DoubleMargin: Efficient Training of Robust and Verifiable Neural Networks

Anonymous authors
Paper under double-blind review

Abstract

Recent works have developed several methods of defending neural networks against adversarial attacks with certified guarantees. We propose that many common certified defenses can be viewed under a unified framework of regularization. This unified framework provides a technique for comparing different certified defenses with respect to robust generalization. In addition, we develop a new regularizer that is both more efficient than existing certified defenses and can be used to train networks with higher certified accuracy. Our regularizer also extends to an ℓ_0 threat model and ensemble models. Through experiments on MNIST, CIFAR-10 and GTSRB, we demonstrate improvements in training speed and 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 becomes 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 et al., 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 & Wagner, 2017; Athalye & Sutskever, 2017; Su et al., 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 et al., 2018). The work Madry et al. (2018) proposed a stronger defense method, adversarial training, which minimizes the *worst-case* training loss under adversarial perturbations. However, it is restricted to a specific type of adversarial perturbations and not generalized to other types of perturbations (Tramèr & Boneh, 2019).

Motivated by the limitation of heuristic defense, another line of research (known as verified/certified robustness) aims to provide *provable* robust guarantees of DNNs against an input with *arbitrary perturbation* within a certain ℓ_p ball region (Katz et al., 2017; Cheng et al., 2017; Carlini et al., 2017; Kolter & Wong, 2018; Raghunathan et al., 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., 2018; Mirman et al., 2018; Dvijotham et al., 2018a). The recent progress on verification spans from the exact verification method (Katz et al., 2017; Cheng et al., 2017; Carlini et al., 2017) to the approximate (relaxed) verification method (Kolter & Wong, 2018; Raghunathan et al., 2018; Weng et al., 2018; Zhang et al., 2018; Boopathy et al., 2019; Dvijotham et al., 2018b; Wong et al., 2018). Here the former uses expensive computation methods, e.g., mixed-integer programming (MIP), to find the exact robustness bound, and the latter considers a relaxed verification problem by convexifying the adversarial polytope but significantly improves the computation efficiency compared to the exact method. Besides resorting to approximate verification method, the recent work Xiao et al. (2019) proposed the principle of co-design between training and verification, and showed that the exact verification method can be accelerated by imposing weight sparsity and activation stability (so-called ReLU stability) on trainable network models. On the other hand, it was shown in Kolter & Wong (2018); Raghunathan et al. (2018); Gowal et al. (2018); Mirman et al. (2018); Dvijotham et al. (2018a) that by incorporating the relaxed but computationally efficient verification methods into the training process, the learnt model yields strengthened robustness with certificate. Following this line of research, in this work:

- We develop a unified framework of certified defense and propose that many common certified defenses can be incorporated as different types of regularization. We theoretically analyze general regularizations and provide guidelines on how to select regularization.
- We propose an efficient regularizer that better quantifies adversarial sensitivity during the training process to yield more certifiable models. Through experiments on six different architectures, we demonstrate our method's computational efficiency and higher certified accuracies on large perturbations compared to *all* current state-of-the-art certification based training methods. In particular, we demonstrate up to a **55%** increase in certified accuracy at $\epsilon = 0.2$ on MNIST, up to a **20.5%** increase at $\epsilon = 2/255$ on CIFAR, and up to a **24%** increase at $\epsilon = 8/255$ on GTSRB. In addition, on all architectures and datasets we achieve faster training than any other certification based training methods.
- We extend our regularizer to an ℓ_0 threat model and model ensembles. We demonstrate empirically that our regularizer achieves high certified accuracy on this threat model. We also empirically show that model ensembling of models trained with our regularizer achieve higher certified accuracies.

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 & Andriushchenko, 2017; Raghunathan et al., 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 & Wong, 2018; Weng et al., 2018; Wang et al., 2018), as well as networks with general activation functions (Zhang et al., 2018) and architectures (Singh et al., 2018). An "any-time" certifier has also been developed that allows for a trade-off between certification time and bound quality (Dvijotham et al., 2018b).

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 et al., 2018). However, adversarial training is not targeted towards verification or certification methods, and we therefore do not consider it a certified defense. One step towards certified defenses is natural regularization 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.

Other defenses specifically target certifiers or use certification methods as part of the training procedure. 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 certifiable models, although training is relatively slow (Kolter & Wong, 2018; Wong et al., 2018). Using interval bounds propagation (IBP) to bound the adversarial loss is much cheaper to train (Gowal et al., 2018) and has surprisingly become the state-of-the-art certifiably robust training method (Gowal et al., 2018; Salman et al., 2019) despite IBP performing much more poorly than the convex outer bounds (Kolter & Wong, 2018; Wong et al., 2018) in certifications. In this paper, we unify certified defenses under a unified framework of regularization. We also develop an efficient certified defense that has similar computation overhead as IBP-based defense (Gowal et al., 2018) while yielding better performance than existing methods.

2.3 Certification 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 B Table 3 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^* 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., 2018; 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 & 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: $\alpha^i \odot \mathbf{z}^i + \beta_L^i \leq \sigma(\mathbf{z}^i) \leq \alpha^i \odot \mathbf{z}^i + \beta_U^i$, where α^i represents the slopes of linear bounds on the activation and β_L^i,β_U^i representing intercepts of linear bounds on the activation. When σ is ReLU activation, Fast-Lin sets the coefficients to be:

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

if the neuron j is uncertain, meaning $\mathbf{z}_{l,j}^i < 0$, $\mathbf{z}_{u,j}^i > 0$ and $\mathbf{z}_{l,j}^i \leq \mathbf{z}_j^i \leq \mathbf{z}_{u,j}^i$. 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, α_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: $\mathbf{A}\mathbf{x} + \mathbf{b}_L \leq f(\mathbf{x}) \leq \mathbf{A}\mathbf{x} + \mathbf{b}_U$. Then Fast-Lin bounds the network output over all possible adversarial distortion measured by ϵ-ℓ_p ball by:

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

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 $\mathbf{z}_u, \mathbf{z}_l$, 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^i in CROWN (Zhang et al., 2018), and has been extended to general network architectures in CNN-Cert (Boopathy et al., 2019).

3 Unified Framework of Certified Defense

We propose a unified framework of certified defense under which many common certified defenses can be viewed as regularizations of a particular form. We first use our framework to provide analysis of

Training Method	$\mathcal{R}(\theta)$	Training cost
ℓ_2 Regularization	$\sum_i \|\|\mathbf{W}^i\|\|_F^2$ or $\prod_i \|\|\mathbf{W}^i\|\|_F$	$\times 1$
ℓ_1 Regularization	$\sum_i \|\|\mathbf{W}^i\|\|_{op,\infty}$ or $\prod_i \|\|\mathbf{W}^i\|\|_{op,\infty}$	$\times 1$
ReLU Stability Loss (Xiao et al., 2019)	$\mathbb{E}[-\sum_i \tanh(\mathbf{1} + \mathbf{u}^i \odot \mathbf{l}^i)]$	$\times 1$
Interval Bounds Propagation (Gowal et al., 2018)	$\mathbb{E}[L_+(\mathbf{u}^n) + L_-(\mathbf{l}^n) - L(\mathbf{z}^n)]$	$\times 3$
TRADES (Zhang et al., 2019)	$\mathbb{E}[\max_{\delta:\|\|\delta\|\|_p \le \epsilon} g(f(\mathbf{x}_{nom} + \delta), f(\mathbf{x}_{nom}))]$	> 10 (empirically)
Convex Outer Bounds (Kolter & Wong, 2018)	$\mathbb{E}[L_+(\mathbf{u}_c^n) + L_-(\mathbf{l}_c^n) - L(\mathbf{z}^n)]$	> 100 (empirically)
MMR (Croce et al., 2019)	$\mathbb{E}[max(0, 1 - \frac{d_B}{\gamma_B}) + max(0, 1 - \frac{\overline{d_D}}{\gamma_D})]$	> 10000 (empirically)
Double Margin	$\mathbb{E}[L_+(\mathbf{z}^n + \mathbf{s}^n + \mathbf{v}^n) + L_-(\mathbf{z}^n - \mathbf{s}^n - \mathbf{v}^n) - L(\mathbf{z}^n)]$	$\times \mathbf{3}$

Table 1: $\mathbf{u}^i$ and $\mathbf{l}^i$ are layer-wise bounds found using interval bounds propagation, and $\mathbf{u}_c^i$ and $\mathbf{l}_c^i$ are layer-wise bounds using convex outer bounds. $\odot$ denotes element-wise multiplication. $||\cdot||_{op,\infty}$ denote the ℓ_∞ induced operator norm. g is a classification-calibrated loss function. d_B and $\overline{d_D}$ are quantities relating to lower bounds on the minimum adversarial distortion. Expectations are taken over training data. The training cost is compared to the standard training.

general certified defenses. Under this framework, we then propose our regularizer which outperforms other current certified defense methods as suggested by the experiments. Because IBP has been shown to be the current state-of-the-art in efficiently training certifiable models, we consider it as our main baseline and highlight our method's advantages. Finally, we discuss extensions of our regularization.

3.1 Analysis of Regularizers

A Unified Regularizer Framework for Certified Defense Many different training methods can often be formulated as different regularizations added to a standard loss function. Given model parameters θ, a loss function $\mathcal{L}(\theta)$ and a regularizer $\mathcal{R}(\theta)$, the total regularized loss with a regularization parameter λ is given by:

$$\mathcal{L}(\theta) + \lambda\mathcal{R}(\theta)$$

Note that we denote a model's parameters in general as θ, and we use calligraphic symbols to denote functions of general parameters θ and standard symbols to denote functions varying at different data points. Standard regularizations chosen to enhance generalization include weight-based regularization where the regularizer is chosen to be $\sum_i ||\mathbf{W}_i||_l$ with l being some matrix norm. In fact, many certification-based robust training methods can also be viewed as different regularizations $R(\theta)$ as seen in Table 1. Therefore, both standard regularizations and many robust training methods can be viewed under a unified framework. Training methods can be designed by selecting a particular regularizer according to the designer's objective.

Regularizer Construction Certified defenses such as IBP construct a regularizer by using bounds on the last layer of the network $\mathbf{u}^n$ and $\mathbf{l}^n$ to find an upper bound on the adversarial loss. Suppose a standard loss function $\mathcal{L}(\theta)$ of model parameters θ is defined as a function of the last layer of the network $\mathbf{z}^n$: $\mathcal{L}(\theta) = \mathbb{E}[L(\mathbf{z}^n)]$, where the expectation is taken over the training set. For many loss functions L, including softmax cross-entropy loss and squared loss, it is possible to decompose the loss into an increasing and decreasing component:

$$L(\mathbf{z}^n) = L_+(\mathbf{z}^n) + L_-(\mathbf{z}^n), \tag{3}$$

where $L_+(\mathbf{z}^n)$ increases element-wise with $\mathbf{z}^n$ and $L_-(\mathbf{z}^n)$ decreases element-wise with $\mathbf{z}^n$. For example in the case of softmax cross-entropy loss:

$$L(\mathbf{z}^n) = -\mathbf{y}^T \log\left(e^{\mathbf{z}^n}/(\mathbf{1}^T e^{\mathbf{z}^n})\right) = \log \mathbf{1}^T e^{\mathbf{z}^n} - \mathbf{y}^T\mathbf{z}^n,$$

where $e^{\mathbf{x}}$ is element-wise exponential function and $\mathbf{y}$ are one-hot encoded labels, the first term can be mapped to $L_+(\mathbf{z}^n)$ while the second corresponds to $L_-(\mathbf{z}^n)$ in (3). IBP then constructs the following regularizer: $\mathcal{R}(\theta) = \mathbb{E}[L_+(\mathbf{u}^n) + L_-(\mathbf{l}^n) - L(\mathbf{z}^n_{nom})]$, where $\mathbf{u}^n$ and $\mathbf{l}^n$ are defined by (2) and $\mathbf{z}^n_{nom}$ is the value of layer $\mathbf{z}^n$ with unperturbed input $\mathbf{x}_{nom}$. When this regularizer is added to the standard loss function $\mathcal{L}(\theta)$ computed with unperturbed input $\mathbf{x}_{nom}$, the resulting quantity $\mathcal{L}(\theta) + \mathcal{R}(\theta)$ is an upper bound on the adversarial loss since: $L(\mathbf{z}^n_{nom}) + (L_+(\mathbf{u}^n) + L_-(\mathbf{l}^n) - L(\mathbf{z}^n_{nom})) \ge L(\mathbf{z}^n)$, where $\mathbf{z}^n$ is a function of perturbed input $\mathbf{x}$, obtained by monotonicity of L_+ and L_- and $\mathbf{l}^n \le \mathbf{z}^n \le \mathbf{u}^n$.

Note that the adversarial loss $L(\mathbf{z}^n)$ is unknown, but the upper bound constructed here depends only on the IBP bounds $\mathbf{u}^n$ and $\mathbf{l}^n$ which are known. Typically, the model is trained with objective $\mathcal{L}(\theta) + \lambda\mathcal{R}(\theta)$ with $\lambda < 1$ instead of $\lambda = 1$ in order to ensure stable training.

