Training Provably Robust Models by Polyhedral Envelope Regularization

Chen Liu, Student Member, IEEE, Mathieu Salzmann, Member, IEEE and Sabine Süsstrunk, Fellow, IEEE

***Abstract*—Training certifiable neural networks enables us to obtain models with robustness guarantees against adversarial attacks. In this work, we introduce a framework to obtain a provable adversarial-free region in the neighborhood of the input data by a polyhedral envelope, which yields more fine-grained certified robustness than existing methods. We further introduce polyhedral envelope regularization (PER) to encourage larger adversarial-free regions and thus improve the provable robustness of the models. We demonstrate the flexibility and effectiveness of our framework on standard benchmarks; it applies to networks of different architectures and with general activation functions. Compared with state of the art, PER has negligible computational overhead; it achieves better robustness guarantees and accuracy on the clean data in various settings.**

***Index Terms*—Adversarial Training. Provable Robustness.**

I. Introduction

Despite their great success in many applications, modern deep learning models are vulnerable to adversarial attacks: small but well-designed perturbations can make the state-of-the-art models predict wrong labels with very high confidence [16], [30], [42]. The existence of such adversarial examples indicates unsatisfactory properties of the deep learning models' decision boundary [20], and poses a threat to the reliability of safety-critical machine learning systems.

As a consequence, studying the robustness of deep learning has attracted growing attention, from the perspective of both attack and defense strategies. Popular attack algorithms, such as the *Fast Gradient Sign Method* (FGSM) [16], the CW attack [6] and the *Projected Gradient Descent* (PGD) [28], typically exploit the gradient of the loss w.r.t. the input to generate adversarial examples. Recently, the state-of-the-art success rates have been attained with adaptive methods, such as *Auto Attack*[12]. All these methods assume that the attackers have access to the model parameters and thus belong to the "white-box attacks". On the contrary, "black-box attacks" tackle the cases where the attackers have limited access to the model, such as limited access to the output logits [2]), hard-label predictions ([8] and task settings [14].

To counteract such attacks, robust learning aims to learn a model which optimizes the worst-case loss over the allowable perturbations. Formally, given a model f_θ parameterized by θ, the loss function ℓ and a dataset $\mathcal{D}$, robust learning solves the following min-max problem:

$$\min_{\theta} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}} \max_{\mathbf{x}'\in\mathcal{S}_\epsilon(\mathbf{x})} \ell(f_\theta(\mathbf{x}'), y) \tag{1}$$

C. Liu, M. Salzmann, S. Süsstrunk are with the School of Computer and Communication Sciences, École polytechnique fédérale de Lausanne (EPFL), Lausanne, Switzerland.

Here, $\mathcal{S}_\epsilon(\mathbf{x})$ denotes the adversarial budget, which is the allowable perturbed input of the clean input $\mathbf{x}$. To solve (1), many defense algorithms have been proposed [5], [13], [27], [33], [34], [38], [51]. However, it was shown by [3], [12], [44] that most of them depend on obfuscated gradients for perceived robustness. In other words, these methods train models to fool gradient-based attacks but do not achieve true robustness. As a consequence, they become ineffective when subjected to stronger attacks. The remaining effective defense is adversarial training [28] and its extensions [1], [7], [19], [21], [29], [41], [50], [53], which augments the training data with adversarial examples. Nevertheless, while adversarial training yields good empirical performance under adaptive attacks, it still provides no *guarantees* of a model's robustness.

In this work, we focus on constructing certifiers to find *certified regions* of the input neighborhood where the model is *guaranteed* to give the correct prediction, and on using such certifiers to train a model to be *provably* robust against adversarial attacks. To obtain such robustness guarantee, there are two categories of methods: complete certifiers and incomplete certifiers. Complete certifiers can either guarantee the absence of an adversary or find an adversarial example given an adversarial budget. They are typically built on either *Satisfiability Modulo Theories* (SMT) [22] or *Mixed Integer Programming* (MIP) [43], [52]. The major disadvantages of complete certifiers are their super-polynomial complexity and applicability to only piecewise linear activation functions, such as ReLU. By contrast, incomplete certifiers are faster, more widely applicable but more conservative in terms of certified regions because they rely on approximations. In this context, techniques such as linear approximation [4], [48], [47], [49], [55], symbolic interval analysis [46], abstract transformers [15], [39], [40] and semidefinite programming [35], [36] have been exploited to offer better certified robustness. In addition, recent works use randomized smoothing [9], [37] to construct probabilistic certifiers, which provides robustness guarantees with high probability by Monte Carlo sampling. Some of these methods enable training provably robust models [9], [48], [35], [37], [49] by optimizing the model parameters so as to maximize the area of the certified regions.

While effective, all the above-mentioned certification methods, except for randomized smoothing, which gives probabilistic guarantees, only provide binary results given a *fixed* adversarial budget in their vanilla version. That is, if a data point is certified, it is guaranteed to be robust in the entire given adversarial budget; otherwise no guaranteed adversary-free region is estimated. To overcome this and search for the *optimal size of the adversarial budget* that can be certified, [48], [47], [55] use either Newton's method or binary search.

By contrast, [10] takes advantage of the geometric property of ReLU networks and gives more fine-grained robustness guarantees. Based on the piecewise linear nature of the ReLU function, any input is located in a polytope where the network can be considered a linear function. Based on geometry, robustness guarantees can thus be calculated using the input data's distance to the polytope boundary and the decision boundary constraints. Unfortunately, in practice, the resulting certified bounds are trivial because such polytopes are very small even for robust models. Nevertheless, [10] introduces a regularization scheme based on these bounds, models trained using this regularizer are provably robust by other certifiers.

In this paper, we construct a stronger certifier, as well as a regularization scheme to train provably robust models. Instead of relying on the linear regions of the ReLU networks, we estimate a linear bound on the model's output given a predefined adversarial budget. Then, the condition to guarantee robustness inside this budget is also linear and forms a polyhedral envelope of the model's decision boundary. The intersection of the polyhedral envelope and the predefined adversarial budget is then guaranteed to be adversary-free. In contrast to [10], our method can be based on any model linearization method and is thus applicable to general network architectures and activation functions. To train provably robust neural network models, we further introduce a hinge-loss-like regularization term to encourage larger certified bounds. Furthermore, we boost the performance of our method with adversarial training. We also use a stochastic robust approximation [45] to accelerate our method and reduce its memory consumption.

Based on the geometry of the decision boundary, our proposed certification method significantly improves the one in [10] and yields a more accurate estimation of the decision boundary. Furthermore, it is more generally applicable to different activation functions. In contrast to Fast-Lin [47] and CROWN [55], our certification method can prove that a subset of the adversarial budget is adversary-free. We show that such partial credit can accelerate the search for the optimal size of the adversarial budget. On the training side, in contrast to KW [48], [49], which, as pointed out by [54], over-regularizes the model, our proposed method achieves better certified robustness without sacrificing too much clean accuracy. In the remainder of the paper, we refer to our certification method as *Polyhedral Envelope Certifier (PEC)* and to our regularization scheme as *Polyhedral Envelope Regularizer (PER)*.

