractable. Several techniques find these lower bounds only as a function of model weights (Szegedy et al. 2013; Peck et al. 2017; Hein and Andriushchenko 2017; Raghunathan, Steinhardt, and Liang 2018), but these methods typically provide very loose bounds for neural networks with more than 2 layers. Using an input-specific certification method, it is possible to find non-trivial bounds for fully connected ReLU networks (Kolter and Wong 2018; Weng et al. 2018; Wang et al. 2018), as well as networks with general activation functions (Zhang et al. 2018) and general CNN and RNN architectures (Boopathy et al. 2019; Ko et al. 2019).

2.2 Certified Defenses

Recent works have also developed methods of defending against adversarial attacks. One line of work uses adversarial training with adversarial attacks and empirically demonstrates high resistance to attacks (Madry et al. 2018; Sinha, Namkoong, and Duchi 2018). However, adversarial training is not targeted towards verification or certification methods and is therefore not considered as a certified defense. One step towards certified defenses is natural regularizations on the model parameters, such as sparsity-inducing weight magnitude penalization. This method combined with adversarial training yields highly verifiable models (Xiao et al. 2019). Using an additional ReLU stability regularizer to enhance ease-of-verification allows for even more verifiable models, but at the cost of at least $3\times$ of standard training even without adversarial training. Since layer-specific regularizations alone empirically do not help certifiable robustness, in this paper we focus on the minimum computational overhead achievable using additional forward passes.

Other defenses specifically target certifiers or use certification methods as part of the training procedure, and this type of defenses is known as *certified defense*. We note that these "certified" defenses are not truly certified since they cannot ensure robustness to unseen points without using a certifier on these points. These defenses instead produce models that are empirically more certifiable on unseen test points. Using convex outer bounds to bound the adversarial loss function has been shown to be effective at producing more certifiable models on the targeted verifiers (Kolter and Wong 2018; Weng et al. 2018), although training is relatively slow (Kolter and Wong 2018; Wong et al. 2018). Using interval bounds propagation (IBP) to bound the adversarial loss is much cheaper to train (Gowal et al. 2019) and has surprisingly become the state-of-the-art certifiably robust training method (Gowal et al. 2019; Salman et al. 2019) despite IBP generally performing much worse than the convex outer bounds (Kolter and Wong 2018; Wong et al. 2018) in certifications of standard networks. Recently, another verifier was developed by combining IBP with the CROWN certifier (Zhang et al. 2018) to produce CROWN-IBP training. The authors show that the trained networks could outperform IBP in certified accuracy by up to 2.74% on MNIST and up to 9.16% on CIFAR . However, CROWN-IBP is much more computationally expensive during training, specifically $9\times$ slower than IBP (Zhang et al. 2020), which is equivalent to imposing total additional $26\times$ cost of standard training. As our emphasis is on the *efficiency* of certifiable robust training, we focus on comparing with IBP rather than CROWN-IBP. In summary, we develop an efficient certified defense that has even lower computation overhead (only $1\times$ additional to standard training cost) than IBP-based defenses (Gowal et al. 2019; Zhang et al. 2020) while yielding comparable performance with existing methods.

Threat Model. In this paper, we will use the notation of fully-connected neural networks for exposition, but our method works for general convolutional neural networks including residual networks. Appendix A Table 2 includes

descriptions of the main notations used in this paper. Consider an n layer neural network $f(\mathbf{x})$ with input $\mathbf{x}$ where the first layer of the network $\mathbf{z}^0$ is set to $\mathbf{x}$. Given weights $\mathbf{W}^i$, biases $\mathbf{b}^i$ and an activation function σ, for $i = 0, \ldots, n-1$, subsequent layers are defined as:

$$\mathbf{z}^{i+1} = \mathbf{W}^{i+1}\sigma(\mathbf{z}^i) + \mathbf{b}^{i+1}, \tag{1}$$

with $f(\mathbf{x}) = \mathbf{z}^n$. With this expression, the first layer of the network is defined by using the identity activation at the first layer. We assume the following threat model: a nominal input $\mathbf{x}_{nom}$ is perturbed by perturbation δ to produce a perturbed input $\mathbf{x} = \mathbf{x}_{nom} + \delta$, where $||\delta||_p \leq \epsilon$ and $||\cdot||_p$ represents an ℓ_p norm. Suppose the correct classification is given by c. Then the minimum distortion ϵ^* for misclassification is the minimal $\epsilon \in \mathbb{R}^+$ satisfying: $\max_{j \neq c} \mathbf{z_j^n} - \mathbf{z_c^n} > 0$.

Interval Bounds Propagation. There exist several methods to efficiently find certified lower bounds on the minimum distortion necessary for misclassification. One such method is interval bounds propagation (IBP) (Gowal et al. 2019; Gehr et al. 2018) which bounds each layer in a network with a fixed upper and lower bound. These bounds are then propagated at each layer of the network using the previous layer's bounds. Specifically, given layer-wise bounds where $\mathbf{l}^i \leq \mathbf{z^i} \leq \mathbf{u}^i$, the next layer's bounds are found as:

$$\mathbf{u}^{i+1} = \mathbf{W}_+^{i+1}\sigma(\mathbf{u}^i) + \mathbf{W}_-^{i+1}\sigma(\mathbf{l}^i) + \mathbf{b}^{i+1}, \tag{2}$$

where $\mathbf{W}_+$ and $\mathbf{W}_-$ denote the positive and negative components of $\mathbf{W}$ respectively with other entries being zeros otherwise. Lower bounds are found similarly. Intuitively, IBP finds a box bounding each layer, which can result in very loose bounds for general network as demonstrated in (Kolter and Wong 2018; Gehr et al. 2018).