Connecting regularization to robustness and generalization A regularized loss function often can be seen as an upper bound on a particular robustness or generalization objective. As shown previously, adding a robust training regularizer such as IBP is equivalent to minimizing an upper bound on the adversarial loss: $\frac{1}{m}\sum_i \max_{\mathbf{x}:||\mathbf{x}-\mathbf{x}_i||_p\le\epsilon} L(f(\mathbf{x})) \le \mathcal{L}(\theta) + \lambda\mathcal{R}(\theta)$, for sufficiently large λ where there are m points $\mathbf{x}_i$ in the training set. Similarly ℓ_1 and ℓ_2 regularization can be seen as minimizing a probabilistic upper bound on the test set loss: $\mathcal{L}_{test}(\theta) \le \mathcal{L}(\theta) + \lambda\mathcal{R}(\theta)$. This is because many generalization error bounds (Bartlett & Mendelson, 2002; Neyshabur et al., 2015) are statements of the following form: with high probability,

$$\mathcal{L}_{test}(\theta) \le \mathcal{L}(\theta) + \frac{K\prod_j ||\mathbf{W}^j|| + C}{\sqrt{m}}, \tag{4}$$

where $\mathcal{L}_{test}(\theta)$ is the test set error, $||\cdot||$ is either a Frobenius or ℓ_∞ operator norm, m represents the number of training set points, and K and C are constants depending on the network architecture. It is possible to analyze the effect of general regularizations under the assumption that λ is close to zero. This corresponds to ϵ close to zero for robustness-based regularizers or m very large for generalization-based regularizers such as ℓ_1 or ℓ_2 weight regularization that can be interpreted as minimizing a generalization bound of the form in Equation 4.

Proposition 1. *Given a classifier with parameters θ and loss $\mathcal{L}(\theta)$ with a local minimum $\hat{\theta}$, then for λ near 0, the local minimum of $\mathcal{L}(\theta) + \lambda\mathcal{R}(\theta)$ is approximately:*

$$\hat{\theta} - \lambda H(\mathcal{L}(\hat{\theta}))^{-1}\nabla_\theta\mathcal{R}(\hat{\theta}), \tag{5}$$

where $H(\mathcal{L}(\hat{\theta}))$ is the Hessian of the loss at the original local minimum $\hat{\theta}$.

This statement provides a method of relating a regularizer to its effect on the classifier. Please refer to Appendix E for the proof and additional intuition.

Proposition 2. *Given a classifier with parameters θ and loss $\mathcal{L}(\theta)$ with a local minimum $\hat{\theta}$. For λ near 0, suppose θ^* is the corresponding local minimum of a regularized loss $\mathcal{L}(\theta) + \lambda\mathcal{R}_1(\theta)$. Then, for λ near 0, $\mathcal{L}(\theta^*) + \lambda\mathcal{R}_2(\theta^*) < \mathcal{L}(\hat{\theta}) + \lambda\mathcal{R}_2(\hat{\theta})$ if and only if:*

$$||\nabla_\theta\mathcal{R}_1(\hat{\theta}) - \nabla_\theta\mathcal{R}_2(\hat{\theta})||_{H(\mathcal{L}(\hat{\theta}))^{-1}} < ||\nabla_\theta\mathcal{R}_2(\hat{\theta})||_{H(\mathcal{L}(\hat{\theta}))^{-1}}, \tag{6}$$

where $||a||_B = \sqrt{a^TBa}$.

This statement can be used to provide guidelines on regularization selection. Intuitively, the condition implies that selecting regularizers with close gradient to a particular "optimal" regularizer $\mathcal{R}_2(\theta^*)$ is advantageous where distance is defined using the left hand side of the Equation 6. Please refer to Appendix F for the proof and additional intuition.

3.2 Our Regularizer

Double Margin: Motivation and Rationale IBP is one way to propagate the adversarial sensitivity at different layers of the neural network and construct a robust loss function, but there exist other methods of quantifying sensitivity in the network. Different from IBP that bounds the input of each layer $\mathbf{z}$ through $\mathbf{u}$ and $\mathbf{l}$ in (2), we introduce *double margins* $\mathbf{s}$ and $\mathbf{v}$ such that $[\mathbf{z}_{nom} - \mathbf{s} - \mathbf{v}, \mathbf{z}_{nom} + \mathbf{s} + \mathbf{v}]$ represents the approximate range of values each layer $\mathbf{z}$ can take, where $\mathbf{z}_{nom}$ represents the value of the layer for unperturbed input $\mathbf{x}_{nom}$. In the similar spirit of IBP, we define the $\mathbf{s}$ term to approximate error margins at each layer of the network, but this margin $\mathbf{s}$ only requires propagation of one quantity while IBP requires two quantities $\mathbf{u}$ and $\mathbf{l}$.

Existing certifiers such as Fast-Lin (Weng et al., 2018) and Reluplex (Katz et al., 2017) rely on activations to be locally linear in order to achieve tight bounds. However, using the $\mathbf{s}$ term by itself does not necessarily result in local linearity at activations. Thus, we also include the $\mathbf{v}$ term which penalizes a finite difference approximation of the second derivative of the activation function. This margin has the property that when the network is locally linear in the sense that $\sigma(\mathbf{z}_j^i) \approx a\mathbf{z}_j^i + b$ for some a, b at all layers i, all margins $\mathbf{v}^i \approx 0$. Intuitively, the $\mathbf{s}$ term can be seen as measuring a 1st order, linear level of adversarial sensitivity, while the $\mathbf{v}$ term measures 2nd order, nonlinear sensitivity (see Appendix C Figure 2 for a visual illustration). The inclusion of the $\mathbf{v}$ margin is the key qualitative difference between our approach and IBP. Because our approach explicitly penalizes non-linearity, it results in tighter certification and therefore more certifiable networks compared to IBP.

Double Margin: Formal Definition Assuming a known ℓ_p perturbation size ϵ, our margins $\mathbf{s}^i$ and $\mathbf{v}^i$ at layer i have the same dimensionality as $\mathbf{z}^i$, whether they are vectors in fully connected layers or tensors in convolutional layers. Only the margins at the second layer depend on p.

These margins are initialized as: $\mathbf{s}^0 = \epsilon\mathbf{1}, \mathbf{v}^0 = \mathbf{0}$. For $i = 0, \ldots, n-1$, subsequent layer margins are defined as:

$$\mathbf{s}^{i+1} = \frac{1}{2}|\mathbf{W}^{i+1}|\left(\sigma(\mathbf{z}_{nom}^i + \mathbf{s}^i) - \sigma(\mathbf{z}_{nom}^i - \mathbf{s}^i)\right) \tag{7}$$

$$\begin{aligned}\mathbf{v}^{i+1} &= \frac{1}{2}|\mathbf{W}^{i+1}|\left(\sigma(\mathbf{z}_{nom}^i + \mathbf{v}^i) - \sigma(\mathbf{z}_{nom}^i - \mathbf{v}^i)\right) \\ &\quad + \frac{1}{2}|\mathbf{W}^{i+1}||\sigma(\mathbf{z}_{nom}^i + \mathbf{s}^i + \mathbf{v}^i) + \sigma(\mathbf{z}_{nom}^i - \mathbf{s}^i - \mathbf{v}^i) - 2\sigma(\mathbf{z}_{nom}^i)|\end{aligned} \tag{8}$$

For general ℓ_p norms $p \neq \infty$, the second layer margins are modified so that $\mathbf{s}^1$ is exactly half the range of values $\mathbf{z}^1$ can take. To motivate the form of Equation (7), note that IBP can be reparameterized in terms of the half bound gap $\mathbf{h} = (\mathbf{u} - \mathbf{l})/2$ and bound average $\mathbf{m} = (\mathbf{u} + \mathbf{l})/2$. Then the half bound gap $\mathbf{h}$ is propagated as:

$$\mathbf{h}^{i+1} = \frac{1}{2}|\mathbf{W}^{i+1}|(\sigma(\mathbf{m}^i + \mathbf{h}^i) - \sigma(\mathbf{m}^i - \mathbf{h}^i))$$

The $\mathbf{s}$ margin is propagated similarly, with the difference being that IBP uses $(\mathbf{u}^i + \mathbf{l}^i)/2$ instead of $\mathbf{z}^i$ as the centering point. This supports the intuition that the $\mathbf{s}$ margin by itself approximates the IBP bounds. The $\mathbf{v}$ margin is propagated the same way, with an additional term that uses a finite difference approximation of the second derivative of $\sigma(\cdot)$ at $\mathbf{z}^i$. Specifically, note that given a 2nd order Taylor expansion of $\sigma(\cdot)$ at a specific scalar point x_0, $\sigma(x - x_0) \approx \frac{1}{2}a(x - x_0)^2 + b(x - x_0) + c$, then the second derivative of $\sigma(\cdot)$ at x_0 is given by: $\sigma''(x_0) = a \approx \frac{1}{\gamma^2}(\sigma(x_0 + \gamma) + \sigma(x_0 - \gamma) - 2\sigma(x_0))$, where $\gamma > 0$ is small. Applying this approximation to the second term in Equation (8):

$$\frac{1}{2}|\mathbf{W}^{i+1}||\sigma(\mathbf{z}_{nom}^i + \mathbf{s}^i + \mathbf{v}^i) + \sigma(\mathbf{z}_{nom}^i - \mathbf{s}^i - \mathbf{v}^i) - 2\sigma(\mathbf{z}^i)| \approx \frac{1}{2}|\mathbf{W}^{i+1}||\sigma''(\mathbf{z}_{nom}^i) \odot (\mathbf{s}^i + \mathbf{v}^i)^2|,$$

where all operators are applied element-wise. By approximating the second derivative of the activation, this term penalizes a level of non-linearity. To motivate our use of a finite difference instead of a second derivative directly, note that in the case of ReLU, using a second derivative would not work since ReLU has second derivative zero almost everywhere. A finite difference approximation also quantifies nonlinearity over a larger neighborhood; see Eq. 1 in Moosavi-Dezfooli et al. (2018) for a similar justification. Because we add two additional margins $\mathbf{s}$ and $\mathbf{v}$ in addition to the original network, we call our method double margin. Although the margins are defined for a fully connected network, the margins can be computed analogously for special cases or extensions such as convolutional neural networks or residual networks. Our regularizer is defined implicitly as a function of the model parameters θ:

$$\mathcal{R}(\theta) = \mathbb{E}[L_+(\mathbf{z}_{nom}^n + \mathbf{s}^n + \mathbf{v}^n) + L_-(\mathbf{z}_{nom}^n - \mathbf{s}^n - \mathbf{v}^n) - L(\mathbf{z}_{nom}^n)] \tag{9}$$

We note that our regularizer when added to the standard loss function is not a true upper bound on the adversarial loss since the margins do not provide exact bounds on each layer. However, as shown by Proposition 2, the *gradient* of a regularizer rather than its bound validity determines its certified test loss. Therefore, and as we also show experimentally, using an upper bound on the adversarial loss is not necessary to train certifiable models.

Training Procedure Training proceeds by using standard optimizers on the regularized loss. See Appendix D Algorithm 1 for the full procedure. Note that using a value of $\lambda = 0$ corresponds to standard training while using $\lambda = 1$ corresponds to using only the regularizer (i.e. robust loss). In practice, robust training methods such as IBP typically increase the value of λ during training process. IBP specifically uses a warm-up period where $\lambda = 0$ first, then linearly increases λ until it reaches $\lambda = 0.5$. Note 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 $\epsilon = 0$ also 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. IBP uses a piece-wise linear schedule for ϵ with warm-up and ramp-up periods coinciding with the schedule for λ. We use a similar piece-wise linear schedule as IBP.

3.3 Extensions

We extend our method to an ℓ_0 threat model by modifying the double margin propagation at the second layer as done for general ℓ_p perturbations. We also extend our method to ensembles of models trained with our regularizer. Additional details are provided in Appendix H.

4 Experiments

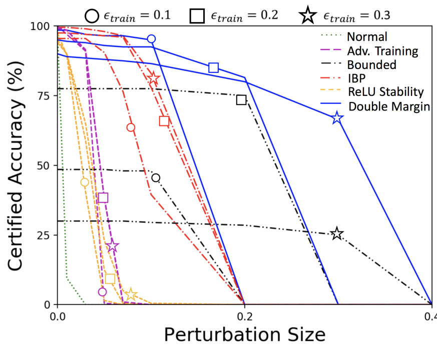
We conduct experiments comparing our regularizer to other robust training methods including ℓ_1 weight regularization, adversarial training, IBP, ReLU stability and convex outer bounds (Bounded).