II. Preliminaries

A. Notation and Terminology

For simplicity, we discuss our approach using a standard N-layer fully-connected network. We will discuss how this formulation can be extended to other architectures in Section II-B. A fully-connected network parameterized by $\{\mathbf{W}^{(i)}, \mathbf{b}^{(i)}\}_{i=1}^{N-1}$ can be expressed as the following equations:

$$\begin{aligned} \mathbf{z}^{(i+1)} &= \mathbf{W}^{(i)}\widehat{\mathbf{z}}^{(i)} + \mathbf{b}^{(i)} \quad && i = 1, 2, ..., N-1 \\ \widehat{\mathbf{z}}^{(i)} &= \sigma(\mathbf{z}^{(i)}) && i = 2, 3, ..., N-1 \end{aligned} \tag{2}$$

where $\mathbf{z}^{(i)}$ and $\widehat{\mathbf{z}}^{(i)}$ are the pre- & post-activations of the i-th layer, respectively, and $\widehat{\mathbf{z}}^{(1)} \stackrel{\text{def}}{=\!=} \mathbf{x}$ is the input of the network. An l_p norm-based adversarial budget $\mathcal{S}_\epsilon^{(p)}(\mathbf{x})$ is defined as the set $\{\mathbf{x}' | \|\mathbf{x}' - \mathbf{x}\|_p \leq \epsilon\}$. $\mathbf{x}'$, $\mathbf{z}'^{(i)}$ and $\widehat{\mathbf{z}}'^{(i)}$ represent the adversarial input and the corresponding pre- & post-activations. For layer i having n_i neurons, we have $\mathbf{W}^{(i)} \in \mathbb{R}^{n_{i+1} \times n_i}$ and $\mathbf{b}^{(i)} \in \mathbb{R}^{n_{i+1}}$. We use $K \stackrel{\text{def}}{=\!=} n_N$ to represent the output dimension.

Throughout this paper, underlines and bars are used to represent lower and upper bounds of the corresponding variables, respectively, i.e., $\underline{\mathbf{z}}^{(i)} \leq \mathbf{z}'^{(i)} \leq \bar{\mathbf{z}}^{(i)}$. A "$+$" or "$-$" subscript indicates the positive or negative elements of a tensor, with all other elements replacing with 0. We use $[K]$ as the abbreviation for the set $\{1, 2, ..., K\}$.

B. Model Linearization

Given an adversarial budget $\mathcal{S}_\epsilon^{(p)}(\mathbf{x})$, we study the linear bound of the output logits $\mathbf{z}'^{(N)}$, given by

$$\mathbf{U}^{(N)}\mathbf{x}' + \mathbf{p}^{(N)} \leq \mathbf{z}'^{(N)} \leq \mathbf{V}^{(N)}\mathbf{x}' + \mathbf{q}^{(N)} \,. \tag{3}$$

The linear coefficients introduced above can be calculated by iteratively estimating the bounds of intermediate layers and linearizing the activation functions. In Appendix A-A, we discuss this for several activation functions, including ReLU, sigmoid and tanh. Note that our method differs from [55] as we need the analytical form of the linear coefficients for training. For example [55] uses some numerical methods such as binary search, while our method does not. The bounding algorithm trades off computational complexity and bound tightness. In this work, we study two such algorithms. One, which we call *CROWN-based bounds*, is based on Fast-Lin / CROWN [47], [55]. It yields tighter bounds but has higher computational complexity. The other, which we call *IBP-inspired bounds*, is inspired by the *Interval Bound Propagation* (IBP) [18]. It is faster but leads to looser bounds. The details of both algorithms are provided in Appendices A-B and A-C, respectively. We discuss the complexity of both algorithms in detail in Section V.

Although the formulation above is based on the fully-connected network, it can be straightforwardly extend to any network whose corresponding computational graph can be represented by a *Directed Acyclic Graph*. All the factors in our bounds, including $\mathbf{U}^{(N)}$, $\mathbf{V}^{(N)}$, $\mathbf{p}^{(N)}$ and $\mathbf{q}^{(N)}$ in (3), can be propagated along the computational graph. This has been shown in detail in Appendix D of [26]. Therefore, our method is also applicable to other network architectures, such as convolutional neural networks (CNN), residual networks (ResNet) and recurrent neural networks (RNN).

III. Algorithms

A. Robustness Guarantees by Polyhedral Envelope

For an input point $\mathbf{x}$ with label $y \in [K]$, a sufficient condition to guarantee robustness is that the lower bounds of $\mathbf{z}_y'^{(N)} - \mathbf{z}_i'^{(N)}$ are positive for all $i \in [K]$. Here, we use the *elision of the last layer* introduced in [18] to merge the subtraction of $\mathbf{z}_y'^{(N)}$ and $\mathbf{z}_i'^{(N)}$ with the last linear layer. Therefore, we obtain the lower bound of $\mathbf{z}_y'^{(N)} - \mathbf{z}_i'^{(N)}$ as a whole: $\underline{\mathbf{z}_y'^{(N)} - \mathbf{z}_i'^{(N)}} \stackrel{\text{def}}{=\!=} \mathbf{U}_i\mathbf{x}' + \mathbf{p}_i$.

Then, the sufficient condition to ensure robustness within a budget $\mathcal{S}_\epsilon^{(p)}(\mathbf{x})$ can be written as the following inequality:

$$\underline{\mathbf{z}_y'^{(N)} - \mathbf{z}_i'^{(N)}} = \mathbf{U}_i\mathbf{x}' + \mathbf{p}_i \geq 0 \quad \forall i \in [K]\,. \tag{4}$$

The constraint is trivial when $i = y$, so there are $(K-1)$ such linear constraints, corresponding to $K-1$ hyperplanes in the input space. Within the adversarial budget, these hyperplanes provide a polyhedral envelope of the true decision boundary. In the remainder of the paper, we use the term d_{iy} to represent the distance between the input and the hyperplane defined in (4) and define $d_y = \min_{i\in[K], i\neq y} d_{iy}$ as the distance between the input and the polyhedral envelope's boundary. The distance can be based on different l_p norms, and $d_{iy} = 0$ when the input itself does not satisfy the inequality (4). Since (4) is a sufficient condition for robustness given the adversarial budget $\mathcal{S}_\epsilon^{(p)}(\mathbf{x})$, it is guaranteed there is no adversarial example in the intersection of $\mathcal{S}_\epsilon^{(p)}(\mathbf{x})$ and the polytope defined in (4).

The lemma below formalizes the vanilla case of our robustness certification, when there are no additional constraints on the input. We defer its proof to Appendix C-A and call our method *Polyhedral Envelope Certification* (PEC).