Linear Bounding Framework. Certified bounds can also be found using a linear bounding framework as first proposed in Fast-Lin (Weng et al. 2018) and later in the Neurify (Wang et al. 2018) and DeepZ (Singh et al. 2018) frameworks. This approach typically finds tighter bounds on minimum distortion than IBP. This framework bounds each activation layer $\sigma(\mathbf{z}^i)$ as follows: $\alpha_L^i \odot \mathbf{z}^i + \beta_L^i \leq \sigma(\mathbf{z}^i) \leq \alpha_U^i \odot \mathbf{z}^i + \beta_U^i$, where α_L^i, α_U^i represents the slopes of linear bounds on the activation and β_L^i,β_U^i represents intercepts of linear bounds on the activation. When σ is the ReLU activation, provided bounds $\mathbf{z}_L^i, \mathbf{z}_U^i$ on $\mathbf{z}^i$ satisfying $\mathbf{z}_L^i \leq \mathbf{z}^i \leq \mathbf{z}_U^i$, Fast-Lin sets the coefficients to be:

$$\alpha_{L,j}^i = \alpha_{U,j}^i = \frac{\mathbf{z}_{U,j}^i}{\mathbf{z}_{U,j}^i - \mathbf{z}_{L,j}^i},\ \beta_{L,j}^i = 0,\ \beta_{U,j}^i = -\frac{\mathbf{z}_{U,j}^i \mathbf{z}_{L,j}^i}{\mathbf{z}_{U,j}^i - \mathbf{z}_{L,j}^i},$$

if the neuron j is uncertain, meaning $\mathbf{z}_{L,j}^i < 0$, $\mathbf{z}_{U,j}^i > 0$. When both bounds are positive or negative, the bound on the activation is exactly the linear component on the corresponding side (i.e. when $\mathbf{z}_{L,j}^i > 0$ for example, $\alpha_{L,j}^i = \alpha_{U,j}^i = 1$, $\beta_{L,j}^i = \beta_{U,j}^i = 0$). Using these layer-wise bounds, Fast-Lin finds a pair of linear bounds on the network: $\mathbf{A}_L\mathbf{x} + \mathbf{b}_L \leq f(\mathbf{x}) \leq \mathbf{A}_U\mathbf{x} + \mathbf{b}_U$. Then Fast-Lin bounds the network output over all possible adversarial distortions measured by ϵ-ℓ_p ball by:

$$\mathbf{A}_L\mathbf{x}_{nom} + \mathbf{b}_L - \epsilon||\mathbf{A}_L||_{:,q} \leq f(\mathbf{x}) \leq \mathbf{A}_U\mathbf{x}_{nom} + \mathbf{b}_U + \epsilon||\mathbf{A}_U||_{:,q}, \tag{3}$$

where $||\cdot||_{:,q}$ denotes a row-wise q norm, dual to the norm p of the assumed attack threat model. ϵ is the assumed attack norm size. Intuitively, Fast-Lin finds linear upper and lower bounds on the entire network to analyze the output layer. Because Fast-Lin finds linear bounds on the network as an intermediate step to finding output bounds $\mathbf{z}_L, \mathbf{z}_U$, the bounds are tighter than the corresponding IBP bounds $\mathbf{u}, \mathbf{l}$. Fast-Lin is equivalent to using convex outer bounds to bound the set of possible values at each layer of the network. Fast-Lin has been extended to general activation functions and asymmetric upper and lower bounds with different values of α_L^i, α_U^i in CROWN (Zhang et al. 2018), and has been extended to general network architectures in CNN-Cert (Boopathy et al. 2019).

3 SingleProp: An Efficient Robust Training Framework

In this section, we propose a new robust training method SingleProp which is $1.5\times$ more computationally efficient than the most efficient SOTA certified training algorithm. We start by first deriving SingleProp regularizers as approximations of linear bounding certifiers in Sec 3.1. We then analyse the run time of our method in Sec 3.2 and detail the training procedure in Sec 3.3.

3.1 Robust Loss Function with SingleProp Regularizers

Let θ denotes the parameters of neural network f_θ, and let $\mathcal{L}_\theta(\mathbf{z}^n, \mathbf{y})$ be a standard loss function as a function of network output $\mathbf{z}^n$ and one-hot-encoded label $\mathbf{y}$. Many robust training methods including IBP can all be interpreted as adding a regularizer $\mathcal{R}_\theta(\mathbf{l}^n, \mathbf{u}^n, \mathbf{y})$ to the standard loss function and thus forming a *robust* loss function: $\mathcal{L}_\theta(\mathbf{z}^n, \mathbf{y}) + \lambda\mathcal{R}_\theta(\mathbf{l}^n, \mathbf{u}^n, \mathbf{y})$, where $\mathbf{u}^n$ and $\mathbf{l}^n$ are layer-wise output bounds of the neural network $f_\theta(\mathbf{x})$ when the input $\mathbf{x}$ is perturbed and λ is a regularization parameter. The IBP regularizer can be written in the following form (Gowal et al. 2019):

$$\mathcal{R}_\theta(\mathbf{l}^n, \mathbf{u}^n, \mathbf{y}) = \mathbb{E}[\mathcal{L}(\mathbf{l}^n \circ \mathbf{y} + \mathbf{u}^n \circ (1-\mathbf{y}), \mathbf{y}) - \mathcal{L}(\mathbf{z}_{nom}^n, \mathbf{y})], \tag{4}$$

where $\mathbf{z}_{nom}^n = f_\theta(\mathbf{x}_{nom})$ is the output at unperturbed input $\mathbf{x}_{nom}$ and $\circ$ denotes element-wise multiplication. However, IBP requires two additional propagations through the network during training relative to standard training due to computing $\mathbf{u}^n$ and $\mathbf{l}^n$. Instead of using two quantities $\mathbf{u}^n$ and $\mathbf{l}^n$ to propagate uncertainty, we design a new regularizer:

$$\begin{aligned}\mathcal{R}_\theta(\mathbf{v}^n, \mathbf{z}_{nom}^n, \mathbf{y}) = \mathbb{E}[\mathcal{L}((\mathbf{z}_{nom}^n - \mathbf{v}^n) \circ \mathbf{y} \\ + (\mathbf{z}_{nom}^n + \mathbf{v}^n) \circ (1-\mathbf{y}), \mathbf{y}) - \mathcal{L}(\mathbf{z}_{nom}^n, \mathbf{y})]\end{aligned} \tag{5}$$

where $\mathbf{v}^n$ is a trainable quantity that can be computed with only a *single propagation* and is thus more efficient than the IBP regularizer. In fact, the quantity $\mathbf{v}^n$ is motivated by approximating the bounds in linear bounding certifiers such

as Fast-Lin (Weng et al. 2018). In the following, we show that it is possible to compute $\mathbf{v}^n$ with only a *single propagation* in each training step with the derived recursive relation Eq. (6).

Derivation of SingleProp Regularizer. We use fully-connected network for easier exposition, but our method can be easily extended to CNNs or residual networks. As described in Sec 2, the output of NN can be bounded by a lower bound $\mathbf{z}_L = \mathbf{A}_L\mathbf{x}_{nom} + \mathbf{b}_L - \epsilon||\mathbf{A}_L||_{:,1}$ and an upper bound $\mathbf{z}_U = \mathbf{A}_U\mathbf{x}_{nom} + \mathbf{b}_U + \epsilon||\mathbf{A}_U||_{:,1}$ in the linear bounding method assuming an ℓ_∞ perturbation norm[1]. At each layer i, we wish to approximate the bounds $\mathbf{z}_L^i$ and $\mathbf{z}_U^i$ using only a single forward propagation. To do so, we define the half bound gap at layer i as $\mathbf{v}^i = \frac{1}{2}(\mathbf{z}_U^i - \mathbf{z}_L^i)$. At all layers i, the dimension of each $\mathbf{v}^i$ is the same as the corresponding layer $\mathbf{z}^i$. We will show that for $i > 0$, $\mathbf{v}^i$ can be approximated recursively using only the previous value $\mathbf{v}^{i-1}$. To avoid the need for additional forward propagations, we will make an additional assumption on the *average* of the bounds $\frac{1}{2}(\mathbf{z}_U + \mathbf{z}_L)$. This will allow us to approximate the range of values each layer $\mathbf{z}$ can take as $[\mathbf{z}_{nom} - \mathbf{v}, \mathbf{z}_{nom} + \mathbf{v}]$, where $\mathbf{z}_{nom}$ represents the value of the layer for unperturbed input $\mathbf{x}_{nom}$. To derive the recursive approximation of $\mathbf{v}$, we first start with the definition of $\mathbf{v}^i$ in terms of $\mathbf{z}_L^i$ and $\mathbf{z}_U^i$ in (i) below. Expanding $\mathbf{z}_L^i$ and $\mathbf{z}_U^i$ by their definitions implies (ii). The terms involving $||\cdot||_{:,1}$ can be upper bounded to yield (iii) using the fact that $||AB||_{:,1} \leq |A||B||_{:,1}$ elementwise. This can be expressed recursively in terms of $\mathbf{v}^{i-1}$ in (iv). Finally, using the assumption that the average of the bounds is approximately the value of the network at $\mathbf{x}_{nom}$ (i.e. $\mathbf{z}_{nom}^{i-1} \approx \frac{1}{2}[\mathbf{z}_U^{i-1} + \mathbf{z}_L^{i-1}]$), this reduces to (*):

$$
\begin{aligned}
\mathbf{v}^i &\underset{\text{(i)}}{=} \frac{1}{2}(\mathbf{z}_U^i - \mathbf{z}_L^i) \\
&\underset{\text{(ii)}}{=} \frac{1}{2}[|\mathbf{W}^i|(\alpha_U^{i-1}(\mathbf{z}_U^{i-1} - \epsilon||\mathbf{A}_U^{i-1}||_{:,1}) + \beta_U^{i-1}) \\
&\quad - |\mathbf{W}^i|(\alpha_L^{i-1}(\mathbf{z}_L^{i-1} + \epsilon||\mathbf{A}_L^{i-1}||_{:,1}) + \beta_L^{i-1})] \\
&\quad + \frac{1}{2}[\epsilon||\mathbf{A}_U^i||_{:,1} + \epsilon||\mathbf{A}_L^i||_{:,1}] \\
&\underset{\text{(iii)}}{\leq} \frac{1}{2}[|\mathbf{W}^i|(\alpha_U^{i-1}\mathbf{z}_U^{i-1} + \beta_U^{i-1}) - |\mathbf{W}^i|(\alpha_L^{i-1}\mathbf{z}_L^{i-1} + \beta_L^{i-1})] \\
&\underset{\text{(iv)}}{=} |\mathbf{W}^i|\frac{\alpha_U^{i-1} + \alpha_L^{i-1}}{2}\mathbf{v}^{i-1} + \frac{1}{2}|\mathbf{W}^i|(\beta_U^{i-1} - \beta_L^{i-1}) \\
&\quad + |\mathbf{W}^i|\frac{\alpha_U^{i-1} - \alpha_L^{i-1}}{4}(\mathbf{z}_U^{i-1} + \mathbf{z}_L^{i-1}) \\
&\underset{(*)}{\approx} |\mathbf{W}^i|[\frac{\alpha_U^{i-1} + \alpha_L^{i-1}}{2}\mathbf{v}^{i-1} + \frac{\beta_U^{i-1} - \beta_L^{i-1}}{2} \\
&\quad + \frac{\alpha_U^{i-1} - \alpha_L^{i-1}}{2}\mathbf{z}_{nom}^{i-1}] \qquad (6)
\end{aligned}
$$