4.1 Setup, Models, Dataset

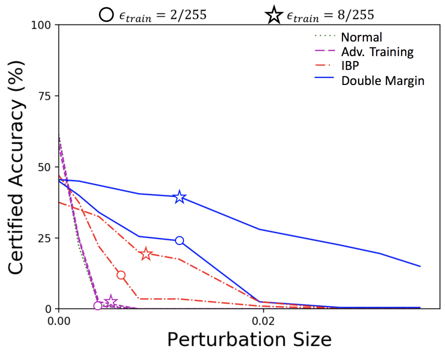
Implementation, Architectures, Training Parameters Training methods are implemented in Python with Tensorflow (Abadi et al., 2016) and training is conducted on a NVIDIA Tesla P100 GPU. We evaluate the training methods on networks trained on the MNIST, CIFAR-10 and GTSRB (Stallkamp et al., 2012) datasets. We consider 6 model architectures listed roughly in increasing model size: (i) Small CNN, (ii) Pureconv CNN, (iii) Resnet CNN, (iv) Pooling CNN, (v) Medium CNN, (vi) Large CNN. For a fair comparison among training methods, we used fixed training parameters. Architecture and hyperparameter details are deferred to Appendix G.

Comparative Methods We train networks with (i) normal training, (ii) ℓ_1 weight regularization, (iii) adversarial training, (iv) IBP, (v) ReLU stability, (vi) Bounded, (vii) MMR (Croce et al., 2019), (viii) TRADES (Zhang et al., 2019) and (ix) our method (Double Margin). Due to the computational cost of running Bounded training, we train this method only on the small CNN architecture. For training Bounded models, we use the technique of random projections as proposed by (Wong et al., 2018), using 50 random projections. For the small CNN architecture, we try additional robust training methods including (x) combining adversarial training with the ReLU stability loss and weight pruning (Adv+RS+Pruning), and (xi) combining ℓ_1 weight regularization with ReLU stability and weight pruning (L1+RS+Pruning). We also (xii) add ℓ_1 weight regularization to our regularizer (Double Margin + L1).

Evaluation Although our regularizer applies to a general ℓ_p threat model, we primarily conduct experiments assuming a ℓ_∞ threat model. The networks are evaluated on certified accuracy computed using CNN-Cert-ReLU, which efficiently finds a lower bound on the worst-case adversarial accuracy under norm bounded perturbations. We use CNN-Cert due to the high computational cost of using other methods such as Mixed Integer Optimization (MIO) (Tjeng & Tedrake, 2019), even with using a optimization timeout as done by Gowal et al. (2018). In particular, we found that the initialization stage of the MIO verifier itself typically took longer than CNN-Cert. Baseline certified accuracies are generally higher in prior works because they are computed using slower certifiers. Certified accuracies are computed for 200 test set points (unless otherwise noted) over a range of perturbation sizes ϵ in $[0, 0.4]$ for MNIST and $[0, 9/255]$ for CIFAR and GTSRB. In addition, for selected networks we compute certified accuracies using IBP. For all certified accuracy comparisons, we use the same randomly selected set of test points. For two selected networks, we also compute certified accuracies on the entire test set and verify that the certified accuracies are similar to ensure that our randomly chosen test points are representative (see Appendix I).



(a) MNIST



(b) CIFAR

Figure 1: CNN-Cert certified accuracy against perturbation size for models trained normally and with several robust training methods including Bounded, IBP and our method, Double Margin. Results are shown for different values of ϵ_{train}. (a) shows results on MNIST and (b) on CIFAR. Refer to Table 2 for additional baselines including MMR and TRADES.

4.2 Results

Baselines Comparison In all CNN-Cert comparison tables, the best methods in each column are highlighted, excluding normal training and standard weight regularization. Table 2 provides a summary of the comparative experiments. We first highlight that over all tested datasets, our training methods train in less time than IBP, Bounded, TRADES and MMR. On MNIST, our methods, in bold, outperform ReLU stability based methods and adversarial training for $\epsilon = 0.01$ and larger, and outperform IBP and Bounded for large perturbation sizes beyond $\epsilon = 0.07$. We note that adding ℓ_1 weight regularization to our regularizer further enhances our certified accuracies. On CIFAR, our method outperforms ReLU stability based methods and adversarial training for $\epsilon = 1/255$ and larger, and outperforms IBP and Bounded for perturbation sizes beyond $\epsilon = 0.5/255$. On MNIST and CIFAR, we observe that due to high hyperparameter sensitivity, Bounded and ReLU stability sometimes have low clean accuracy and TRADES achieves low certified accuracy. Certified accuracy results for selected small MNIST and CIFAR networks are also presented visually in Figure 1. On GTSRB, our method outperforms IBP and MMR for $\epsilon = 3/255$ and larger. Additional results on MNIST, CIFAR and GTSRB are shown in Appendix J, Tables 8, 9 and 6 respectively. **In brief, all the experiments show that our method achieves higher CNN-Cert accuracy for large ϵ than any prior work.**

Ablation Experiments and Hyperparameter Analysis In addition, we separately analyze the effect of s and v margins in Double Margin by running the following comparisons: training with only the s margin (Variation 1: 's only'), and training Double Margin with IBP bounds u, l replacing s (Variation 2: 'IBP+v'). Table 2 shows that Variation 1 performs comparably to or outperforms IBP while requiring only fewer propagation during training, supporting the intuition that the s margin by itself approximates IBP. Variation 2 performs comparably to Double Margin but it requires one additional propagation during training (10-40% increased training time). These observations suggest that the 'v' margin is primarily responsible for Double Margin's improved certifiable robustness for large perturbations, but at the expense of lower clean accuracy compared to IBP. In order to improve the clean accuracy of networks trained with our method, we also train under a different hyperparameter setting where λ is halved during training, reducing its final value from $\frac{1}{2}$ to $\frac{1}{4}$. As seen in Table 2 Double Margin still outperforms IBP on the largest ϵ ($\mathbf{60.4}$ vs $\mathbf{0.0}$% on MNIST and $\mathbf{46.4}$ vs $\mathbf{26.8}$% on GTSRB) while achieving a higher clean accuracy ($\mathbf{91.1}$% on MNIST and $\mathbf{75.6}$% on GTSRB). These results suggest that fine-tuning hyperparameters can improve clean accuracy to be more comparable to prior work while maintaining our better certified accuracy on large ϵ.

Extensions We find that our method's high certified accuracy extends to an ℓ_0 norm threat model. In addition, we find that using model ensembles with our method can be used to enhance certified accuracy. Results are shown in Appendix I.

5 Conclusion

In this paper, we propose a unified framework of certified defense as regularization. We introduce a new efficient regularizer under this framework, and extend it to a ℓ_0 threat model and model ensembles. Through comparative experiments on several model architectures, we demonstrate that our method outperforms state-of-the-art certification based training methods in terms of training time and certified accuracy.

Method (200 points)		Training Time (sec)	$\epsilon_{cert} = 0$	0.01	0.03	0.05	0.07	0.10	0.20	0.30	0.40
			Small CNN MNIST, 4 layer								
Normal		185.7	98.5%	9.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Weight Regularization		626.3	99.5%	81.0%	1.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Adv. Training	$\epsilon = 0.3$	8760.5	**99.5%**	**98.5%**	91.5%	38.5%	1.0%	0.0%	0.0%	0.0%	0.0%
Bounded	$\epsilon = 0.3$	27966.8	30.0%	30.0%	30.0%	30.0%	30.0%	29.5%	28.5%	25.0%	0.0%
IBP	$\epsilon = 0.3$	611.2	95.5%	95.5%	**95.5%**	**90.5%**	76.5%	39.5%	0.0%	0.0%	0.0%
ReLU Stabilty	$\epsilon = 0.3$	595.8	94.0%	90.0%	65.0%	23.0%	5.0%	0.5%	0.0%	0.0%	0.0%
L1+RS+Pruning	$\epsilon = 0.3$	686.5	95.0%	88.0%	53.5%	13.5%	3.0%	0.5%	0.0%	0.0%	0.0%
Adv+RS+Pruning	$\epsilon = 0.3$	9657.1	90.0%	86.5%	70.5%	55.0%	30.5%	13.5%	0.0%	0.0%	0.0%
Double Margin	$\epsilon = 0.3$	**520.0**	90.0%	89.0%	88.5%	88.0%	87.5%	86.5%	80.0%	66.5%	0.0%
Double Margin +L1	$\epsilon = 0.3$	547.1	91.0%	90.5%	90.0%	89.5%	**89.0%**	**87.5%**	**83.5%**	**68.0%**	0.0%
Method (200 points)		Training Time (sec)	$\epsilon_{cert} = 0$	0.5/255	1/255	2/255	3/255	5/255	7/255	8/255	9/255
			Small CNN CIFAR, 4 layer								
Normal		141.1	61.5%	22.0%	1.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Weight Regularization		1062.8	58.0%	24.0%	2.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Adv. Training	$\epsilon = 8/255$	51893.6	**60.5%**	25.0%	2.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Bounded	$\epsilon = 8/255$	21918.7	4.0%	4.0%	4.0%	4.0%	4.0%	4.0%	3.5%	3.5%	3.5%
IBP	$\epsilon = 8/255$	410.0	37.5%	35.0%	32.5%	20.0%	17.5%	2.5%	0.0%	0.0%	0.0%
ReLU Stabilty	$\epsilon = 8/255$	**396.1**	12.0%	12.0%	12.0%	12.0%	12.0%	12.0%	12.0%	12.0%	12.0%
Double Margin	$\epsilon = 8/255$	409.7	45.5%	**45.0%**	**43.5%**	**40.5%**	**39.5%**	**28.0%**	**22.5%**	**19.5%**	**15.0%**
			Pooling CNN CIFAR, 5 layer								
Normal		1164.9	65.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Weight Regularization		828.8	59.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
IBP	$\epsilon = 8/255$	4015.1	43.0%	15.5%	8.5%	3.0%	3.0%	0.0%	0.0%	0.0%	0.0%
ReLU Stabilty	$\epsilon = 8/255$	**1676.9**	**58.5%**	8.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Double Margin	$\epsilon = 8/255$	2899.5	40.0%	**32.5%**	**30.5%**	**24.0%**	**23.5%**	**5.0%**	**1.0%**	**1.0%**	**0.5%**
			Pureconv CNN GTSRB, 4 layer								
Normal		119.6	92.0%	77.5%	57.5%	37.0%	35.0%	6.5%	1.0%	0.0%	0.0%
IBP	$\epsilon = 8/255$	287.7	64.0%	63.0%	62.0%	58.5%	57.0%	44.5%	31.5%	26.0%	22.0%
Double Margin	$\epsilon = 8/255$	**283.1**	**68.0%**	**67.5%**	**67.0%**	**65.0%**	**64.5%**	**58.0%**	**52.5%**	**50.0%**	**45.5%**

Method (1000 points)		Training Time (sec)	$\epsilon_{cert} = 0$	0.05	0.10	0.20	0.30
			Small CNN MNIST, 4 layer				
TRADES	$\epsilon = 0.3$	8264.6	**99.5 (0.7)%**	0.0 (0.0)%	0.0 (0.0)%	0.0 (0.0)%	0.0 (0.0)%
IBP	$\epsilon = 0.3$	611.2	96.3 (1.7)%	92.2 (2.5)%	46.5 (5.1)%	0.0 (0.0)%	0.0 (0.0)%
IBP *	$\epsilon = 0.3$	460.9	96.7 (1.4)%	90.3 (2.6)%	39.3 (4.6)%	0.0 (0.0)%	0.0 (0.0)%
DM Variation 1	$\epsilon = 0.3$	**273.3**	94.0 (2.1)%	**92.7 (2.5)%**	**90.1 (3.0)%**	78.0 (3.9)%	33.1 (4.7)%
DM Variation 2	$\epsilon = 0.3$	589.6	90.1 (2.5)%	88.9 (3.1)%	86.5 (3.2)%	**79.3 (3.7)%**	59.0 (5.0)%
Double Margin	$\epsilon = 0.3$	520.0	88.2 (2.6)%	86.7 (3.1)%	84.6 (3.4)%	76.2 (3.9)%	**61.9 (4.6)%**
Double Margin *	$\epsilon = 0.3$	415.3	91.1 (2.6)%	90.1 (2.9)%	87.6 (3.2)%	78.4 (3.9)%	60.4 (4.7)%
Method (500 points)		Training Time (sec)	$\epsilon_{cert} = 0$	1/255	3/255	5/255	8/255
			Small CNN CIFAR, 4 layer				
TRADES	$\epsilon = 8/255$	56391.6	**62.2 (4.4)%**	7.2 (2.5)%	0.0 (0.0)%	0.0 (0.0)%	0.0 (0.0)%
IBP	$\epsilon = 8/255$	2446.6	38.0 (4.2)%	32.8 (4.0)%	14.6 (3.1)%	6.0 (2.2)%	1.2 (0.8)%
DM Variation 1	$\epsilon = 8/255$	**1475.8**	45.0 (5.1)%	**41.0 (3.8)%**	33.8 (4.4)%	22.8 (3.9)%	10.4 (2.6)%
DM Variation 2	$\epsilon = 8/255$	3062.8	43.0 (4.6)%	40.6 (4.2)%	**34.2** (4.0)%	27.0 (4.1)%	18.2 (3.2)%
Double Margin	$\epsilon = 8/255$	2198.7	43.2 (4.0)%	40.4 (4.6)%	32.8 (4.4)%	**28.6 (4.4)%**	**19.8 (3.5)%**
			Pureconv CNN GTSRB, 4 layer				
MMR (4 epochs)	$\epsilon = 8/255$	85377.2	**88.2 (2.8)%**	53.6 (4.0)%	18.2 (3.7)%	4.2 (1.8)%	0.0 (0.0)%
IBP	$\epsilon = 8/255$	287.7	71.8 (3.9)%	68.0 (4.3)%	60.8 (4.3)%	47.6 (4.7)%	28.2 (3.5)%
IBP *	$\epsilon = 8/255$	266.6	78.0 (3.7)%	**74.8 (3.7)%**	62.2 (4.3)%	45.6 (4.7)%	26.8 (3.5)%
Double Margin	$\epsilon = 8/255$	283.1	67.8 (4.4)%	66.8 (4.3)%	61.4 (4.3)%	56.4 (4.7)%	**48.2 (4.5)%**
Double Margin *	$\epsilon = 8/255$	**258.1**	75.6 (3.9)%	73.8 (4.2)%	**65.0 (4.5)%**	**57.2 (4.4)%**	46.4 (4.4)%

Table 2: CNN-Cert certified accuracies and training times. std in $(\cdot)$. * denotes a different setting of regularization parameter λ, decreased from $\frac{1}{2}$ to $\frac{1}{4}$. MMR is trained for 4 epochs due to its long training time. Our methods or variants are in blue.