Lemma 1 (PEC in Unconstrained Cases)**.** *Given a model $f : \mathbb{R}^{n_1} \to [K]$ and an input point $\mathbf{x}$ with label y, let $\mathbf{U}$ and $\mathbf{p}$ in (4) be calculated using a predefined adversarial budget $\mathcal{S}_\epsilon^{(p)}(\mathbf{x})$. Then, there is no adversarial example inside an l_p norm ball of radius d centered around $\mathbf{x}$, with $d = \min\{\epsilon, d_y\}$, where $d_{iy} = \max\left\{0, \frac{\mathbf{U}_i\mathbf{x}+\mathbf{p}_i}{\|\mathbf{U}_i\|_q}\right\}$. l_q is the dual norm of the l_p norm, i.e., $\frac{1}{p} + \frac{1}{q} = 1$.*

Based on Lemma 1, when $\epsilon < d_y$, PEC has the same robustness guarantees as KW [48], Fast-Lin [47] and CROWN [55] using the same model linearization method. When $0 < d_y < \epsilon$, KW / Fast-Lin / CROWN cannot certify the data point at all, while PEC still gives non-trivial robustness guarantees thanks to the geometric interpretability of the polyhedral envelope. Figure 1 compares the certified bounds of KW / Fast-Lin[1] and PEC on a randomly picked input for different values of ϵ in the predefined adversarial budget. We can clearly see the two-phase behavior of both methods. In the second phase, unlike KW / Fast-Lin, PEC still provides a non-trivial certification bound.

Figure 2 shows a 2D sketch of the two phases mentioned above. When ϵ is smaller than a threshold, as in the left half of the figure, the linear bounds in (4) are tight but only valid in a small region $\mathcal{S}_\epsilon^{(p)}(\mathbf{x})$. Therefore, the certified robustness is ϵ at most. When ϵ is bigger than this threshold, the linear bounds are valid in a larger region but becomes inevitably loose. This is because the value of d_{ic} monotonically decreases with the increase of ϵ for all model linearization methods. This is depicted in the right half of the figure, where the distances between the input and the hyperplanes are smaller. The certified robustness is then d_c. The hyperplane segments inside the adversarial budget (green bold lines) never exceed the decision boundary (dark blue bold lines), by definition of the polyhedral envelope. The threshold here is the maximum

[1]In the case of ReLU networks, Fast-Lin and KW are algorithmically the same and yield the same robustness certification.

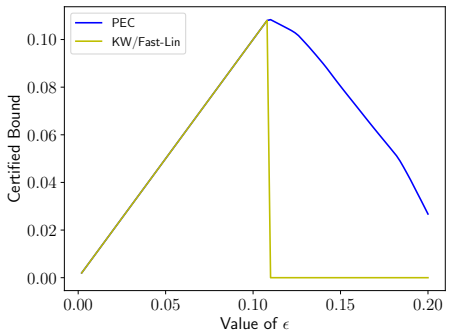


Fig. 1: Certified l_∞-based bound of a randomly picked input by PEC and KW / Fast-Lin for different values of ϵ. The model is the 'FC1' model on MNIST trained by 'MMR+at' in [10]

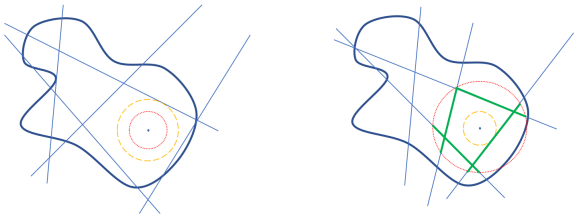


Fig. 2: 2D sketch of decision boundary (dark blue bold lines), hyperplane defined by (4) (light blue lines), adversarial budget (red dotted circle), polyhedral envelope (green bold lines) in PEC. The distance between the input data and the hyperplanes is depicted by a yellow dashed circle. The left and right half correspond to the cases when d_c is bigger and smaller than ϵ, respectively.

certified bound, corresponding to the 'peak' of both curves in Figure 1. We call this threshold *the optimal value of* ϵ.

To search for the optimal value of ϵ, [48] uses Newton's method, which is an expensive second-order method. [47], [55] use binary search to improve efficiency. Thanks to the non-trivial certified bounds in the second phase, our proposed PEC can further accelerate their strategy. During the search, when the value guess $\hat{\epsilon}$ is in the second phase, the vanilla binary search in Fast-Lin / CROWN can only conclude that the optimal value is smaller than $\hat{\epsilon}$. In addition to this upper bound of the optimal value, PEC can output a non-trivial certified bound d_c, in which case we can also conclude that the optimal value is larger than d_c. The tighter lower bound on the optimal value makes PEC need fewer steps to reach the required optimal value precision and thus accelerates the search. We provide more detailed discussion and the pseudo code in Appendix B-A.

In many applications, the input is constrained in a hypercube $[r^{(min)}, r^{(max)}]^{n_1}$. For example, for images with normalized pixel intensities, an attacker will not perturb the image outside the hypercube $[0, 1]^{n_1}$. Such constraint on the attacker allows us to ignore the regions outside the allowable input space, even if they are inside the adversarial budget $\mathcal{S}_\epsilon^{(p)}(\mathbf{x})$.

To obtain robustness guarantees in this scenario, we need to recalculate d_{ic}, which is now the distance between the input and the hyperplanes in (4) within the hypercube. The value of

d_{ic} is then the minimum of the following optimization problem

$$\begin{aligned} &\min_{\Delta} \|\Delta\|_p \\ s.t.\ \ &\mathbf{a}\Delta + b \leq 0, \quad \Delta^{(min)} \leq \Delta \leq \Delta^{(max)} \end{aligned} \tag{5}$$

where, to simplify the notation, we define $\mathbf{a} = \mathbf{U}_i$, $b = \mathbf{U}_i\mathbf{x}+\mathbf{p}$, $\Delta^{(min)} = r^{(min)} - \mathbf{x}$ and $\Delta^{(max)} = r^{(max)} - \mathbf{x}$. When $b \leq 0$, the minimum is obviously 0 as the optimal Δ is an all-zero vector. In this case, either we cannot certify the input at all, or even the clean input is misclassified. When $b > 0$, by Hölder's inequality, $\mathbf{a}\Delta + b \geq -\|\Delta\|_p\|\mathbf{a}\|_q + b$, with equality reached when Δ^p and $\mathbf{a}^q$ are collinear. Based on this, the optimal Δ of minimum l_p norm to satisfy $\mathbf{a}\Delta + b \leq 0$ is

$$\widehat{\Delta}_i = -\frac{b}{\|\mathbf{a}\|_q^q} \text{sign}(\mathbf{a}_i) \, |\mathbf{a}_i|^{\frac{q}{p}} \ , \tag{6}$$

where sign$(\cdot)$ returns $+1$ for positive numbers and -1 for negative numbers.