Eq. (6) with equality defines the propagation of quantity $\mathbf{v}$ through the network. Given $\mathbf{v}^0$, this provides an update equation to find $\mathbf{v}$ for subsequent layers, which can be used to approximate bound margins at all layers. Since $\mathbf{v}^0$ corresponds to bounds at the first layer, $\mathbf{v}^0$ is initialized to $\epsilon\mathbf{1}$. We note that Eq. (6) actually holds as an approximate inequality ($\lesssim$), and by treating it as an equality ($=$) in our computation of $\mathbf{v}$, we approximately overestimate the true bounds of the linear bounding certifier. Despite being only an approximate overestimation of true bounds, using $\mathbf{v}$ during training is justifiable because empirically the approximation is highly accurate: averaging z_U, z_L is very close to z_{nom} with a difference is on the order of 1e-7 (see experiments in Sec 4.1).

Note that the specific values of α^{i-1} and β^{i-1} in Eq. (6) will depend on the exact bounds used for the activation function considered. Different bounds on the activation function correspond to approximating different certifiers. In the case of ReLU, the quantities in the expression above depend on exact values of $(\mathbf{z}_{nom}^{i-1} - \mathbf{v}^{i-1})_j$ and $(\mathbf{z}_{nom}^{i-1} + \mathbf{v}^{i-1})_j$ for each neuron j. In other words, the bracketed quantity in Eq. (6) related to neuron j can be written as:

$$\frac{\alpha_{U,j}^{i-1} + \alpha_{L,j}^{i-1}}{2}\mathbf{v}_j^{i-1} + \frac{\beta_{U,j}^{i-1} - \beta_{L,j}^{i-1}}{2} + \frac{\alpha_{U,j}^{i-1} - \alpha_{L,j}^{i-1}}{2}(\mathbf{z}_{nom}^{i-1})_j,$$

which is equivalent to the following equations depending on the neuron status:

$$
\begin{cases}
\alpha_j^{i-1}\mathbf{v}_j^{i-1} & \text{, if neuron } j \text{ is stable} \\
\frac{3}{4}\mathbf{v}_j^{i-1} + \frac{1}{2}(\mathbf{z}_{nom}^{i-1})_j - \frac{(\mathbf{z}_{nom}^{i-1})_j^2}{4\mathbf{v}_j^{i-1}} & \text{, if neuron } j \text{ is unstable}
\end{cases} \qquad (7)
$$

We refer a neuron to be stable if it satisfies $(\mathbf{z}_{nom}^{i-1} + \mathbf{v}^{i-1})_j(\mathbf{z}_{nom}^{i-1} - \mathbf{v}^{i-1})_j > 0$, and the resulting upper and lower bounds on ReLU are equal, where α_j^{i-1} is 1 for ReLUs with positive inputs and 0 for ReLUs with negative inputs. On the other hand, we refer a neuron to be unstable if $(\mathbf{z}_{nom}^{i-1} + \mathbf{v}^{i-1})_j > 0$ and $(\mathbf{z}_{nom}^{i-1} - \mathbf{v}^{i-1})_j < 0$), and the bracket quantity in Eq. (6) can be re-written as in Eq. (7), which we call this choice of activation bounds SingleProp-FastLin. For methods such as CROWN (Zhang et al. 2018) which use adaptive selection of lower bounds on ReLU, the value of expression for unstable neurons depends on the choice of lower bound slope $\alpha_{L,j}^{i-1}$. We consider the case where unstable neurons have a lower bound of slope 0 and we call this choice of activation bounds SingleProp-Zero, which yields the following equations:

$$
\begin{cases}
\alpha_j^{i-1}\mathbf{v}_j^{i-1} & \text{, if neuron } j \text{ is stable} \\
\frac{1}{2}(\mathbf{z}_{nom}^{i-1} + \mathbf{v}^{i-1})_j & \text{, if neuron } j \text{ is unstable}
\end{cases} \qquad (8)
$$

Note that these bounds correspond to a variation of CROWN (Zhang et al. 2018) where the lower bound is always chosen to have slope zero, which is a strictly stronger verifier than IBP since IBP can be seen as a variation of CROWN with constant upper and lower bounds. Therefore, SingleProp-Zero can be seen as approximating IBP bounds with a single additional propagation compared to two for IBP.

It is worth noting that deriving the true upper bound on the adversarial loss (as done by existing certified defenses (Kolter and Wong 2018; Raghunathan, Steinhardt, and Liang 2018; Gowal et al. 2019) is not necessary, since they cannot ensure

[1] other p can be derived similarly.

robustness to unseen points without using a certifier on these points. Hence, the success of SingleProp on achieving better efficiency is due to the use of the approximated upper bound on the adversarial loss, and the effectiveness of SingleProp is demonstrated in the our experiments. We are also not aware of any exact certification-based regularizer that is competitive with our method in terms of computational efficiency: we use only $2\times$ the memory and training time of standard training (vs. at least $3-27\times$ for other methods).

3.2 Runtime Analysis

The robust loss $\mathcal{L}_\theta + \lambda\mathcal{R}_\theta$ requires computation of both the last layer of the unperturbed network $\mathbf{z}^n_{nom}$ and $\mathbf{v}^n$. Note that given the unperturbed value of all intermediate layers $\mathbf{z}^i_{nom}$, $\mathbf{v}^i$ for all layers can be computed with a single forward pass using (6). Since the unperturbed layers can be computed with a single forward pass, computing the robust loss requires two forward passes total.