REFERENCES

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. 2016.

Anish Athalye and Ilya Sutskever. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397*, 2017.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *ICML*, 2018.

Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *JMLR*, 2002.

Akhilan Boopathy, Tsui-Wei Weng, Pin-Yu Chen, Sijia Liu, and Luca Daniel. Cnn-cert: An efficient framework for certifying robustness of convolutional neural networks. In *AAAI*, Jan 2019.

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

Nicholas Carlini, Guy Katz, Clark Barrett, and David L Dill. Provably minimally-distorted adversarial examples. *arXiv preprint arXiv:1709.10207*, 2017.

Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Ead: elastic-net attacks to deep neural networks via adversarial examples. *AAAI*, 2018.

Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess. Maximum resilience of artificial neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pp. 251–268. Springer, 2017.

Francesco Croce, Maksym Andriushchenko, and Matthias Hein. Provable robustness of relu networks via maximization of linear regions. In *AISTATS*, 2019.

Krishnamurthy Dvijotham, Sven Gowal, Robert Stanforth, Relja Arandjelovic, Brendan O'Donoghue, Jonathan Uesato, and Pushmeet Kohli. Training verified learners with learned verifiers. *arXiv preprint arXiv:1805.10265*, 2018a.

Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy Mann, and Pushmeet Kohli. A dual approach to scalable verification of deep networks. *UAI*, 2018b.

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

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *ICLR*, 2015.

Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy A. Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.

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

Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pp. 97–117. Springer, 2017.

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

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *ICLR*, 2018.

Matthew Mirman, Timon Gehr, and Martin Vechev. Differentiable abstract interpretation for provably robust neural networks. In *International Conference on Machine Learning*, pp. 3575–3583, 2018.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Jonathan Uesato, and Pascal Frossard. Robustness via curvature regularization, and vice versa. In *CVPR*, 2018.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. *JMLR*, 2015.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 372–387, 2016.

Jonathan Peck, Joris Roels, Bart Goossens, and Yvan Saeys. Lower bounds on the robustness to adversarial perturbations. In *NIPS*, 2017.

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *ICLR*, 2018.

Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. A Convex Relaxation Barrier to Tight Robustness Verification of Neural Networks. *arXiv preprint arXiv:1902.08722*, 2019.

Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. Fast and effective robustness certification. In *NeurIPS*, 2018.

Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. An abstract domain for certifying neural networks. In *POPL*, 2019.

Aman Sinha, Hongseok Namkoong, and John Duchi. Certifiable distributional robustness with principled adversarial training. *ICLR*, 2018.

J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 2012.

Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 2019.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Vincent Tjeng and Russ Tedrake. Verifying neural networks with mixed integer programming. In *ICLR*, 2019.

Florian Tramèr and Dan Boneh. Adversarial training and robustness for multiple perturbations. *arXiv preprint arXiv:1904.13000*, 2019.

Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Efficient formal safety analysis of neural networks. In *NeurIPS*, 2018.

Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Duane Boning, Inderjit S Dhillon, and Luca Daniel. Towards fast computation of certified robustness for relu networks. *ICML*, 2018.

Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J Zico Kolter. Scaling provable adversarial defenses. *arXiv preprint arXiv:1805.12514*, 2018.

Kai Y. Xiao, Vincent Tjeng, Nur Muhammad Shafiullah, and Aleksander Madry. Training for faster adversarial robustness verification via inducing relu stability. In *ICLR*, 2019.

Kaidi Xu, Sijia Liu, Pu Zhao, Pin-Yu Chen, Huan Zhang, Deniz Erdogmus, Yanzhi Wang, and Xue Lin. Structured adversarial attack: Towards general implementation and better interpretability. *arXiv preprint arXiv:1808.01664*, 2018.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically Principled Trade-off between Robustness and Accuracy. In *ICML*, 2019.

Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In *NIPS*, dec 2018.

APPENDIX

A APPENDIX OUTLINE

B TABLE OF NOTATION

Notation	**Definition**	**Notation**	**Definition**
$\mathbf{x}_{nom}$	unperturbed input	$\mathbf{W}^i, \mathbf{b}^i$	weights, biases of layer i
δ	input perturbation	$\mathbf{z}^i$	input of layer i
ϵ	maximum ℓ_p perturbation size $\|\|\delta\|\|_p$	$\mathbf{u}^i, \mathbf{l}^i$	pre-activation interval bounds
$\mathbf{x}$	perturbed input	$\mathbf{s}^i, \mathbf{v}^i$	double margins
f	neural network classifier	$\alpha^i, \beta_U^i, \beta_L^i$	linear bounding parameters
σ	neural network activation	$\mathbf{A}, \mathbf{b}_L, \mathbf{b}_U$	linear network bounds
n	number of network layers	$\mathcal{L} : \theta \to \mathbb{R}$	training loss function
m	number of training points	$\mathcal{L}_{test} : \theta \to \mathbb{R}$	test loss function
$L : \mathbf{z}^n \to \mathbb{R}$	point-wise loss function	$\mathcal{R} : \theta \to \mathbb{R}$	regularizer
θ	model parameters	λ	regularizer coefficient

Table 3: Table of Notation

C Double Margin Visual Illustration

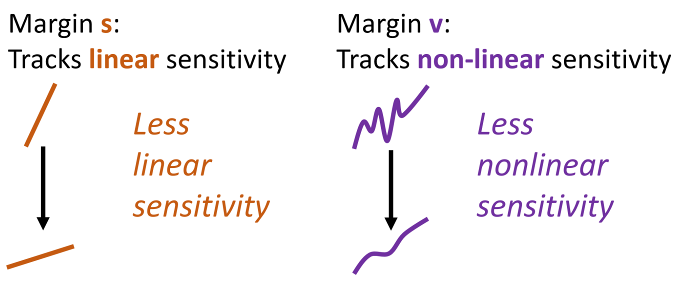

Margin s:
Tracks linear sensitivity
Less linear sensitivity
Margin v:
Tracks non-linear sensitivity
Less nonlinear sensitivity

Figure 2: A visual illustration of Double Margin's sensitivity quantification. $\mathbf{s}$ is propagated as $\mathbf{s} = \frac{1}{2}|\mathbf{W}|\left(\sigma(\mathbf{z}+\mathbf{s}) - \sigma(\mathbf{z}-\mathbf{s})\right)$. $\mathbf{v}$ is propagated as $\mathbf{v} = \frac{1}{2}|\mathbf{W}|\left(\sigma(\mathbf{z}+\mathbf{v}) - \sigma(\mathbf{z}-\mathbf{v})\right) + \frac{1}{2}|\mathbf{W}||\sigma(\mathbf{z}+\mathbf{s}+\mathbf{v}) + \sigma(\mathbf{z}-\mathbf{s}-\mathbf{v}) - 2\sigma(\mathbf{z})|$.

D Double Margin Algorithm

Algorithm 1: Double Margin Training Algorithm

```
Data: Training data $\mathcal{D}$, Randomly initialized weights and biases $\mathbf{W}^i, \mathbf{b}^i, i = 1:N$, Number of
      epochs $E$, Perturbation size $\epsilon$, Regularization weight $\lambda$
Result: Trained weights and biases $\mathbf{W}^i, \mathbf{b}^i$
for epoch = 1:E do
    for $\mathbf{x}_{nom}$ in $\mathcal{D}$ do
        $\mathbf{z}^0 = \mathbf{x}_{nom}, \mathbf{s}^0 = \epsilon\mathbf{1}, \mathbf{v}^0 = \mathbf{0}$;
        for i = 1:N do
            Propagate $\mathbf{z}^{i+1}$ with Equation (1)
            Propagate $\mathbf{s}^{i+1}$ with Equation (7)
            Propagate $\mathbf{v}^{i+1}$ with Equation (8)
        end
        Construct regularized loss
         $\mathcal{L} + \lambda\mathcal{R} = \lambda L_+(\mathbf{z}^n + \mathbf{s}^n + \mathbf{v}^n) + \lambda L_-(\mathbf{z}^n - \mathbf{s}^n - \mathbf{v}^n) + (1-\lambda)L(\mathbf{z}^n)$
        Compute gradients of $\mathcal{L} + \lambda\mathcal{R}$ w.r.t. $\mathbf{W}^i, \mathbf{b}^i$
        Update $\mathbf{W}^i, \mathbf{b}^i$ via gradient descent
    end
end
```

E Proof of Proposition 1

Proof. Suppose $\mathcal{L}(\theta) + \lambda\mathcal{R}(\theta)$ has local minimum $\theta^*(\lambda)$. Note that at $\lambda = 0$, $\theta^* = \hat{\theta}$. Note that the derivative of the regularized loss is 0 at θ^*:

$$\nabla_\theta \mathcal{L}(\theta^*) + \lambda \nabla_\theta \mathcal{R}(\theta^*) = 0$$

Differentiating both sides with respect to λ:

$$H(\mathcal{L}(\theta^*))\frac{d}{d\lambda}\theta^* + \nabla_\theta\mathcal{R}(\theta^*) + \lambda\frac{d}{d\lambda}\nabla_\theta\mathcal{R}(\theta^*) = 0$$

Solving for $\frac{d}{d\lambda}\theta^*$:

$$\frac{d}{d\lambda}\theta^* = -H(\mathcal{L}(\theta^*))^{-1}(\nabla_\theta\mathcal{R}(\theta^*) + \lambda\frac{d}{d\lambda}\nabla_\theta\mathcal{R}(\theta^*))$$

Evaluating at $\lambda = 0$:

$$\frac{d}{d\lambda}\theta^*|_{\lambda=0} = -H(\mathcal{L}(\hat{\theta}))^{-1}\nabla_\theta\mathcal{R}(\hat{\theta})$$

This implies that for small λ:

$$\theta^* \approx \hat{\theta} - \lambda H(\mathcal{L}(\hat{\theta}))^{-1}\nabla_\theta \mathcal{R}(\hat{\theta})$$

□

Intuitively, this statement indicates that for small amounts of regularization, the gradient of the regularizer is sufficient to determine the behavior of the regularizer: note that in Equation (5) the position of the new local minimizer only depends on the regularizer via its gradient. This can be used to analyze which robust regularizer is optimal with respect to another known metric such as generalization error.