To satisfy the constraint $\Delta^{(min)} \leq \Delta \leq \Delta^{(max)}$, we use a greedy algorithm that approaches this goal progressively. That is, we first calculate the optimal $\widehat{\Delta}$ based on Equation (6) and check if the constraint $\Delta^{(min)} \leq \Delta \leq \Delta^{(max)}$ is satisfied. For the elements where it is not, we clip their values within $[\Delta^{(min)}, \Delta^{(max)}]$ and keep them fixed. We then optimize the remaining elements of Δ in the next iteration and repeat this process until the constraint is satisfied for all elements. The pseudo-code is provided as Algorithm 1 below and its optimality is guaranteed.

Theorem 1. *If the maximum number of iterations $I^{(max)}$ in Algorithm 1 is large enough to satisfy $\Delta^{(min)} \leq \widehat{\Delta} \leq \Delta^{(max)}$ in Problem* (5)*, then the output $\|\widehat{\Delta}\|_p$ is the optimum of Problem (5), i.e., d_{ic}.*

We can use the primal-dual method to prove Theorem 1, which we defer to Appendix C-B. Once we have the value of d_{ic} and thus d_c, the certified bound in this constrained case is then $\min\{\epsilon, d_c\}$, similar to Lemma 1.

Algorithm 1: Greedy algorithm to solve Problem (5).

Input: $\mathbf{x}$, $\mathbf{a}$, b, $\Delta^{(min)}$, $\Delta^{(max)}$ in (5) and maximum number of iterations allowed $I^{(max)}$
Set of fixed elements $\mathcal{S}^{(f)} = \emptyset$
Iteration number $i = 0$
Calculate $\widehat{\Delta}$ according to (6)
while $\Delta^{(min)} \leq \widehat{\Delta} \leq \Delta^{(max)}$ not satisfied and $i < I^{(max)}$ **do**
  Violated entries $\mathcal{S}^{(v)} = \{i | \widehat{\Delta}_i < \Delta_i^{(min)}$ or $\widehat{\Delta}_i > \Delta_i^{(max)}\}$
  $\widehat{\Delta}_i = \text{clip}(\widehat{\Delta}_i, \text{min} = \Delta_i^{(min)}, \text{max} = \Delta_i^{(max)}), i \in \mathcal{S}^{(v)}$
  $\mathcal{S}^{(f)} = \mathcal{S}^{(f)} \cup \mathcal{S}^{(v)}$
  Update $\widehat{\Delta}$ according to (6) with elements in $\mathcal{S}^{(f)}$ fixed
  Update $i = i + 1$
end while
Output: $\|\widehat{\Delta}\|_p$

If $I^{(max)}$ is set so small that the while-loop breaks with $\Delta^{(min)} \leq \widehat{\Delta} \leq \Delta^{(max)}$ unsatisfied, then the output of Algorithm 1 is the upper bound of Problem (5), and thus we eventually get a suboptimal but still valid robustness guarantee. [11] solves the same problem when designing an attack and points out Algorithm 1 will converge in $O(n_1 \log n_1)$ time. We observed $I^{(max)} = 20$ to be sufficient to satisfy the condition in Theorem 1. In practice, the while-loop breaks within 5 iterations in most cases, which means Algorithm 1 introduces very little overhead.

B. Geometry-Inspired Regularization

As in [10], we can incorporate our certified bounds in Theorem 1 in the training process so as to obtain provably robust models. To this end, we design a regularization term that encourages larger values of d_c. We first introduce the *signed distance* $\tilde{d}_{ic}$: when $d_{ic} > 0$, the clean input satisfies (4) and $\tilde{d}_{ic} = d_{ic}$; when $d_{ic} = 0$, the clean input does not satisfy (4) and there is no certified region; $\tilde{d}_{ic}$ in this case is a negative number whose absolute value is the distance between the input and the hyperplane defined in (4). If the input is unconstrained, we have $\tilde{d}_{ic} = \frac{\mathbf{U}_i\mathbf{x}+\mathbf{p}_i}{\|\mathbf{U}_i\|_q}$. Otherwise, following the notation of (5), $\tilde{d}_{ic} = \text{sign}(b)\|\widehat{\Delta}\|_p$, where $\widehat{\Delta} = \arg\min_{\Delta} \|\Delta\|_p, \ s.t. \ \mathbf{a}\Delta + b = 0, \Delta^{(min)} \leq \Delta \leq \Delta^{(max)}$. This problem can be solved by a greedy algorithm similar to the one in Section III-A.

Now, we sort $\{\tilde{d}_{ic}\}_{i=0,i\neq c}^{K-1}$ as $\tilde{d}_{j_0c} \leq \tilde{d}_{j_1c} \leq ... \leq \tilde{d}_{j_{K-3}c} \leq \tilde{d}_{j_{K-2}c}$ and then define the *Polyhedral Envelope Regularization (PER)* term, based on the smallest T distances, as

$$\text{PER}(\mathbf{x}, \alpha, \gamma, T) = \gamma \sum_{i=0}^{T-1} \max\left(0, 1 - \frac{\tilde{d}_{j_ic}}{\alpha}\right) . \tag{7}$$

Note that, following [10], to accelerate training, we take into account the smallest T distances. When $\tilde{d}_{j_ic} \geq \alpha$, the distance is considered large enough, so the corresponding term is zero and will not contribute to the gradient of the model parameters. This avoids over-regularization and allows us to maintain accuracy on clean inputs. In practice, we do not activate PER in the early training stages, when the model is not well trained and the corresponding polyhedral envelope is meaningless. Such a 'warm up' trick is commonly used in deep learning practice [17].

We can further incorporate PER with adversarial training in a similar way to [10]. Here, the distance $\tilde{d}_{j_ic}$ in (7) is calculated between the polyhedral envelope and the adversarial example generated by PGD [28] instead of the clean input. Note that, the polyhedral envelope is the same in both cases because it only depends on the adversarial budget $\mathcal{S}_\epsilon^{(p)}(\mathbf{x})$. We call this method *PER+at*.

Calculating the polyhedral envelope is expensive in terms of both computation and memory because of the need to obtain linear bounds of the output logits. We conduct a comprehensive complexity analysis in Section V. To prevent such a prohibitive computational and memory overhead, we use the stochastic robust approximation in [45]. For a mini-batch of size B, we only calculate the PER or PER+at regularization term for $B' < B$ instances randomly sub-sampled from this mini-batch. Each instances in the mini-batch has the same probability to be sampled. [30] empirically observed the geometric correlation of high-dimensional decision boundaries near the data manifold. Although this finding is based on regularly trained models, we find it also holds for models trained by

PER / PER+at: in practice, a B' much smaller than B provides a good approximation of the full-batch regularization.

The full pipeline of PER+at method is demonstrated as Algorithm 2. $\mathcal{D}$ and ℓ represent the dataset and the loss function, respectively.