During training, computing gradients of the regularized loss with respect to network parameters requires computing the gradients with respect to intermediate layers which we denote as $\nabla\mathbf{z}^i_{nom}$ and $\nabla\mathbf{v}^i$ for all i. Note that by Equations (1) and (6), $\nabla\mathbf{z}^i_{nom}$ can be computed using the value of the next layer's gradients $\nabla\mathbf{z}^{i+1}_{nom}$ and $\nabla\mathbf{v}^{i+1}$. Similarly, $\nabla\mathbf{v}^i$ can be computed from $\nabla\mathbf{v}^{i+1}$. This implies that $\nabla\mathbf{v}^i$ for all layers can be computed with a single backward propagation. These gradients can be used to compute $\nabla\mathbf{z}^{i+1}_{nom}$ for all layers with a single additional backward propagation, resulting in two backward propagations total. In summary, SingleProp requires only one additional forward pass and one additional backward pass relative to standard training. Therefore, assuming that numerical operations of fixed dimensionality are performed constant time and treating as negligible the cost of layer-wise operations such as activation functions, SingleProp training only requires 1 additional forward pass and $1\times$ of additional memory overhead compared to standard training, which is $1.5\times$ faster than IBP (Gowal et al. 2019) (empirically 1.5-2$\times$ faster) in speed and $1.5\times$ reduction in the memory usage. To our best knowledge, **SingleProp** is the fastest and most efficient certified training procedures.

3.3 Training Procedure

Training proceeds by using standard optimizers on the robust loss. See Appendix B Algorithm 1 for the full procedure. Note that using a value of $\lambda = 0$ corresponds to standard training while using $\lambda > 0$ corresponds to using the regularizer (i.e. robust loss). In practice, robust training methods such as IBP typically increase the value of λ during training process. It is also worth mentioning that the value of λ is separate from the value of ϵ used during training which parameterizes the regularizer $\mathcal{R}_\theta$. A value of $\epsilon = 0$ corresponds to regular training, and higher values of ϵ correspond to defending against larger perturbation attacks. In addition to increasing λ during training, robust training methods also typically increase ϵ during training.

Adaptive Hyperparameter Selection The exact schedules of λ and ϵ represent a large hyperparameter space and can require careful tuning for methods like IBP (Gowal et al. 2019). To ensure consistent performance without extensive hyperparameter tuning, we propose an adaptive method of selecting the regularization hyperparameter λ using a validation set. Specifically, at each epoch, we set the value of λ as: $\lambda = \frac{\gamma\mathcal{L}_{val,\theta}(\mathbf{z}^n,\mathbf{y})}{(1+\gamma)\mathcal{L}_{val,\theta}(\mathbf{z}^n,\mathbf{y})+\mathcal{R}_{val,\theta}(\cdot,\cdot,\mathbf{y})}$, where $\mathcal{L}_{val,\theta}$ and $\mathcal{R}_{val,\theta}$ represent quantities computed on the validation set and $\gamma > 0$ is a constant hyperparameter. Intuitively, this choice of λ ensures that standard accuracy is maintained throughout the training process. If standard performance is high relative to robust performance, λ will be low and vice versa. Therefore, as robust performance increases during the course of training, λ will gradually increase. The parameter γ controls the trade-off between robustness and standard accuracy, with smaller choices of γ corresponding to a higher preference for standard accuracy. As a baseline, we also use a piece-wise linear schedule for λ starting from 0 and increasing to 0.5, following the parameters used by IBP (Gowal et al. 2019). We also use the piece-wise linear ϵ schedule suggested by IBP, where ϵ is set to 0 for a warm-up period, followed by a linear increase until reaching the desired value, after which ϵ stays at this value. The schedule of learning rates is tuned for each method individually using a validation set.

4 Experiments

Implementation, Architectures, Training Parameters. We directly use the code provided for IBP (Gowal et al. 2019) and use the same CNN architectures (small, medium, large and wide) on MNIST and CIFAR-10 datasets. We use adaptive hyperparameter selection for λ or a piecewise linear schedule for λ for all robust training methods. The MNIST networks are trained for 100 epochs each with a batch size of 100 while the CIFAR networks are trained for 350 epochs each with a batch size of 50. We use the standard values of $\epsilon = 0.3$ for MNIST and $\epsilon = 8/255$ as the training target perturbation size ϵ_{train}. Following (Gowal et al. 2019), the schedule of ϵ starts at 0 for a warmup period (2000 training steps on MNIST, 5000 training steps on CIFAR), followed by a linear increase to the desired target perturbation size (10000 training steps on MNIST, 50000 training steps on CIFAR), after which ϵ is fixed at the target level. Additional details are reported in Appendix C.

Comparative Methods & Evaluation Metric. We compare the two variants of our method SingleProp-FastLin and SingleProp-Zero with the baseline IBP (Gowal et al. 2019) on the same metric, i.e. certified accuracy (acc.). We also report the total combined fraction of points certifiable by *either* networks trained with IBP or SingleProp which we call IBP+SingleProp, which will always be greater or equal than the certified acc. by individual methods since it finds the union of points certifiable under individual IBP or SingleProp models. This ***does not*** correspond to certified accuracy on a robustly trained model or model ensemble, but is included to show the complementarity between the certifications of IBP and SingleProp models. The main verifier used to compute certified accs. is IBP, which will favor IBP-trained models. However, we show that SingleProp not only has competitive

certified accs. on the IBP verifier, it actually outperforms IBP-trained models on the Fast-Lin verifier (Weng et al. 2018). On the IBP verifier, we report the full test set results (10000 points) over a range of perturbation sizes ϵ in $[0, 0.4]$ for MNIST and $[0, 9/255]$ for CIFAR. For certain networks, we also compute certified accs. using CNN-Cert (Boopathy et al. 2019) under Fast-Lin type bounds (Weng et al. 2018) and a variation of CROWN type bounds (Zhang et al. 2018) with ReLU lower bound of slope zero, which we call CNN-Cert-Zero. Due to the computational cost of these certifiers, we certify on 100 randomly chosen test set points. We use the official code released by (Gowal et al. 2019) to implement IBP training.