F PROOF OF PROPOSITION 2

Proof. Note that at $\lambda = 0$, $\theta^* = \hat{\theta}$. Consider the difference between the performance of θ^* and $\hat{\theta}$ on regularized loss $\mathcal{L}(\theta) + \lambda\mathcal{R}_2(\theta)$:

$$D(\lambda) = \mathcal{L}(\hat{\theta}) + \lambda\mathcal{R}_2(\hat{\theta}) - \mathcal{L}(\theta^*) - \lambda\mathcal{R}_2(\theta^*)$$

Taking the derivative with respect to λ:

$$\frac{d}{d\lambda}D(\lambda) = -\nabla_\theta\mathcal{L}(\theta^*)^T\frac{d}{d\lambda}\theta^* - \lambda(\nabla_\theta\mathcal{R}_2(\theta^*)^T\frac{d}{d\lambda}\theta^*) + (\mathcal{R}_2(\hat{\theta}) - \mathcal{R}_2(\theta^*))$$

Note that evaluated at $\lambda = 0$, this quantity is 0 since $\nabla_\theta\mathcal{L}(\hat{\theta}) = \mathbf{0}$. Taking a second derivative with respect to λ:

$$\begin{aligned}\frac{d^2}{d\lambda^2}D(\lambda) = -\frac{d}{d\lambda}\theta^{*T}H(\mathcal{L}(\theta^*))\frac{d}{d\lambda}\theta^* - \nabla_\theta\mathcal{L}(\theta^*)^T\frac{d^2}{d\lambda^2}\theta^* - 2\nabla_\theta\mathcal{R}_2(\theta^*)^T\frac{d}{d\lambda}\theta^* \\ - \lambda\frac{d}{d\lambda}(\nabla_\theta\mathcal{R}_2(\theta^*)^T\frac{d}{d\lambda}\theta^*)\end{aligned}$$

Evaluating at $\lambda = 0$:

$$\frac{d^2}{d\lambda^2}D(\lambda)|_{\lambda=0} = -\frac{d}{d\lambda}\theta^{*T}|_{\lambda=0}H(\mathcal{L}(\hat{\theta}))\frac{d}{d\lambda}\theta^*|_{\lambda=0} - 2\nabla_\theta\mathcal{R}_2(\hat{\theta})^T\frac{d}{d\lambda}\theta^*|_{\lambda=0},$$

since $\nabla_\theta\mathcal{L}(\hat{\theta}) = 0$. Substituting the expression for $\frac{d}{d\lambda}\theta^*|_{\lambda=0}$ from Proposition 1:

$$\begin{aligned}\frac{d^2}{d\lambda^2}D(\lambda)|_{\lambda=0} = -\nabla_\theta\mathcal{R}_1(\hat{\theta})^TH(\mathcal{L}(\hat{\theta}))^{-1}H(\mathcal{L}(\hat{\theta}))H(\mathcal{L}(\hat{\theta}))^{-1}\nabla_\theta\mathcal{R}_1(\hat{\theta}) \\ + 2\nabla_\theta\mathcal{R}_2(\hat{\theta})^TH(\mathcal{L}(\hat{\theta}))^{-1}\nabla_\theta\mathcal{R}_1(\hat{\theta})\end{aligned}$$

Simplifying:

$$\frac{d^2}{d\lambda^2}D(\lambda)|_{\lambda=0} = -||\nabla_\theta\mathcal{R}_1(\hat{\theta}) - \nabla_\theta\mathcal{R}_2(\hat{\theta})||^2_{H(\mathcal{L}(\hat{\theta}))^{-1}} + ||\nabla_\theta\mathcal{R}_2(\hat{\theta})||^2_{H(\mathcal{L}(\hat{\theta}))^{-1}}$$

Where $||a||_B = \sqrt{a^TBa}$. This implies that for small $\lambda \neq 0$, $D(\lambda) > 0$ is equivalent to the condition:

$$||\nabla_\theta\mathcal{R}_1(\hat{\theta}) - \nabla_\theta\mathcal{R}_2(\hat{\theta})||_{H(\mathcal{L}(\hat{\theta}))^{-1}} < ||\nabla_\theta\mathcal{R}_2(\hat{\theta})||_{H(\mathcal{L}(\hat{\theta}))^{-1}}$$

□

This condition provides some guidelines on how to select a regularizer provided some "true" objective which is also a function of model parameters θ. The true objective $\mathcal{J}(\theta)$ could, for example, be a robust generalization bound or the difference between robust validation set loss and clean training set loss. Given a standard loss function $\mathcal{L}(\theta)$ and regularization parameter λ, the true objective can be expressed in the form of a regularization: $\mathcal{J}(\theta) = \mathcal{L}(\theta) + \lambda\mathcal{R}_\mathcal{J}(\hat{\theta})$. Suppose $\mathcal{L}(\theta)$ has minimum $\hat{\theta}$, and suppose there is a set of feasible regularizers that can be used during training of the network $\{\mathcal{R}_1(\theta), \mathcal{R}_2(\theta), \ldots, \mathcal{R}_k(\theta)\}$ with regularization coefficient λ. Then, Proposition 2 suggests selecting the regularizer $\tilde{\mathcal{R}}(\theta) = \arg\min_i ||\nabla_\theta\mathcal{R}_i(\hat{\theta}) - \nabla_\theta\mathcal{R}_\mathcal{J}(\hat{\theta})||_{H(\mathcal{L}(\hat{\theta}))^{-1}}$. Here $\mathcal{R}_i(\hat{\theta})$ corresponds to the minimized regularizer $\mathcal{R}_1(\hat{\theta})$ in Proposition 2 and $\mathcal{R}_\mathcal{J}(\hat{\theta})$ corresponds to the target regularizer $\mathcal{R}_2(\hat{\theta})$. Intuitively, the regularizer that is closest to $\mathcal{R}_\mathcal{J}(\theta)$ in gradient is selected. This selection of regularizer is optimal as λ approaches zero. For example, in the case of robust training, the robust training method with the closest gradient to $\mathcal{R}_\mathcal{J}(\theta)$ is best.

G Architecture and Hyperparameter Details

The small CNN architecture uses convolution layers of 16 and 32 filters followed by a 100 unit fully connected layer. The pureconv CNN architecture uses three convolution layers of 16 filters. The medium CNN architecture uses convolutional layers of 16, 16, 32 and 32 filters followed by two 512 unit fully connected layers. The large CNN architecture uses convolutional layers of 64, 64, 64, 128 and 128 filters followed by a 200 unit fully connected layer. The pooling CNN architecture uses two convolution layers of 16 filters each followed by a max-pooling layer of stride 2-by-2. The resnet CNN architecture uses a 16 filter convolution layer followed by a simple residual block with two 16 filter convolution layers. The MNIST and GTSRB networks are trained for 100 epochs each while the CIFAR networks are trained for 350 epochs each. For robust training methods, we use $\epsilon \in \{0.1, 0.2, 0.3\}$ for MNIST and $\epsilon \in \{2/255, 8/255\}$ for CIFAR and GTSRB as the training target perturbation size ϵ_{train}. We used fixed hyperparameters for a fair comparison among training methods (including ours without fine parameter tuning), which may cause different baseline clean accuracies from prior work.

H Extensions

H.1 ℓ_0 Norm Threat Model

In addition to a ℓ_p threat model, we can consider a ℓ_0 threat model where an input $\mathbf{x}_{nom}$ is perturbed by perturbation δ to produce perturbed input $\mathbf{x} = \mathbf{x}_{nom} + \delta$ where $||\delta||_0 \leq k$, where $||\delta||_0$ is the number of non-zero elements in δ. This corresponds to perturbing at most k dimensions of the input by any amount. In practice, in the case of image inputs for example, perturbations are limited by the range of values each dimension can take. For example, pixels are typically restricted to be in a limited range such as $[0, 1]$.

In this case, our regularizer can be used directly without modification when using a large ℓ_∞ perturbation size ϵ. It is also possible to modify our regularizer at the first layer of margin propagation. At the first layer, the margin $\mathbf{s}^1$ can be set exactly to one half the range of values each node can take:

$$\mathbf{s}^1_j = \frac{1}{2}\sum_{l=1}^{k} rank(\mathbf{W}^0_{j,:} \odot (1 - \mathbf{x}_{nom}), l) - rank(\mathbf{W}^0_{j,:} * \mathbf{x}_{nom}, l),$$

assuming the input is restricted to range $[0, 1]$, where $rank(a, l)$ selects the lth largest element of a, and $\odot$ denotes element-wise multiplication. Doing so makes our regularizer more closely match the true range of values layers can take for ℓ_0 perturbations.

H.2 Model Ensemble

It is also possible to extend our regularizer to model ensembles. Suppose there are N models $f_1(\mathbf{x}), f_2(\mathbf{x}), ...$
$f_N(\mathbf{x})$. Then the N models can be trained together using our regularizer. One method is to average together the output layers of the individual models to yield an ensemble model:

$$f_{ensemble}(\mathbf{x}) = \frac{1}{N}\sum_i f_i(\mathbf{x})$$

When using our regularizer, the regularization margins of the N models can also be averaged together. The ensemble loss is then constructed by substituting the averaged values into Equation (9). During inference, the final layers can be averaged together before classification. Alternatively, we can select the class with the most votes among the ensembled models.

I Additional Results

I.1 Effectiveness on other norms

Table 4 shows ℓ_0 certified accuracies for a small CNN architecture trained on MNIST and CIFAR. We show results for both standard Double Margin trained with $\epsilon = 0.3$ and $\epsilon = 8/255$ for MNIST and CIFAR respectively. We also show results for Double Margin modified for the ℓ_0 threat model trained with $\epsilon = 2$. As illustrated, the Double Margin ℓ_0 regularizer yields certified accuracies that decay slowly with the number of perturbed dimensions ϵ. Surprisingly, we find that standard Double Margin performs better than the ℓ_0 modification for ℓ_0 certified accuracies, and moreover has consistent performance across perturbation sizes. This suggests that training methods that yield high ℓ_∞ certified performance on also transfer to yield high ℓ_0 certified performance. Intuitively, this may be because networks which are highly certifiable for low levels of ℓ_∞ perturbations have particular weight patterns such as sparsity that allow for high certification in general including for ℓ_0 perturbations. We hypothesize that this effect is particular to ℓ_0 due to the non-convexity of the ℓ_0 norm during training.

To support the reasoning that the transferability of Double Margin to a ℓ_0 threat model is unique to ℓ_0, we conduct experiments comparing different versions of Double Margin ℓ_p with $p = \infty, 2, 1$ (see Section 3.2, Double Margin: Formal Definition for details). The three different versions of Double Margin are each certified on the three different ℓ_p norms. Small CNN networks are trained on MNIST and CIFAR with $\epsilon = 0.3, 1, 4$ and $\epsilon = 8/255, 0.2, 2$ for $\ell_\infty, \ell_2, \ell_1$ norms respectively. All versions are trained with fixed hyperparameters for each dataset. We find that on the largest perturbation size evaluated, each version of Double Margin generally performs best on its respective norm (Double Margin ℓ_∞ has the highest ℓ_∞ certified accuracy etc.) This observation supports the hypothesis that the effectiveness of Double Margin ℓ_∞ on ℓ_0 is unique. Interestingly, for smaller perturbation sizes Double Margin ℓ_∞ achieves better certified accuracy on ℓ_2 perturbations than Double Margin ℓ_2. Similarly, for small perturbation sizes, Double Margin ℓ_2 achieves better certified accuracy than Double Margin ℓ_1. This may be because certifiability on ℓ_2 and ℓ_1 norms incurs a stronger penalty on clean accuracy compared to the ℓ_∞ norm.

I.2 Model Ensemble

Table 5 shows certified accuracies for two ensemble models trained on MNIST and CIFAR, each composed of three models. The ensemble models are compared with averaged certified accuracies for three individual models. Certified accuracies for the ensemble models are computed with CNN-Cert by considering the ensemble model as a single model with multiple parallel sub-components. We find that the ensemble model achieves the same or higher certified accuracies for nearly all the perturbation sizes tested. This suggests that using ensembles may be a method of enhancing certified accuracies for smaller perturbation sizes without sacrificing performance on larger perturbation sizes.

I.3 GTSRB

Table 6 shows certified accuracies computed using CNN-Cert for a pureconv CNN model trained on GTSRB. Certified accuracies are computed over 200 random test set points. We compare Double Margin with IBP and normal training. For IBP and Double Margin, training is performed with perturbation sizes of $\epsilon = 2/255, 8/255$. Double Margin achieves higher certified accuracy than IBP over all tested perturbation sizes $\epsilon \geq 2/255$. In particular, we achieve a 24% improvement in certified accuracy for $\epsilon = 8/255$. Double Margin also trains in less time than IBP.

I.4 Medium CNN Model

Tables 8 and 9 show complete certified accuracy results computed using CNN-Cert for including for Medium CNN models. Due to the cost of certifying on models of this size, we compute certified accuracies over 20 points. Double Margin matches or outperforms IBP over most perturbation sizes, and particularly improves on IBP for large perturbation sizes ($\epsilon = 0.1, 0.2, 0.3$ for MNIST and $\epsilon = 2/255, 3/255, 5/255$ for CIFAR).

I.5 Full Test Set

Table 10 shows CNN-Cert certified accuracies computed on the full test set for two selected models trained on MNIST and CIFAR. The certified accuracies are similar to those computed on 200 randomly selected test set points, indicating that the random sample is fairly representative.