Algorithm 2: Full pipeline of PER+at method

1: **Input:** $\mathcal{D}$, γ, α, T, B, B'
2: Sample $(\mathbf{X}, \mathbf{y}) \in (\mathbb{R}^{B\times m}, [K]^B)$ from the dataset $\mathcal{D}$.
3: Subsample $(\mathbf{X}_s, \mathbf{y}_s) \in (\mathbb{R}^{B'\times m}, [K]^{B'})$ from the minibatch.
4: Using model linearization to calculate $\mathbf{U}$ and $\mathbf{p}$ in Equation (3) for each instance in $(\mathbf{X}_s, \mathbf{y}_s)$.
5: Using PGD attack to generate adversarial examples $(\mathbf{X}', \mathbf{y}')$ of the whole mini-batch, including the subsamples.
6: Calculate PER regularization term based on linearization $\mathbf{U}$, $\mathbf{p}$ and input $(\mathbf{X}'_s, \mathbf{y}'_s)$ using Algorithm 1.
7: The final loss is $\frac{1}{2}(\ell(\mathbf{X}, \mathbf{y}) + \ell(\mathbf{X}', \mathbf{y}')) + \mathrm{PER}(\mathbf{X}'_s, \alpha, \gamma, T)$.
8: Back-propagation and update model parameters.

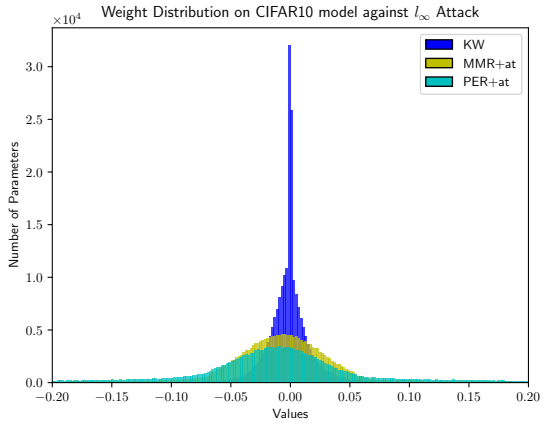


Fig. 3: Parameter value distributions of CIFAR10 models trained against l_∞ attacks. The Euclidean norms of KW, MMR+at, PER+at models against l_∞ attack are 18.08, 38.36 and 94.63 respectively, which evidences that the KW model is over-regularized while our PER model best preserves the model capacity.

IV. Experiments

To validate the theorem and algorithms above, we conducted several experiments on two popular image classification benchmarks: MNIST and CIFAR10. Each of these experiments can be completed on a single NVIDIA TITAN XP GPU machine of 12GB memory within several hours. Our code and checkpoints are publicly available at https://github.com/liuchen11/PolyEnvelope.

A. Training and Certifying ReLU Networks

We first demonstrate the benefits of our approach over existing training and certification methods under the same computational complexity. To this end, we use the same model architectures as in [10], [48]: **FC1**, which is a fully-connected network with one hidden layer of 1024 neurons; and **CNN**, which has two convolutional layers followed by two fully-connected layers. For this set of experiments, all activation functions are ReLU.

When it comes to training, we consider 7 baselines, including plain training (plain), adversarial training (at) [28], KW [48], IBP [18], CROWN-IBP [54], MMR and MMR plus adversarial training (MMR + at) [10]. We denote our method as C-PER, C-PER+at when we use CROWN-style model linearization for PER and PER+at, respectively, and as I-PER and I-PER+at when using IBP-inspired model linearization. We do not compare randomized smoothing [9], [37] or layerwise training [4]. This is because the certified bounds of randomized smoothing are not exact but probabilistic, and layerwise training has significant computational overhead.[2] For fair comparison, we use the same adversarial budget in both the training and the test phases.

To evaluate the models' performance on the test set, we first report the clean test error (CTE) and the empirical robust error against PGD (PGD). Based on the discussions in Section III-A, KW, Fast-Lin and PEC have the same certified robust error, which is the proportion of the input data whose certified regions are smaller than the adversarial budget. Therefore, for these three methods, we report the certified robust error as CRE Lin. We also report the certified robust error by IBP [18]. For l_∞ robustness, we use a complete certifier called MIPVerify [43] to calculate the exact robust error, denoted by CRE MIP. [3] In addition, we calculate the average certified bound obtained by Fast-Lin / KW (ACB Lin)[4], IBP (ACB IBP) and PEC (ACB PEC). Note that the average certified bound here is from the *one-shot* certifier, i.e., without searching for the optimal adversarial budget. We do not report the certified bound obtained by MMR [10], because, in practice, it only gives trivial results. As a matter of fact, [10] emphasize their training method and report certification results using only KW and MIP.

We use the same adversarial budgets and model architectures as [10] and thus directly download the KW, MMR and MMR+at models from the checkpoints provided online.[5] For IBP and CROWN-IBP (C-IBP), we use the same hyper-parameter settings as [54] except that we align the training duration to other methods and the use stochastic robustness approximation of Section III-B to reduce the computational and memory consumption. For CNN models, we use the *warm up* trick consisting of performing adversarial training before adding our PER or PER+at regularization term. The running time overhead of pre-training is negligible compared with computing the regularization term. We train all models for 100 epochs and provide the detailed hyper-parameter settings in Appendix D-A.

We constrain the attacker to perturb the images within $[0, 1]^{n_1}$, and the full results for l_∞ attacks are summarized in Table I. The results of l_2 attacks are demonstrated in Table VIII of Appendix D-B1. For l_∞ attacks, our (C/I)-PER or (C/I)-PER+at achieve the best certified accuracy, calculated by the

[2] For CNN models, [4] trains 200 epochs for each layer and 800 epochs in total, while the other baselines use only 100 epochs. If we reduce the training epochs of each layer to 25 epochs, the model does not converge well. For FC1 models, [4] is the same as KW, because there is only one hidden layer.

[3] MIPVerify is available on https://github.com/vtjeng/MIPVerify.jl

[4] Fast-Lin and KW is algorithmically the same in ReLU networks

[5] https://github.com/max-andr/provable-robustness-max-linear-regions.