Remark. In our experiments, we exactly match the parameters used by the authors, with the exception of the number of epochs and batch size used on CIFAR. The authors train networks for 3200 epochs with a batch size of 1600, which we find computationally infeasible. Instead, we use the setting of 350 epochs with a batch size of 50, which is provided as an alternative in (Gowal et al. 2019). This results in some discrepancy with the main results reported by (Gowal et al. 2019), but is more consistent with the results reported under the alternative parameter setting in Appendix A of (Gowal et al. 2019). In addition, we use IBP a as verifier while (Gowal et al. 2019) uses exact verifier MILP and consequently gets higher certified accuracy. To avoid other discrepancies, we train IBP networks using code from (Gowal et al. 2019), although to ensure a fair comparison, our reported training times are recorded under a consistent implementation of IBP and SingleProp.

4.1 Results

Results on MNIST and CIFAR. In Table 1 we find that on MNIST, SingleProp-Zero achieves similar performance to IBP on both clean acc. and robust acc. over all ϵ. Specifically, on the Small model, SingleProp-Zero achieves a clean acc. of 94.71% and a certified acc. of 82.93% at $\epsilon = 0.3$ compared to a clean acc. of 96.21% and a certified acc. of 84.82% for IBP. We observe similar results under an average of 5 trials (see App. D Table 7), with a maximum certified accuracy standard deviation of 2.36%. On the Medium model, SingleProp-Zero achieves a clean acc. of 97.45% and a certified acc. of 86.05% at $\epsilon = 0.3$ compared to a clean acc. of 97.17% and a certified acc. of 88.63% for IBP[2]. On the Wide model, SingleProp-Zero achieves a clean acc. of 97.01% and a certified acc. of 87.59% at $\epsilon = 0.3$ compared to a clean acc. of 98.52% and a certified acc. of 89.35% for IBP.

For the CIFAR Small model, SingleProp-FastLin achieves similar clean acc. and robust acc. over all ϵ under multiple trials (see App. D Table 7). Specifically, on the Small model, SingleProp-FastLin achieves a clean acc. range of $[36.97\%, 37.79\%]$ and a certified acc. range of $[23.94\%, 24.71\%]$ at $\epsilon = 8/255$ compared to a clean acc. range of $[36.46\%, 39.05\%]$ and a certified acc. of $[25.99\%, 26.57\%]$ for IBP. Interestingly, we observe that certified accs. are slightly lower on the Large model, but clean accs. are much larger with SingleProp achieving a clean acc. of 44.36% and a certified acc. of 21.94% at $\epsilon = 8/255$ compared to a clean acc. of 46.80% and a certified acc. of 25.68% for IBP.

Runtime Improvement. As seen in the last column of Table 1, SingleProp consistently trains in less time than IBP, with between $1.54\times$ and $2.24\times$ speedup (measured as IBP runtime/our runtime). This performance is faster than expected by the number of forward propagations: since IBP uses three forward propagations compared to two for SingleProp, a speedup of $1.5\times$ would be expected. Slower methods such as IBP or CROWN-IBP can achieve higher certified accuracies at the cost of computational efficiency, corresponding to different points on a trade-off curve between training time and certification level (see App. D Table 3). Thus, SingleProp may be preferable to other methods when considering computation and memory costs.

SingleProp Approximation Error. We evaluate the quality of the approximation in Eq. (6) which is used to derive SingleProp as an approximation of linear bounding certifiers. We compute two metrics: Metric 1 = $\sum_{i,j} |\mathbf{z}^i_{nom,j} - (\mathbf{z}^i_{U,j} + \mathbf{z}^i_{L,j})/2|$, metric 2 = $\sum_{i,j} |\mathbf{z}^i_{nom,j} - (\mathbf{z}^i_{U,j} + \mathbf{z}^i_{L,j})/2| / (\mathbf{z}^i_{U,j} - \mathbf{z}^i_{L,j})$. On Small CNN MNIST models, the (mean, std) of metric 1 is IBP: (8.83E-07, 1.57E-07), SingleProp-Zero: (4.08E-07, 3.80E-08), and on metric 2 is IBP: (0.018, 0.016), SingleProp-Zero: (0.003, 0.004). This shows that SingleProp-Zero indeed closely approximates upper and lower bounds on a linear bounding certifier on models trained with SingleProp-Zero or IBP.

Combining Model Certifications Greatly Improves Certified Accuracies. As observed in App. D Table 7, combining the points certified under IBP and SingleProp models individually increases certified accs. by 2-4% at $\epsilon = 0.3$ and by 6-9% at $\epsilon = 8/255$ on CIFAR. In other words, points uncertifiable by an IBP network are certifiable under a SingleProp network and vice versa. This demonstrates that IBP and SingleProp-FastLin networks complement each other in their certifications. We also evaluate the complementarity of IBP and SingleProp certifications in the set of points correctly classified by both methods ($\{C\}$ in Table 7), and find that nearly all points in this set are certifiable under at least one model. We note that bound complementarity is greater between IBP and SingleProp-FastLin (compared to SingleProp-Zero). This may be because while SingleProp-Zero approximates IBP (see Sec 3.1), SingleProp-FastLin approximates Fast-Lin, encouraging certifiability under Fast-Lin, which verifies different points than IBP. Therefore, SingleProp-FastLin could induce robustness in points that are difficult to certify under IBP-like training, leading to certifiability even under IBP verification.