J Additional Tables

Method		Training Time (sec)	$\epsilon_{cert} = 0$	0.05	0.10	0.20	0.30
			Small CNN MNIST, 4 layer, ℓ_∞				
Double Margin ℓ_∞	$\epsilon = 0.3$	609.7	**89.0%**	**89.0%**	**88.0%**	**81.0%**	**63.5%**
Double Margin ℓ_1	$\epsilon = 4$	593.7	68.5%	59.5%	50.5%	9.0%	0.5%
Double Margin ℓ_0	$\epsilon = 2$	593.3	86.0%	76.0%	57.5%	4.0%	0.0%
Method		Training Time (sec)	$\epsilon_{cert} = 0$	0.25	0.50	0.75	1.00
			Small CNN MNIST, 4 layer, ℓ_1				
Double Margin ℓ_∞	$\epsilon = 0.3$	609.7	**89.0%**	0.0%	0.0%	0.0%	0.0%
Double Margin ℓ_1	$\epsilon = 4$	593.7	68.5%	**62.5%**	**53.0%**	**35.5%**	**11.0%**
Method		Training Time (sec)	$\epsilon_{cert} = 0$	1.00	2.00	3.00	4.00
			Small CNN MNIST, 4 layer, ℓ_0				
Double Margin ℓ_∞	$\epsilon = 0.3$	609.7	**89.0%**	**89.0%**	**89.0%**	**89.0%**	**89.0%**
Double Margin ℓ_0	$\epsilon = 2$	593.3	86.0%	85.5%	84.0%	82.0%	60.5%
Method		Training Time (sec)	$\epsilon_{cert} = 0$	$1/255$	$3/255$	$5/255$	$8/255$
			Small CNN CIFAR, 4 layer, ℓ_∞				
Double Margin ℓ_∞	$\epsilon = 8/255$	3058.8	**42.5%**	**40.0%**	**35.0%**	**31.0%**	**20.5%**
Double Margin ℓ_1	$\epsilon = 2$	2975.4	30.0%	28.5%	24.5%	20.5%	19.5%
Double Margin ℓ_0	$\epsilon = 2$	2991.7	28.0%	27.0%	25.0%	22.5%	19.5%
Method		Training Time (sec)	$\epsilon_{cert} = 0$	0.05	0.10	0.15	0.20
			Small CNN CIFAR, 4 layer, ℓ_1				
Double Margin ℓ_∞	$\epsilon = 8/255$	3058.8	**42.5%**	0.0%	0.0%	0.0%	0.0%
Double Margin ℓ_1	$\epsilon = 2$	2975.4	30.0%	**28.5%**	**27.0%**	**23.5%**	**22.5%**
Method		Training Time (sec)	$\epsilon_{cert} = 0$	1.00	2.00	3.00	4.00
			Small CNN CIFAR, 4 layer, ℓ_0				
Double Margin ℓ_∞	$\epsilon = 8/255$	3058.8	**42.5%**	**42.5%**	**42.5%**	**42.5%**	**42.5%**
Double Margin ℓ_0	$\epsilon = 2$	2991.7	28.0%	27.5%	27.5%	26.0%	24.0%

Table 4: CNN-Cert certified accuracies for **Effectiveness on other norms**

Method		Training Time (sec)	$\epsilon_{cert} = 0$	0.01	0.03	0.05	0.07	0.10	0.20	0.30	0.40
					Small CNN MNIST, 4 layer						
Double Margin	$\epsilon = 0.2$	521.1	**95.5%**	**95.5%**	95.2%	94.2%	93.5%	91.7%	81.3%	0.0%	0.0%
Double Margin Ensemble		1411.2	**95.5%**	**95.5%**	**95.5%**	**95.5%**	**94.5%**	**93.5%**	**85.5%**	0.0%	0.0%
Method		Training Time (sec)	$\epsilon_{cert} = 0$	$0.5/255$	$1/255$	$2/255$	$3/255$	$5/255$	$7/255$	$8/255$	$9/255$
					Small CNN CIFAR, 4 layer						
Double Margin	$\epsilon = 8/255$	2517.3	42.8%	41.8%	39.8%	36.8%	36.3%	26.8%	18.8%	15.8%	**13.8%**
Double Margin Ensemble		6566.2	**46.0%**	**44.0%**	**43.5%**	**40.0%**	**38.0%**	**27.0%**	**22.0%**	**16.0%**	11.0%

Table 5: CNN-Cert accuracies for **Model ensemble**. The ensemble uses three models with different random initialization and random training batches. For fair comparison with standard Double Margin, three models with the same random initializations and training batches are trained independently. The training times and certified accuracies for standard Double Margin are averaged over these three models.

Method		Training Time (sec)	$\epsilon = 0$	$0.5/255$	$1/255$	$2/255$	$3/255$	$5/255$	$7/255$	$8/255$	$9/255$
			Pureconv CNN GTSRB, 4 layer								
Normal		119.6	92.0%	77.5%	57.5%	37.0%	35.0%	6.5%	1.0%	0.0%	0.0%
IBP	$\epsilon = 2/255$	289.3	**84.0%**	**81.5%**	**78.0%**	65.0%	63.5%	31.0%	14.5%	10.0%	5.5%
Double Margin		**282.6**	83.0%	80.0%	77.5%	**68.5%**	**67.0%**	43.5%	27.5%	22.0%	16.5%
IBP	$\epsilon = 8/255$	287.7	64.0%	63.0%	62.0%	58.5%	57.0%	44.5%	31.5%	26.0%	22.0%
Double Margin		283.1	68.0%	67.5%	67.0%	65.0%	64.5%	**58.0%**	**52.5%**	**50.0%**	**45.5%**

Table 6: CNN-Cert certified accuracies on GTSRB networks.

Method		Training Time (sec)	$\epsilon_{cert} = 0$	0.05	0.10	0.20	0.30
Small CNN MNIST, 4 layer, ℓ_∞							
Double Margin ℓ_∞	$\epsilon = 0.3$	609.7	**89.0%**	**89.0%**	**88.0%**	**81.0%**	**63.5%**
Double Margin ℓ_2	$\epsilon = 1$	592.0	68.5%	61.5%	50.0%	22.5%	1.5%
Double Margin ℓ_1	$\epsilon = 4$	593.7	68.5%	59.5%	50.5%	9.0%	0.5%
Double Margin ℓ_0	$\epsilon = 2$	593.3	86.0%	76.0%	57.5%	4.0%	0.0%
Method		Training Time (sec)	$\epsilon_{cert} = 0$	0.25	0.50	0.75	1.00
Small CNN MNIST, 4 layer, ℓ_2							
Double Margin ℓ_∞	$\epsilon = 0.3$	609.7	**89.0%**	**84.5%**	0.0%	0.0%	0.0%
Double Margin ℓ_2	$\epsilon = 1$	592.0	68.5%	62.0%	**52.0%**	**41.5%**	**24.0%**
Double Margin ℓ_1	$\epsilon = 4$	593.7	68.5%	61.0%	51.5%	34.0%	9.0%
Double Margin ℓ_0	$\epsilon = 2$	593.3	86.0%	80.5%	69.0%	39.0%	7.5%
Method		Training Time (sec)	$\epsilon_{cert} = 0$	1.00	2.00	3.00	4.00
Small CNN MNIST, 4 layer, ℓ_1							
Double Margin ℓ_∞	$\epsilon = 0.3$	609.7	**89.0%**	0.0%	0.0%	0.0%	0.0%
Double Margin ℓ_2	$\epsilon = 1$	592.0	68.5%	60.5%	39.0%	0.0%	0.0%
Double Margin ℓ_1	$\epsilon = 4$	593.7	68.5%	**62.5%**	**53.0%**	**35.5%**	**11.0%**
Double Margin ℓ_0	$\epsilon = 2$	593.3	86.0%	81.0%	69.0%	39.5%	6.5%
Method		Training Time (sec)	$\epsilon_{cert} = 0$	1.00	2.00	3.00	4.00
Small CNN MNIST, 4 layer, ℓ_0							
Double Margin ℓ_∞	$\epsilon = 0.3$	609.7	**89.0%**	**89.0%**	**89.0%**	**89.0%**	**89.0%**
Double Margin ℓ_2	$\epsilon = 1$	592.0	68.5%	68.5%	68.5%	68.5%	68.5%
Double Margin ℓ_1	$\epsilon = 4$	593.7	68.5%	68.5%	68.5%	68.5%	67.5%
Double Margin ℓ_0	$\epsilon = 2$	593.3	86.0%	85.5%	84.0%	82.0%	60.5%
Method		Training Time (sec)	$\epsilon_{cert} = 0$	$1/255$	$3/255$	$5/255$	$8/255$
Small CNN CIFAR, 4 layer, ℓ_∞							
Double Margin ℓ_∞	$\epsilon = 8/255$	3058.8	**42.5%**	**40.0%**	**35.0%**	**31.0%**	**20.5%**
Double Margin ℓ_2	$\epsilon = 0.2$	2969.9	35.0%	32.5%	27.5%	25.5%	17.5%
Double Margin ℓ_1	$\epsilon = 2$	2975.4	30.0%	28.5%	24.5%	20.5%	19.5%
Double Margin ℓ_0	$\epsilon = 2$	2991.7	28.0%	27.0%	25.0%	22.5%	19.5%
Method		Training Time (sec)	$\epsilon_{cert} = 0$	0.05	0.10	0.15	0.20
Small CNN CIFAR, 4 layer, ℓ_2							
Double Margin ℓ_∞	$\epsilon = 8/255$	3058.8	**42.5%**	**33.5%**	18.5%	3.5%	0.0%
Double Margin ℓ_2	$\epsilon = 0.2$	2969.9	35.0%	33.0%	**31.5%**	**28.0%**	**26.5%**
Double Margin ℓ_1	$\epsilon = 2$	2975.4	30.0%	28.5%	27.0%	26.0%	23.5%
Double Margin ℓ_0	$\epsilon = 2$	2991.7	28.0%	27.0%	26.5%	25.0%	24.0%
Method		Training Time (sec)	$\epsilon_{cert} = 0$	0.50	1.00	1.50	2.00
Small CNN CIFAR, 4 layer, ℓ_1							
Double Margin ℓ_∞	$\epsilon = 8/255$	3058.8	**42.5%**	0.0%	0.0%	0.0%	0.0%
Double Margin ℓ_2	$\epsilon = 0.2$	2969.9	35.0%	**31.5%**	**28.0%**	**24.0%**	20.0%
Double Margin ℓ_1	$\epsilon = 2$	2975.4	30.0%	28.5%	27.0%	23.5%	22.5%
Double Margin ℓ_0	$\epsilon = 2$	2991.7	28.0%	27.5%	26.0%	**24.0%**	**23.5%**
Method		Training Time (sec)	$\epsilon_{cert} = 0$	1.00	2.00	3.00	4.00
Small CNN CIFAR, 4 layer, ℓ_0							
Double Margin ℓ_∞	$\epsilon = 8/255$	3058.8	**42.5%**	**42.5%**	**42.5%**	**42.5%**	**42.5%**
Double Margin ℓ_2	$\epsilon = 0.2$	2969.9	35.0%	35.0%	35.0%	34.5%	34.0%
Double Margin ℓ_1	$\epsilon = 2$	2975.4	30.0%	30.0%	30.0%	29.0%	27.5%
Double Margin ℓ_0	$\epsilon = 2$	2991.7	28.0%	27.5%	27.5%	26.0%	24.0%

Table 7: CNN-Cert certified accuracies for various ℓ_p versions of Double Margin computed for different ℓ_p perturbation sizes.