Methods	CTE (%)	PGD (%)	CRE Lin (%)	CRE IBP (%)	CRE MIP (%)	ACB Lin	ACB IBP	ACB PEC
MNIST - FC1, ReLU, l_∞, $\epsilon = 0.1$								
plain	1.99	98.37	100.00	100.00	100.00	0.0000	0.0000	0.0000
at	1.42	9.00	97.94	100.00	100.00	0.0021	0.0000	0.0099
KW	2.26	8.59	12.91	69.20	10.90	0.0871	0.0308	0.0928
IBP	1.65	9.67	87.27	**15.20**	12.36	0.0127	**0.0848**	0.0705
C-IBP	1.98	9.50	67.39	**14.45**	11.39	0.0326	**0.0855**	0.0800
MMR	2.11	17.82	33.75	99.88	24.90	0.0663	0.0001	0.0832
MMR+at	2.04	10.39	17.64	95.09	14.10	0.0824	0.0049	0.0905
C-PER	**1.60**	**7.45**	**11.71**	92.89	**7.69**	**0.0883**	0.0071	**0.0935**
C-PER+at	1.81	7.73	12.90	99.90	8.22	0.0871	0.0001	0.0925
I-PER	**1.60**	**6.28**	**11.96**	93.33	**8.10**	**0.0880**	0.0067	**0.0934**
I-PER+at	**1.54**	7.15	13.96	98.55	8.48	0.0868	0.0014	0.0927
MNIST - CNN, ReLU, l_∞, $\epsilon = 0.1$								
plain	1.28	85.75	100.00	100.00	100.00	0.0000	0.0000	0.0000
at	1.02	4.75	91.91	100.00	100.00	0.0081	0.0000	0.0189
KW	1.21	3.03	**4.44**	100.00	4.40	**0.0956**	0.0000	**0.0971**
IBP	1.51	4.43	23.89	**8.13**	5.23	0.0761	**0.0919**	0.0872
C-IBP	1.85	4.28	10.72	**6.91**	4.83	0.0893	**0.0931**	0.0928
MMR	1.65	6.07	11.56	100.00	6.10	0.0884	0.0000	0.0928
MMR+at	1.19	3.35	9.49	100.00	3.60	0.0905	0.0000	0.0939
C-PER	1.44	3.44	5.13	100.00	3.62	0.0949	0.0000	0.0965
C-PER+at	**0.50**	**2.02**	4.85	100.00	**2.21**	0.0952	0.0000	0.0969
I-PER	1.03	2.40	4.64	99.55	2.52	**0.0954**	0.0004	0.0967
I-PER+at	**0.48**	**1.29**	**4.61**	99.94	**1.47**	**0.0954**	0.0001	**0.0971**
CIFAR10 - CNN, ReLU, l_∞, $\epsilon = 2/255$								
plain	24.62	86.29	100.00	100.00	100.00	0.0000	0.0000	0.0000
at	27.04	48.53	85.36	100.00	88.50	0.0011	0.0000	0.0015
KW	39.27	46.60	**53.81**	99.98	**48.00**	**0.0036**	0.0000	**0.0040**
IBP	46.74	56.38	61.81	**67.58**	58.80	0.0030	**0.0025**	0.0034
C-IBP	58.32	63.56	66.28	**69.10**	65.44	0.0026	**0.0024**	0.0029
MMR	34.59	57.17	69.28	100.00	61.00	0.0024	0.0000	0.0032
MMR+at	35.36	49.27	59.91	100.00	54.20	0.0031	0.0000	0.0037
C-PER	39.21	50.98	57.45	99.98	52.70	0.0033	0.0000	0.0038
C-PER+at	**28.87**	**43.55**	**56.59**	100.00	48.43	**0.0034**	0.0000	**0.0040**
I-PER	29.34	51.54	64.34	99.98	54.87	0.0028	0.0000	0.0036
I-PER+at	**26.66**	**43.35**	57.72	100.00	**47.87**	0.0033	0.0000	**0.0040**

TABLE I: Full results of 11 training schemes and 8 evaluation schemes for ReLU networks under l_∞ attacks. The best and the second best results among provably robust training methods (plain and at excluded) are bold. In addition, the best results are underlined.

complete certifier (CRE MIP), in all cases. For l_2 attacks, they also achieve the best estimated certified accuracy, calculated by the Fast-Lin / KW / PEC certifier (CRE Lin), in all cases. In addition, the performance of I-PER and I-PER+at is on par with that of C-PER and C-PER+at, which illustrates that our framework is not sensitive to the tightness of the underlying model linearization method and thus generally applicable.

As observed in previous work [35], different incomplete certifiers are complementary; IBP is only able to certify IBP-trained models and has worse certification results on other models. For the training methods other than IBP and C-IBP, we notice big gaps between the true robustness (CRE MIP) and the IBP certified robustness (CRE IBP). This is because IBP and C-IBP solve a different optimization problem from the other methods. Specifically, IBP and C-IBP do not make any approximation of the activation function, they only utilize the monotonicity of the activation function to propagate the bounds. However, all the other methods use linear approximations to bound the outputs of the activation functions. We also note that the stochastic robustness approximation greatly hurts the performance of IBP and C-IBP on CIFAR10. However, the result reported in [55] without stochastic robustness approximation on the same architecture is still worse than our method.[6] Consistently with Section III-A, our geometry-inspired PEC has better average certified bounds than Fast-Lin / KW given the same adversarial budget. For example, on the CIFAR10 model against l_∞ attack, $10\% - 20\%$ of the test points are not certified by Fast-Lin / KW but have non-trivial bounds with PEC.

When compared with KW, our methods, especially PER+at, have much better clean test accuracy. In other words, a model trained by (C/I)-PER+at is not as over-regularized as other training methods for provable robustness. Figure 3 shows the distribution of parameter values of KW, MMR+at, C-PER+at models on CIFAR10 against l_∞ attacks. The results of CIFAR10 models against the l_2 attack are shown in Figure 8 of Appendix D-B3. As we can see, the parameters of C-PER+at models have much larger norms than KW and MMR+at, whose parameters are more sparse. The norms of the model parameters

[6]The DM-small model in [55] yields a certified robust error of 52.57% on CIFAR10 when $\epsilon = 2/255$.

indicate the model capacity [32], [31], so C-PER+at models better preserve the model capacity.

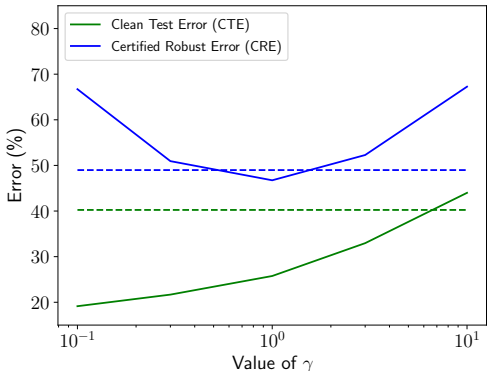


Fig. 4: CTE and CRE for different values of γ in C-PER+at to show their trade-off. The results of KW, for reference, are the horizontal dashed lines. The optimal value of γ for C-PER+at is 1.0, with both CTE and CRE better than KW.

The better performance of (C/I)-PER+at over (C/I)-PER, and of MMR+at over MMR, evidences the benefits of augmenting the training data with the adversarial examples. However, this strategy is only compatible with methods that rely on estimating the distance between the data point and the decision boundary, and thus cannot be combined with methods such as KW. Adding a loss term on the adversarial examples to the loss objective of KW yields a performance between adversarial training and KW. For example, if we optimize the sum of loss objectives of KW and PGD in MNIST - CNN l_∞ cases, the robust error against PGD of the resulting model is 3.64%, the provably robust error by Fast-Lin (CRE Lin) is 8.12%. In other words, such combinations only lead to mixed performance and are weaker than KW in terms of provable robustness.