[2]We note that (Gowal et al. 2019) report higher certified accs. (91.95% on MNIST and 32.04% on CIFAR). On CIFAR, the authors train for 3200 epochs, which we find computationally infeasible. With 350 epochs, the authors report a certified acc. of 28.30% which is more comparable with our results.

Method	$\epsilon_{cert} = 0$	0.05	0.10	0.20	0.30	Per epoch runtime (s)
Small CNN MNIST, 4 layer, $\epsilon_{train} = 0.3$						
IBP	96.21%	95.37%	94.35%	90.93%	84.82%	6.3
SingleProp-Zero	94.71%	93.70%	92.47%	88.96%	82.93%	**4.0**
Standard	99.09%	0.00%	0.00%	0.00%	0.00%	-
Adv Training (Madry et al. 2018)	99.14%	12.26%	0.00%	0.00%	0.00%	-
TRADES (Zhang et al. 2019)	99.09%	0.01%	0.00%	0.00%	0.00%	-
Improvement: Row 2 vs. IBP	-1.50%	-1.67%	-1.88%	-1.97%	-1.89%	**×1.58 faster**
Medium CNN MNIST, 7 layer, $\epsilon_{train} = 0.3$						
IBP	97.17%	96.49%	95.59%	93.09%	88.63%	15.0
SingleProp-Zero	97.45%	96.55%	95.46%	92.40%	86.05%	**7.1**
Improvement: Row 2 vs. IBP	+0.28%	+0.06%	-0.13%	-0.69%	-2.58%	**×2.12 faster**
Wide CNN MNIST, 5 layer, $\epsilon_{train} = 0.3$						
IBP	98.52%	98.12%	97.36%	94.93%	89.35%	87.7
SingleProp-Zero	97.01%	96.27%	95.32%	92.85%	87.59%	**53.1**
Standard	99.28%	0.00%	0.00%	0.00%	0.00%	-
Adv Training (Madry et al. 2018)	99.40%	0.00%	0.00%	0.00%	0.00%	-
Improvement: Row 2 vs. IBP	-1.51%	-1.85%	-2.04%	-2.08%	-1.76%	**×1.65 faster**
Method	$\epsilon_{cert} = 0$	$2/255$	$5/255$	$7/255$	$8/255$	Per epoch runtime (s)
Small CNN CIFAR, 4 layer, $\epsilon_{train} = 8/255$						
IBP	36.46%	33.70%	29.64%	27.14%	25.99%	14.8
SingleProp-FastLin	37.79%	34.39%	29.01%	26.04%	24.51%	**7.2**
Improvement: Row 2 vs. IBP	+1.33%	+0.69%	-0.63%	-1.10%	-1.48%	**×2.07 faster**
Large CNN CIFAR, 7 layer, $\epsilon_{train} = 8/255$						
IBP	46.80%	41.15%	33.16%	28.04%	25.68%	43.7
SingleProp-FastLin	44.36%	37.79%	29.51%	24.28%	21.94%	**21.6**
Improvement: Row 2 vs. IBP	-2.44%	-3.36%	-3.65%	-3.76%	-3.74%	**×2.02 faster**

Table 1: Full test set IBP-certified acc. for IBP and SingleProp-trained networks on MNIST and CIFAR-10. Per epoch runtimes are reported, with improvements computed as (IBP runtime/our runtime). For ease of presentation, we use row index to represent the method at that row.

Evaluating Under Other Certifiers. In App. D Table 5 we find that IBP and CNN-Cert-Zero certified accs. are similar for both IBP and SingleProp-Zero trained models. Because CNN-Cert-Zero bounds are stronger, but similar to IBP (see Sec 3.1), SingleProp-Zero can achieve high acc. even under IBP certification, and IBP also performs similarly on CNN-Cert-Zero as IBP certification. We also observe that under Fast-Lin, IBP networks perform worse than SingleProp-FastLin. In particular, SingleProp-FastLin maintains similar accs. under Fast-Lin as IBP, validating that SingleProp-FastLin is indeed adapted for Fast-Lin. Finally, including points certifiable by either verifier (Fast-Lin+IBP) results in similar accs. as only using IBP, justifying using IBP as our primary verifier. In App. D Table 6, we also evaluate certified accuracies on 200 points for selected networks and find similar results to 100 points.

Evaluating Adaptive Hyperparameter Selection (AHS). We evaluate the effect of adaptive hyperparameter selection on both IBP and SingleProp training by comparing AHS to the tuned piecewise linear schedule for λ used in (Gowal et al. 2019). In App. D Table 4, we show certified accs. under both schemes, reporting the best results under a grid search for learning rates. The piecewise linear schedule is denoted Linear. As illustrated, with the exception of the IBP models trained on Small CNN MNIST, adaptive hyperparameter selection increases certified accs. at $\epsilon = 0.3$ for MNIST and $\epsilon = 8/255$ for CIFAR. Moreover, for SingleProp models, certified accs. increase for most ϵ by up to about 1%, while decreasing at most by 0.05%. This indicates that AHS can achieve high robust accs. while avoiding tuning over the large space of hyperparameter schedules.

5 Conclusion

In this paper, we propose an efficient certified training framework, SingleProp, that requires only one additional forward propagation through a network. We have conducted a comprehensive comparison on SingleProp and current SOTA most efficient certified training framework, IBP, in terms of the training schedules, verifiers and complementary verification results. We found that we achieve comparable certified accuracies to state-of-the-art certified defenses while being 1.5-$2\times$ faster to train and requires $1.5\times$ less memory usage. To our best knowledge, **SingleProp** is the fastest and most efficient among SOTA certified training algorithms.