Method		Training Time (sec)	$\epsilon_{cert} = 0$	0.01	0.03	0.05	0.07	0.10	0.20	0.30	0.40
				Small CNN MNIST, 4 layer							
Normal		185.7	98.5%	9.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Weight Regularization		626.3	99.5%	81.0%	1.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Adv. Training		8873.6	**100.0%**	97.0%	91.0%	0.5%	0.0%	0.0%	0.0%	0.0%	0.0%
Bounded		27994.8	48.5%	48.5%	48.5%	48.5%	48.0%	48.0%	0.0%	0.0%	0.0%
IBP		613.1	99.5%	**99.5%**	**99.0%**	**98.0%**	**96.5%**	83.0%	0.0%	0.0%	0.0%
ReLU Stabilty		602.3	97.5%	89.5%	40.5%	1.5%	0.0%	0.0%	0.0%	0.0%	0.0%
L1+RS+Pruning	$\epsilon = 0.1$	686.7	98.0%	90.0%	34.5%	1.5%	0.0%	0.0%	0.0%	0.0%	0.0%
Adv+RS+Pruning		9710.9	98.5%	96.5%	92.0%	84.5%	75.0%	54.0%	0.0%	0.0%	0.0%
Double Margin		520.0	98.5%	98.5%	97.0%	96.5%	**96.5%**	95.0%	0.0%	0.0%	0.0%
Double Margin +L1		545.5	98.5%	98.5%	97.5%	97.0%	**96.5%**	**95.5%**	0.0%	0.0%	0.0%
Adv. Training		8786.8	99.0%	98.5%	91.0%	38.0%	0.5%	0.0%	0.0%	0.0%	0.0%
Bounded		27999.5	77.5%	77.5%	77.5%	77.5%	77.5%	77.5%	75.0%	0.0%	0.0%
IBP		614.0	97.5%	97.5%	97.0%	96.5%	96.0%	78.0%	0.0%	0.0%	0.0%
ReLU Stabilty		592.6	95.0%	90.0%	59.0%	12.5%	0.5%	0.0%	0.0%	0.0%	0.0%
L1+RS+Pruning	$\epsilon = 0.2$	684.8	95.0%	89.5%	55.5%	10.5%	1.0%	0.0%	0.0%	0.0%	0.0%
Adv+RS+Pruning		9669.4	93.0%	92.0%	86.5%	80.0%	63.5%	46.0%	3.5%	0.0%	0.0%
Double Margin		**518.8**	95.0%	94.5%	94.0%	93.0%	92.5%	92.5%	81.5%	0.0%	0.0%
Double Margin +L1		546.1	95.5%	95.0%	95.0%	95.0%	95.0%	93.5%	82.0%	0.0%	0.0%
Adv. Training		8760.5	99.5%	98.5%	91.5%	38.5%	1.0%	0.0%	0.0%	0.0%	0.0%
Bounded		27966.8	30.0%	30.0%	30.0%	30.0%	30.0%	29.5%	28.5%	25.0%	0.0%
IBP		611.2	95.5%	95.5%	95.5%	90.5%	76.5%	39.5%	0.0%	0.0%	0.0%
ReLU Stabilty		595.8	94.0%	90.0%	65.0%	23.0%	5.0%	0.5%	0.0%	0.0%	0.0%
L1+RS+Pruning	$\epsilon = 0.3$	686.5	95.0%	88.0%	53.5%	13.5%	3.0%	0.5%	0.0%	0.0%	0.0%
Adv+RS+Pruning		9657.1	90.0%	86.5%	70.5%	55.0%	30.5%	13.5%	0.0%	0.0%	0.0%
Double Margin		520.0	90.0%	89.0%	88.5%	88.0%	87.5%	86.5%	80.0%	66.5%	0.0%
Double Margin +L1		547.1	91.0%	90.5%	90.0%	89.5%	89.0%	87.5%	**83.5%**	**68.0%**	0.0%
				Medium CNN MNIST, 7 layer							
Normal		387.4	100%	0%	0%	0%	0%	0%	0%	0%	0%
IBP		1553.6	**100%**	**100%**	**100%**	**100%**	90%	85%	0%	0%	0%
Double Margin	$\epsilon = 0.1$	**1360.4**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	0%	0%	0%
IBP		1553.2	**100%**	**100%**	**100%**	95%	80%	40%	0%	0%	0%
Double Margin	$\epsilon = 0.2$	1360.6	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**90%**	0%	0%
IBP		1550.6	**100%**	**100%**	**100%**	**100%**	90%	50%	0%	0%	0%
Double Margin	$\epsilon = 0.3$	1362.3	90%	90%	90%	90%	90%	90%	**90%**	**80%**	0%
				Pooling CNN MNIST, 5 layer							
Normal		226.0	99.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Weight Regularization		145.1	99.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Adv. Training		7747.8	**99.5%**	**98.5%**	4.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
IBP		1075.6	98.0%	92.0%	77.5%	64.5%	40.5%	1.0%	0.0%	0.0%	0.0%
ReLU Stabilty	$\epsilon = 0.1$	337.3	98.5%	87.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Double Margin		1070.3	97.5%	95.0%	86.0%	80.0%	73.0%	55.5%	0.0%	0.0%	0.0%
Adv. Training		7573.4	**99.5%**	**98.5%**	16.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
IBP		1082.8	96.0%	78.0%	63.5%	51.0%	39.0%	20.0%	0.0%	0.0%	0.0%
ReLU Stabilty	$\epsilon = 0.2$	**336.7**	97.0%	90.0%	4.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Double Margin		680.5	94.5%	94.0%	**93.0%**	**91.5%**	**89.0%**	**82.5%**	4.5%	0.0%	0.0%
Adv. Training		7518.9	**99.5%**	97.5%	23.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
IBP		395.4	95.0%	76.5%	56.0%	44.0%	33.5%	13.5%	0.0%	0.0%	0.0%
ReLU Stabilty	$\epsilon = 0.3$	337.5	97.0%	88.5%	13.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Double Margin		408.0	88.0%	88.0%	87.5%	86.5%	85.0%	81.5%	**43.0%**	**0.5%**	0.0%
				1-Resnet CNN MNIST, 4 layer							
Normal		210.1	98.5%	2.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Weight Regularization		145.1	99.0%	51.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Adv. Training		7705.8	**99.0%**	97.5%	93.5%	22.5%	0.0%	0.0%	0.0%	0.0%	0.0%
IBP		600.2	**99.0%**	95.5%	94.5%	90.5%	87.5%	81.0%	3.0%	0.0%	0.0%
ReLU Stabilty	$\epsilon = 0.1$	318.4	93.0%	85.5%	58.5%	25.5%	9.0%	1.5%	0.0%	0.0%	0.0%
Double Margin		629.3	98.0%	80.5%	77.0%	74.5%	70.0%	65.5%	0.0%	0.0%	0.0%
Adv. Training		7554.4	98.0%	**98.0%**	**95.0%**	69.0%	9.5%	0.0%	0.0%	0.0%	0.0%
IBP		593.3	97.5%	62.0%	57.0%	52.0%	50.0%	45.0%	10.0%	0.0%	0.0%
ReLU Stabilty	$\epsilon = 0.2$	**317.8**	93.0%	85.5%	67.0%	34.5%	17.0%	2.5%	0.0%	0.0%	0.0%
Double Margin		630.0	95.0%	52.0%	51.5%	50.5%	50.0%	46.0%	28.5%	0.0%	0.0%
Adv. Training		7583.4	**99.0%**	97.5%	94.5%	73.0%	17.0%	0.0%	0.0%	0.0%	0.0%
IBP		600.3	96.5%	73.5%	69.0%	63.5%	57.5%	51.0%	13.0%	0.0%	0.0%
ReLU Stabilty	$\epsilon = 0.3$	319.0	93.0%	86.0%	70.5%	40.0%	19.5%	4.5%	0.0%	0.0%	0.0%
Double Margin		630.4	92.5%	77.0%	77.0%	**75.5%**	**75.0%**	**74.5%**	**62.5%**	**46.0%**	0.0%

Table 8: CNN-Cert certified accuracies on all MNIST networks. Adv. Training (Madry et al., 2018), Bounded (Kolter & Wong, 2018), IBP (Gowal et al., 2018), ReLU Stability (Xiao et al., 2019). For the medium CNN architecture, accuracies are found over 20 points.

Method		Training Time (sec)	$\epsilon = 0$	$0.5/255$	$1/255$	$2/255$	$3/255$	$5/255$	$7/255$	$8/255$	$9/255$
Small CNN CIFAR, 4 layer											
Normal		141.1	61.5%	22.0%	1.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Weight Regularization		1062.8	58.0%	24.0%	2.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Adv. Training	$\epsilon = 2/255$	51203.7	57.5%	24.5%	1.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Bounded		22351.6	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
IBP		394.8	47.0%	37.5%	22.0%	3.5%	3.5%	1.0%	0.0%	0.0%	0.0%
ReLU Stabilty		409.4	12.0%	12.0%	12.0%	12.0%	12.0%	12.0%	12.0%	12.0%	12.0%
Double Margin		**394.5**	45.0%	40.0%	34.0%	25.5%	24.0%	2.5%	0.5%	0.5%	0.5%
Adv. Training	$\epsilon = 8/255$	51893.6	**60.5%**	25.0%	2.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Bounded		21918.7	4.0%	4.0%	4.0%	4.0%	4.0%	4.0%	3.5%	3.5%	3.5%
IBP		410.0	37.5%	35.0%	32.5%	20.0%	17.5%	2.5%	0.0%	0.0%	0.0%
ReLU Stabilty		396.1	12.0%	12.0%	12.0%	12.0%	12.0%	12.0%	12.0%	12.0%	12.0%
Double Margin		409.7	45.5%	**45.0%**	**43.5%**	**40.5%**	**39.5%**	**28.0%**	**22.5%**	**19.5%**	**15.0%**
Medium CNN CIFAR, 7 layer											
Normal		1778.3	50.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
IBP	$\epsilon = 2/255$	6613.4	**50.0%**	**35.0%**	20.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Double Margin		5770.2	35.0%	25.0%	15.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
IBP	$\epsilon = 8/255$	6598.8	20.0%	15.0%	15.0%	10.0%	10.0%	0.0%	0.0%	0.0%	0.0%
Double Margin		**5762.2**	25.0%	25.0%	**25.0%**	**25.0%**	**25.0%**	**25.0%**	**20.0%**	**10.0%**	**10.0%**
Pooling CNN CIFAR, 5 layer											
Normal		1164.9	65.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Weight Regularization		828.8	59.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
IBP	$\epsilon = 2/255$	4112.8	52.0%	24.0%	16.0%	6.0%	4.0%	0.0%	0.0%	0.0%	0.0%
ReLU Stabilty		1682.6	**60.5%**	1.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Double Margin		4091.4	50.5%	30.5%	23.5%	11.0%	8.0%	1.0%	0.0%	0.0%	0.0%
IBP	$\epsilon = 8/255$	4015.1	43.0%	15.5%	8.5%	3.0%	3.0%	0.0%	0.0%	0.0%	0.0%
ReLU Stabilty		**1676.9**	58.5%	8.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Double Margin		2899.5	40.0%	**32.5%**	**30.5%**	**24.0%**	**23.5%**	**5.0%**	**1.0%**	**1.0%**	**0.5%**
1-Resnet CNN CIFAR, 4 layer											
Normal		1153.7	52.5%	4.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Weight Regularization		847.3	58.0%	3.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
IBP	$\epsilon = 2/255$	3700.3	**48.5%**	8.0%	6.0%	2.0%	2.0%	0.0%	0.0%	0.0%	0.0%
ReLU Stabilty		**1592.6**	37.5%	4.0%	4.0%	4.0%	4.0%	0.0%	0.0%	0.0%	0.0%
Double Margin		3393.9	**48.5%**	**9.0%**	**8.0%**	**4.0%**	**4.0%**	1.5%	0.5%	0.5%	0.5%
IBP	$\epsilon = 8/255$	3421.6	42.5%	2.5%	2.0%	1.0%	1.0%	0.0%	0.0%	0.0%	0.0%
ReLU Stabilty		1594.4	40.0%	1.0%	1.0%	1.0%	1.0%	1.0%	0.5%	0.5%	0.0%
Double Margin		3531.5	35.0%	4.0%	4.0%	**4.0%**	**4.0%**	**4.0%**	**3.5%**	**3.5%**	**3.5%**

Table 9: CNN-Cert certified accuracies on all CIFAR networks. Adv. Training (Madry et al., 2018), Bounded (Kolter & Wong, 2018), IBP (Gowal et al., 2018), ReLU Stability (Xiao et al., 2019). For the mediun CNN architecture, accuracies are found over 20 points.

Method		Training Time (sec)	$\epsilon_{cert} = 0$	0.01	0.03	0.05	0.07	0.10	0.20	0.30	0.40
Small CNN MNIST, 4 layer											
Double Margin	$\epsilon = 0.2$	518.8	93.44%	93.15%	92.61%	92.06%	91.30%	89.63%	78.55%	0.00%	0.00%
Method		Training Time (sec)	$\epsilon = 0$	$0.5/255$	$1/255$	$2/255$	$3/255$	$5/255$	$7/255$	$8/255$	$9/255$
Small CNN CIFAR, 4 layer											
Double Margin	$\epsilon = 8/255$	409.7	43.00%	41.71%	40.58%	37.72%	37.25%	28.44%	22.39%	19.06%	16.35%

Table 10: CNN-Cert certified accuracies computed on the entire test set.