To demonstrate the trade-off between clean test error and certified robust error, we evaluate our approach with different regularizer strength γ in Equation (7). Figure 4 shows the example of C-PER+at in the l_2 case for CIFAR10. When γ is small, the PER term has little influence on training, and C-PER+at becomes similar to adversarial training (at). It has low clean test error but very high certified robust error. As γ grows, the model is increasingly regularized towards large polyhedral envelopes, which inevitably hurts the performance on the clean input. By contrast, the certified robust error first decreases and then increases. This is because training is numerically more difficult when γ is too large and the model is over-regularized. The results of KW are shown as horizontal dashed lines for comparison. We can see that C-PER+at is in general less over-regularized than KW, with much lower clean test error for the same certified robust error.

To more comprehensively study the performance of our proposed methods, we conduct experiments on larger adversarial budgets. We compare our proposed C-PER+at and I-PER+at with C-IBP [54]. Here, we use the *model architecture E* from [54], consisting of 3 convolutional layers and 2 fully connected layers.[7] We focus on MNIST, where the model can achieve a decent certified robust accuracy under large adversarial budgets, and set ϵ to be 0.1, 0.2, 0.3 and 0.4. For C-IBP, we directly use the publicly available checkpoints from [54]. For (C/I)-PER+at, we use the same settings as the ones in Table I except for the change of adversarial budget. Table III compares the exact certified robust error by MIP (CRE MIP) among the different training methods. The results show that our proposed (C/I)-PER+at yields better results than C-IBP when ϵ is 0.1 and 0.2 but underperforms it when ϵ is 0.3 and 0.4. This phenomenon indicates that our method is more suitable when the adversarial budget is relatively small. This arises from the trade-off between model linearization and interval bound propagation. When the adversarial budget is small, the lower and upper bounds of most ReLU neurons have smaller gaps and are either both negative or both positive. Such neurons can be considered linear and make the model linearization methods, which (C/I)-PER+at is based on, more accurate. With a more accurate bound in the loss function, (C/I)-PER+at outperforms C-IBP, which accumulates the estimation error faster layerwisely [55]. By contrast, a recent study [24] shows that the loss landscape of training methods based on model linearization is less smooth than the ones of IBP and C-IBP. In addition, [25] demonstrates that with the increase of the adversarial budget the loss landscape becomes even more challenging. As a result, when using (C/I)-PER+at with a large adversarial budget, the optimizers cannot find a good minimum. This makes (C/I)-PER+at underperform C-IBP.

B. Training and Certifying Non-ReLU Networks

To validate our method's applicability to non-ReLU networks, we replace the ReLU function in FC1 models with either sigmoid or tanh functions. MMR and MMR+at are no longer applicable here, because they only support piece-wise linear activation functions. MIPVerify does not support sigmoid or tanh functions neither, since it works only on ReLU networks. While [49] claims that their methods apply to non-ReLU networks, their main contribution is rather the extension of KW to a broader set of network architectures, and their public code[8] does not support non-ReLU activations. For evaluation, we replace Fast-Lin and KW with CROWN [55] and thus report its certified robust error (CRE CRO) and average certified bound (ACB CRO). We use the model linearization method in Appendix A-A for (C/I)-PER and (C/I)-PER+at during training, which is slightly different from CROWN. This is because we need an analytical form of the linearization in order to calculate the model parameters' gradients. When we certify models using CROWN, the model linearization method in [55] is used because it is tighter.

The results on l_∞ cases are shown in Table II and the ones on l_2 cases are demonstrated in Table IX of Appendix D-B2. Similar to the ReLU networks in Section IV-A, our (C/I)-PER and (C/I)-PER+at methods have the best performance in all cases, in terms of both certified robust error and average certified bound. IBP can only certify IBP-trained models well and has significantly worse results on other models.

[7]Models available on https://github.com/huanzhang12/CROWN-IBP

[8]Repository: https://github.com/locuslab/convex_adversarial

Methods	CTE (%)	PGD (%)	CRE CRO (%)	CRE IBP (%)	ACB CRO	ACB IBP	ACB PEC
MNIST - FC1, Sigmoid, l_∞, $\epsilon = 0.1$							
plain	2.04	97.80	100.00	100.00	0.0000	0.0000	0.0000
at	1.78	10.05	98.52	100.00	0.0015	0.0000	0.0055
IBP	2.06	10.58	44.14	13.65	0.0559	0.0863	0.0846
C-IBP	2.88	9.83	26.04	**12.51**	0.0740	**0.0875**	0.0886
C-PER	**1.97**	7.55	12.15	84.76	0.0879	0.0152	**0.0930**
C-PER+at	2.16	**7.12**	**11.87**	88.06	**0.0881**	0.0119	0.0927
I-PER	2.15	8.35	12.79	86.99	0.0872	0.0130	0.0926
I-PER+at	2.45	8.05	12.36	88.94	0.0876	0.0111	0.0923
MNIST - FC1, Tanh, l_∞, $\epsilon = 0.1$							
plain	2.00	97.80	100.00	100.00	0.0000	0.0000	0.0000
at	1.28	8.89	99.98	100.00	0.0000	0.0000	0.0001
IBP	**2.04**	9.84	31.81	13.02	0.0682	0.0870	0.0864
C-IBP	2.75	9.57	20.10	**11.80**	0.0799	**0.0882**	0.0894
C-PER	2.19	7.71	11.55	57.81	0.0885	0.0422	**0.0934**
C-PER+at	2.30	**7.45**	**11.39**	56.74	**0.0886**	0.0433	0.0930
I-PER	2.21	8.51	12.23	55.53	0.0878	0.0445	0.0929
I-PER+at	2.46	7.87	12.04	66.04	0.0880	0.0340	0.0929

TABLE II: Full results of 8 training schemes and 7 evaluation schemes for sigmoid and tanh networks under l_∞ attacks. The best results among provably robust training methods (plain and at excluded) are bold and underlined.

Value of ϵ	0.1	0.2	0.3	0.4
C-IBP	3.90	7.25	**11.28**	**18.58**
C-PER+at	**3.52**	7.09	11.34	20.12
I-PER+at	3.58	**7.05**	11.42	21.02

TABLE III: Exact certified robust error by MIP (CRE MIP) of different methods under different sizes of the l_∞ adversarial budget on MNIST. The best results are bold and underlined.

C. Optimal Adversarial Budget

To obtain the biggest certified bound based on the current model linearization method, we need to search for the optimal value of ϵ, i.e., the peak in Figure 1. KW [48] uses Newton's method to solve a constrained optimization problem, which is expensive. Fast-Lin and CROWN [47], [55] apply a binary search strategy to find the optimal ϵ. Based on the discussion in Section III-A, the optimal adversarial budget for a data point is also its optimal certified bound.