Ethical Impact

This work presents a method of efficiently training networks that are verifiable against adversarial attacks. Verifiably robust networks may be important in safety-critical scenarios such as self-driving cars or medical diagnosis where not only must models be robust to adversarial attack, but robustness must be verifiable to establish trust. Our method could help expand access to verifiable models in cases where training verifiable models is difficult due to computational constraints. This includes situations like image classifiers on real world images, where training high-accuracy, verifiable models may be out of reach due to computational cost. Our work presents potential benefits for users and creators of safety-critical models.

At the same time, it is possible that reliance on verification may provide a false sense of security to users who are not familiar with verification techniques. For instance, users may interpret verified model outputs as applying to a broader range of adversarial perturbations than the specific perturbation norm that is verified. As such, this work might potentially harm users less familiar with adversarial robustness and verification. We believe properly communicating the type and level of robustness implied by verification will be important in reducing the risks of our work.

References

Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 265–283.

Athalye, A.; Carlini, N.; and Wagner, D. 2018. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. *ICML* .

Athalye, A.; and Sutskever, I. 2017. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397* .

Boopathy, A.; Weng, T.-W.; Chen, P.-Y.; Liu, S.; and Daniel, L. 2019. CNN-Cert: An Efficient Framework for Certifying Robustness of Convolutional Neural Networks. In *AAAI*.

Carlini, N.; Katz, G.; Barrett, C.; and Dill, D. L. 2017. Provably Minimally-Distorted Adversarial Examples. *arXiv preprint arXiv:1709.10207* .

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

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, 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.; Gowal, S.; Stanforth, R.; Arandjelovic, R.; O'Donoghue, B.; Uesato, J.; and Kohli, P. 2018a. Training verified learners with learned verifiers. *arXiv preprint arXiv:1805.10265* .

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

Gehr, T.; Mirman, M.; Drachsler-Cohen, D.; Tsankov, P.; Chaudhuri, S.; and Vechev, M. 2018. AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation. In *IEEE Symposium on Security and Privacy (SP)*, volume 00, 948–963.

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

Gowal, S.; Dvijotham, K.; Stanforth, R.; Bunel, R.; Qin, C.; Uesato, J.; Arandjelovic, R.; Mann, T. A.; and Kohli, P. 2019. Scalable Verified Training for Provably Robust Image Classification. *ICCV* .

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

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.

Kingma, D.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Ko, C.-Y.; Lyu, Z.; Weng, T.-W.; Daniel, L.; Wong, N.; and Lin, D. 2019. POPQORN: Quantifying robustness of recurrent neural networks. *arXiv preprint arXiv:1905.07387* .

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

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. *ICLR* .

Mirman, M.; Gehr, T.; and Vechev, M. 2018. Differentiable abstract interpretation for provably robust neural networks. In *International Conference on Machine Learning*, 3575–3583.

Papernot, N.; McDaniel, P.; Jha, S.; Fredrikson, M.; Celik, Z. B.; and Swami, A. 2016. The limitations of deep learning in adversarial settings. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, 372–387.

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* .

Salman, H.; Yang, G.; Zhang, H.; Hsieh, C.-J.; and Zhang, P. 2019. A Convex Relaxation Barrier to Tight Robustness Verification of Neural Networks. *arXiv preprint arXiv:1902.08722* .

Singh, G.; Gehr, T.; Mirman, M.; Püschel, M.; and Vechev, M. 2018. Fast and Effective Robustness Certification. In *NeurIPS*.

Singh, G.; Gehr, T.; Mirman, M.; Püschel, M.; and Vechev, M. 2019. An Abstract Domain for Certifying Neural Networks. In *POPL*.

Sinha, A.; Namkoong, H.; and Duchi, J. 2018. Certifiable Distributional Robustness with Principled Adversarial Training. *ICLR* .

Su, J.; Vargas, D. V.; and Sakurai, K. 2019. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation* .

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* .

Tjeng, V.; and Tedrake, R. 2019. Verifying Neural Networks with Mixed Integer Programming. In *ICLR*.

Wang, S.; Pei, K.; Whitehouse, J.; Yang, J.; and Jana, S. 2018. Efficient Formal Safety Analysis of Neural Networks. In *NeurIPS*.

Weng, T.-W.; Zhang, H.; Chen, H.; Song, Z.; Hsieh, C.-J.; Boning, D.; Dhillon, I. S.; and Daniel, L. 2018. Towards Fast Computation of Certified Robustness for ReLU Networks. *ICML* .

Wong, E.; Schmidt, F.; Metzen, J. H.; and Kolter, J. Z. 2018. Scaling provable adversarial defenses. *arXiv preprint arXiv:1805.12514* .

Xiao, K. Y.; Tjeng, V.; Shafiullah, N. M.; and Madry, A. 2019. Training for Faster Adversarial Robustness Verification via Inducing ReLU Stability. In *ICLR*.

Xu, K.; Liu, S.; Zhao, P.; Chen, P.-Y.; Zhang, H.; Erdogmus, D.; Wang, Y.; and Lin, X. 2018. Structured adversarial attack: Towards general implementation and better interpretability. *arXiv preprint arXiv:1808.01664* .

Zhang, H.; Chen, H.; Xiao, C.; Li, B.; Boning, D. S.; and Hsieh, C. 2020. Towards Stable and Efficient Training of Verifiably Robust Neural Networks. *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*.

Zhang, H.; Yu, Y.; Jiao, J.; Xing, E. P.; El Ghaoui, L.; and Jordan, M. I. 2019. Theoretically Principled Trade-off between Robustness and Accuracy. In *ICML*.