Method		Training Time (sec)	$\epsilon_{cert} = 0$	0.01	0.03	0.05	0.07	0.10	0.20	0.30	0.40
					Medium CNN MNIST, 7 layer						
Normal		387.4	99.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
IBP	$\epsilon = 0.1$	1553.6	98.5%	98.5%	98.5%	98.5%	97.5%	96.5%	0.0%	0.0%	0.0%
Double Margin	$\epsilon = 0.1$	1360.4	97.5%	97.5%	97.5%	97.0%	96.5%	96.5%	0.0%	0.0%	0.0%
IBP	$\epsilon = 0.2$	1553.2	98.5%	98.5%	98.5%	98.0%	98.0%	97.0%	94.0%	0.0%	0.0%
Double Margin	$\epsilon = 0.2$	1360.6	95.0%	95.0%	94.5%	94.5%	94.0%	93.5%	89.5%	0.0%	0.0%
IBP	$\epsilon = 0.3$	1550.6	97.5%	97.5%	97.5%	97.5%	96.5%	95.5%	94.0%	90.0%	0.0%
Double Margin	$\epsilon = 0.3$	1362.3	90.5%	90.5%	88.5%	87.5%	87.5%	87.0%	83.5%	79.5%	0.0%
					Large CNN MNIST, 7 layer						
Normal		930.6	99.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
IBP	$\epsilon = 0.1$	4162.8	98.0%	98.0%	98.0%	98.0%	97.0%	95.0%	0.0%	0.0%	0.0%
Double Margin	$\epsilon = 0.1$	3742.5	98.5%	98.5%	98.0%	98.0%	98.0%	97.0%	0.0%	0.0%	0.0%
IBP	$\epsilon = 0.2$	4162.3	98.5%	98.5%	98.5%	98.5%	97.5%	97.5%	93.5%	0.0%	0.0%
Double Margin	$\epsilon = 0.2$	3740.2	97.5%	97.0%	96.5%	96.5%	96.5%	96.5%	95.5%	0.0%	0.0%
IBP	$\epsilon = 0.3$	4161.4	98.5%	98.0%	97.5%	97.5%	97.5%	97.5%	94.5%	91.0%	0.0%
Double Margin	$\epsilon = 0.3$	3739.2	93.0%	93.0%	92.5%	92.5%	92.0%	91.0%	84.5%	0.0%	0.0%
Method		Training Time (sec)	$\epsilon = 0$	$0.5/255$	$1/255$	$2/255$	$3/255$	$5/255$	$7/255$	$8/255$	$9/255$
					Medium CNN CIFAR, 7 layer						
Normal		1778.3	61.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
IBP	$\epsilon = 2/255$	6613.4	45.5%	36.5%	27.0%	11.0%	9.5%	0.0%	0.0%	0.0%	0.0%
Double Margin	$\epsilon = 2/255$	5770.2	45.0%	37.0%	24.5%	10.5%	8.0%	0.5%	0.0%	0.0%	0.0%
IBP	$\epsilon = 8/255$	6598.8	24.5%	24.5%	23.0%	21.5%	21.5%	18.5%	13.5%	12.0%	10.5%
Double Margin	$\epsilon = 8/255$	5762.2	33.5%	33.0%	32.5%	31.5%	31.5%	25.0%	18.5%	16.0%	13.5%
					Large CNN CIFAR, 7 layer						
Normal		4020.6	65.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
IBP	$\epsilon = 2/255$	17260.8	45.5%	33.5%	23.5%	10.0%	8.0%	0.5%	0.0%	0.0%	0.0%
Double Margin	$\epsilon = 2/255$	15141.5	43.5%	33.0%	25.5%	9.5%	9.0%	2.0%	1.0%	0.5%	0.0%
IBP	$\epsilon = 8/255$	17262.3	27.0%	27.0%	26.5%	25.0%	25.0%	23.5%	23.0%	22.5%	21.0%
Double Margin	$\epsilon = 8/255$	15131.7	16.5%	15.5%	15.0%	15.0%	15.0%	14.5%	14.5%	14.5%	14.5%

Table 11: Certified accuracies for large and medium CNN models computed using IBP (Gowal et al., 2018).

K Additional results in Rebuttal

K.1 Comparing different certification methods

Here, we conduct experiments comparing different certification methods. Specifically, we compare IBP with different variations of CNN-Cert including CNN-Cert-ReLU, CNN-Cert-Ada. These bounds differ by how they bound the ReLU function for unstable ReLUs as illustrated in Figure 3 with CNN-Cert-ReLU using parallel bounds and CNN-Cert-Ada using an adaptive lower bound. We also compare with a variation of CNN-Cert using a constant lower bound of zero, which we call CNN-Cert-Zero.

We note that this bound is strictly stronger than IBP as IBP uses a constant upper bound and lower bound to bound the activation function, whereas CNN-Cert-Zero uses a tighter linear upper bound and the same constant lower bound ($y = 0$), as clearly shown in Figure 3. As a result, for any fixed network, IBP yields weaker than CNN-Cert-Zero. Many certification methods in the literature are equivalent to one of these methods as illustrated in Table 12.

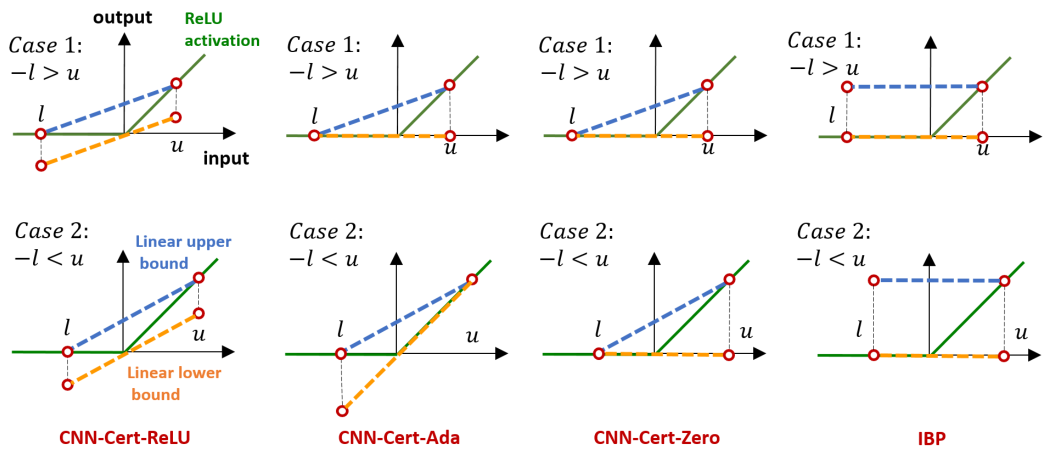


Figure 3: Visualizing the bounds on the ReLU function of different certification methods. Bounds are illustrated for unstable ReLUs where the input bounds l, u satisfy $l < 0$ and $u > 0$. Bounds for two cases are illustrated: 1) $-l > u$ and 2) $-l < u$. The linear upper bounds of the ReLU function are shown in blue, and the lower bounds shown in orange.

Certification method	Equivalents
CNN-Cert-ReLU	Convex Outer Bounds (Kolter & Wong, 2018)
	Fast-Lin (Weng et al., 2018)
	Zonotope (Singh et al., 2018)
	Neurify (Wang et al., 2018)
CNN-Cert-Ada	CROWN (Zhang et al., 2018)
	DeepPoly (Singh et al., 2019)
CNN-Cert-Zero	-
IBP	Box (Mirman et al., 2018)
	Naive Layer-wise Bounds (Kolter & Wong, 2018)

Table 12: Equivalent certification methods to different variants of CNN-Cert and IBP

Tables 13, 14, 15, 16 show certified accuracies using the different certification methods. We find that CNN-Cert-ReLU and CNN-Cert-Ada generally yield the highest accuracies for most networks. Using these certifiers, Double Margin generally yields higher certified accuracies than other methods.

However, for models trained on IBP, IBP certified accuracies are much higher than those of CNN-Cert-ReLU and CNN-Cert-Ada, outperforming Double Margin. The high performance of IBP trained models under IBP certification is most likely because the they are specifically trained to enhance certified performance under interval bounds. This is in line with the observation that training with a loss function targeted towards a specific certifier yields especially high performance under that certifier (Kolter & Wong, 2018; Raghunathan et al., 2018; Dvijotham et al., 2018a). Nevertheless, we find that the CNN-Cert-Zero certifier yields higher or equal accuracies to IBP certification over all networks.

Method (200 points)		$\epsilon_{cert} = 0$	0.01	0.03	0.05	0.07	0.10	0.20	0.30	0.40
				Small CNN MNIST, 4 layer						
Normal		99.5%	15.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Weight Regularization		99.5%	88.5%	2.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Adv. Training		**98.5%**	**96.5%**	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
IBP		95.5%	95.5%	**95.0%**	**92.5%**	87.0%	68.0%	9.5%	0.0%	0.0%
ReLU Stabilty	$\epsilon = 0.3$	94.0%	90.0%	65.0%	23.0%	5.0%	0.5%	0.0%	0.0%	0.0%
L1+RS+Pruning		95.0%	88.0%	53.5%	13.5%	3.0%	0.5%	0.0%	0.0%	0.0%
Double Margin		91.5%	91.0%	90.5%	90.0%	**88.5%**	**87.0%**	**77.0%**	**67.0%**	0.0%
Method (200 points)		$\epsilon_{cert} = 0$	$0.5/255$	$1/255$	$2/255$	$3/255$	$5/255$	$7/255$	$8/255$	$9/255$
				Small CNN CIFAR, 4 layer						
Normal		53.5%	28.5%	5.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Weight Regularization		60.5%	26.5%	3.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Adv. Training		57.0%	27.0%	3.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
IBP	$\epsilon = 8/255$	**47.5%**	**44.5%**	**41.5%**	35.5%	35.0%	18.5%	8.5%	5.5%	3.5%
Double Margin		41.0%	40.0%	**39.0%**	**36.0%**	**35.5%**	**29.0%**	**21.5%**	**19.0%**	**16.0%**

Table 13: CNN-Cert-Ada certified accuracies for **Comparing different certification methods**.

Method (200 points)		$\epsilon_{cert} = 0$	0.01	0.03	0.05	0.07	0.10	0.20	0.30	0.40
				Small CNN MNIST, 4 layer						
Normal		99.5%	5.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Weight Regularization		99.5%	81.0%	1.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Adv. Training		**98.5%**	**95.5%**	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
IBP		95.5%	**95.5%**	**95.5%**	**93.5%**	85.0%	45.5%	0.0%	0.0%	0.0%
ReLU Stabilty	$\epsilon = 0.3$	94.0%	90.0%	65.0%	23.0%	5.0%	0.5%	0.0%	0.0%	0.0%
L1+RS+Pruning		95.0%	88.0%	53.5%	13.5%	3.0%	0.5%	0.0%	0.0%	0.0%
Double Margin		91.5%	91.0%	90.5%	90.0%	**88.0%**	**87.0%**	**77.5%**	**66.0%**	0.0%
Method (200 points)		$\epsilon_{cert} = 0$	$0.5/255$	$1/255$	$2/255$	$3/255$	$5/255$	$7/255$	$8/255$	$9/255$
				Small CNN CIFAR, 4 layer						
Normal		60.5%	23.5%	2.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Weight Regularization		57.0%	24.0%	1.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Adv. Training		53.5%	25.0%	4.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
IBP	$\epsilon = 8/255$	**47.5%**	**44.5%**	**41.5%**	35.0%	34.0%	16.5%	4.5%	3.0%	1.5%
Double Margin		41.0%	40.0%	39.0%	**36.0%**	**35.5%**	**29.5%**	**20.0%**	**17.5%**	**15.0%**

Table 14: CNN-Cert-ReLU certified accuracies for **Comparing different certification methods**.

Method (200 points)		$\epsilon_{cert} = 0$	0.01	0.03	0.05	0.07	0.10	0.20	0.30	0.40
					Small CNN MNIST, 4 layer					
Normal		99.5%	11.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Weight Regularization		99.5%	85.0%	1.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Adv. Training		**98.5%**	**95.5%**	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
IBP		95.5%	**95.5%**	**95.5%**	**95.5%**	**95.5%**	**95.5%**	**93.5%**	**86.5%**	0.0%
ReLU Stabilty	$\epsilon = 0.3$	94.0%	90.0%	65.0%	23.0%	5.0%	0.5%	0.0%	0.0%	0.0%
L1+RS+Pruning		95.0%	87.5%	53.5%	13.0%	3.0%	0.5%	0.0%	0.0%	0.0%
Double Margin		91.5%	91.0%	90.5%	90.5%	88.5%	87.0%	82.0%	75.5%	0.0%
Method (200 points)		$\epsilon_{cert} = 0$	$0.5/255$	$1/255$	$2/255$	$3/255$	$5/255$	$7/255$	$8/255$	$9/255$
					Small CNN CIFAR, 4 layer					
Normal		53.5%	23.0%	3.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Weight Regularization		60.5%	15.5%	1.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Adv. Training		57.0%	20.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
IBP	$\epsilon = 8/255$	**47.5%**	**44.5%**	**41.5%**	**36.5%**	**36.0%**	**26.0%**	19.5%	17.5%	14.5%
Double Margin		41.0%	40.0%	39.5%	36.5%	35.5%	30.5%	**25.0%**	**21.0%**	**19.5%**

Table 15: CNN-Cert-Zero certified accuracies for **Comparing different certification methods**.

Method (200 points)		$\epsilon_{cert} = 0$	0.01	0.03	0.05	0.07	0.10	0.20	0.30	0.40
					Small CNN MNIST, 4 layer					
Normal		99.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Weight Regularization		99.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Adv. Training		**98.5%**	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
IBP		95.5%	**95.5%**	**95.5%**	**95.5%**	**95.5%**	**95.5%**	**93.5%**	**86.5%**	0.0%
ReLU Stabilty	$\epsilon = 0.3$	94.0%	33.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
L1+RS+Pruning		95.0%	44.0%	0.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Double Margin		91.5%	91.0%	90.5%	90.5%	88.5%	87.0%	81.0%	75.5%	0.0%
Method (200 points)		$\epsilon_{cert} = 0$	$0.5/255$	$1/255$	$2/255$	$3/255$	$5/255$	$7/255$	$8/255$	$9/255$
					Small CNN CIFAR, 4 layer					
Normal		53.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Weight Regularization		60.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Adv. Training		**57.0%**	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
IBP	$\epsilon = 8/255$	47.5%	**44.5%**	**41.5%**	**36.0%**	**36.0%**	25.5%	18.5%	16.5%	14.0%
Double Margin		41.0%	40.0%	39.0%	35.0%	35.0%	**29.5%**	**20.5%**	**18.5%**	**16.5%**

Table 16: IBP certified accuracies for **Comparing different certification methods**.