To validate the claim in Section III-A that PEC can find the optimal adversarial budget faster than Fast-Lin / CROWN, we compare the average number of iterations needed to find the optimal value given a predefined precision requirement ϵ_Δ. Using $\underline{\epsilon}$ and $\bar{\epsilon}$ to define the initial lower and upper estimates of the optimal value, we then need $\lceil \log_2 \frac{\bar{\epsilon}-\underline{\epsilon}}{\epsilon_\Delta} \rceil$ steps of bound calculations to obtain the optimal value by binary search in Fast-Lin / CROWN. By contrast, the number of bound calculations needed by PEC is smaller and depends on the model to certify, because the partial certified bounds obtained by PEC indicate tighter lower bounds of the optimal adversarial budget. Our experimental results are based on Algorithm 3 presented in Appendix B-A.

We show the results on l_∞ in Table IV and defer the l_2 results in Table X of Appendix D-B5. For l_∞ cases, the original interval $[\underline{\epsilon}, \bar{\epsilon}]$ is $[0, 0.4]$ for MNIST and $[0, 0.1]$ for CIFAR10. Note that, because PEC has almost no computational overhead

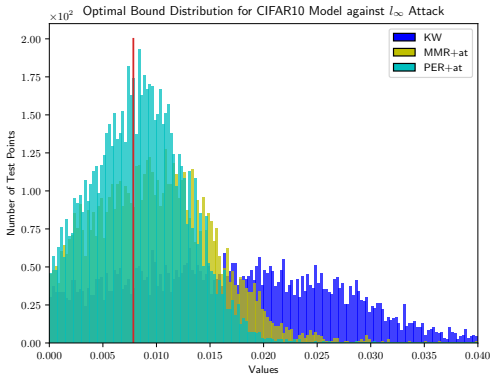


Fig. 5: Distribution of optimal certified bounds of CIFAR10 models trained against l_∞ attacks. The target bound ($2/255$) is indicated by a red vertical line.

compared with Fast-Lin and CROWN,[9] the number of iterations reflects the running time to obtain the optimal certified bounds. Altogether, our results show that PEC can save approximately 25% of the running time for FC1 models and 10% of the running time for CNN models.

Figure 5 shows the distribution of the optimal certified bounds for CIFAR10 models against l_∞ attacks obtained by KW, MMR+at and C-PER+at on the test set. The results on l_2 attacks are shown in Figure 9 of Appendix D-B4. We use vertical red lines to represent the target bounds ($2/255$ in the l_∞ case and 0.1 in the l_2 case), so the area on the right of this line represents the certified robust accuracy. Compared with KW, the mass of C-PER+at is more concentrated on a

[9] We run one-iteration PEC and CROWN to certify CIFAR10-CNN models by C-PER+at for 10 times. To process the entire test set on a single GPU machine, in l_∞ cases, the mean and standard deviation of run time is 217.51 ± 1.95 seconds for CROWN and 219.16 ± 3.23 seconds for PEC; in l_2 cases, it is 236.95 ± 1.64 for CROWN and 239.41 ± 1.92 for PEC. Therefore the difference can be ignored.

Methods	MNIST-FC1, ReLU, l_∞			MNIST-CNN, ReLU, l_∞			CIFAR10-CNN, ReLU, l_∞		
	T_{Lin}	T_{PEC}	$\frac{T_{PEC}}{T_{Lin}}$	T_{Lin}	T_{PEC}	$\frac{T_{PEC}}{T_{Lin}}$	T_{Lin}	T_{PEC}	$\frac{T_{PEC}}{T_{Lin}}$
plain		9.85	0.8207		10.56	0.8804		9.33	0.9331
at		10.77	0.8972		11.39	0.9489		9.12	0.9128
KW		8.48	0.7066		11.61	0.9674		8.43	0.8432
MMR	12	8.04	0.6703	12	10.68	0.8897	10	8.05	0.8053
MMR+at		7.68	0.6402		11.22	0.9351		8.45	0.8450
C-PER		9.34	0.7780		11.17	0.9305		8.61	0.8606
C-PER+at		9.38	0.7816		11.74	0.9784		8.68	0.8681

TABLE IV: Number of steps of bound calculation for the optimal ϵ in Fast-Lin (T_{Lin}) and PEC (T_{PEC}) for ReLU networks under l_∞ attacks. Note that T_{Lin} is a constant for different models given the original interval $[\underline{\epsilon}, \bar{\epsilon}]$.

narrower range on the right of the red line. This evidences that there are significantly fewer points that have unnecessarily large certified bounds for the C-PER+at model than for the KW one. This is because PER+at encourages robustness via a hinge-loss term. When $\tilde{d}_{ic} \geq \alpha$, the regularizer in Equation (7) is a constant zero and does not contribute to the parameter gradient. However, KW first estimates the bound of the worst case output logits and calculates the softmax cross-entropy loss on that. Under this training objective function, each data point is encouraged to make the lower bound of the true label's output logit bigger and the upper bound of false ones smaller, even if the current model is sufficiently robust at this point. This phenomenon also helps to explain why KW tends to over-regulate the model while our methods do not.

We have also tried to replace the cross-entropy loss with the hinge loss in the objective of KW, but observed this not to lead to any improvement over the original KW. This is because KW directly minimizes the gap between the logits of the true and false label, but the logits' magnitude for different instances differs, which makes it difficult or even impossible to set a unified threshold in the hinge loss. By contrast, in PER, we apply the hinge loss to the certified bound directly, which is normalized, easier to interpret and thus makes it much easier to set the threshold in the hinge loss. In practice, the value of α in Equation 7 is set 1.5 times the target adversarial budget.

V. Discussion

Methods	Complexity
PGD	$\mathcal{O}(Nn^2)$
Fast-Lin / CROWN	$\mathcal{O}(N^2n^3)$
KW	$\mathcal{O}(N^2n^3)$
MMR / MMR+at	$\mathcal{O}(Nn^2m)$
IBP	$\mathcal{O}(Nn^2)$
C-IBP	$\mathcal{O}(Nn^3)$
I-PER / I-PER+at	$\mathcal{O}(Nn^2m)$
C-PER / C-PER+at	$\mathcal{O}(N^2n^3)$

TABLE V: Complexity of different methods on an N-layer neural network model with k-dimensional output and m-dimensional input. Each hidden layer has n neurons.

Let us consider an N-layer neural network model with k-dimensional output and m-dimensional input. For simplicity, let each hidden layer have n neurons and usually $n \gg \max\{k, m\}$ is satisfied. In this context, the FLOP complexity of PGD with h iterations is $\mathcal{O}(Nn^2h) \sim \mathcal{O}(Nn^2)$, because typically $h \ll \min\{m, n\}$. Among the methods that train provably robust networks, the linearization algorithm based on Fast-Lin / CROWN needs $\mathcal{O}(N^2n^3)$ FLOPS to obtain the linear bounds of the output logits. However, the complexity can be reduced to $\mathcal{O}(Nn^2m)$ at the cost of bound tightness when we use the IBP-inspired algorithm in Appendix A-C. Note that the IBP-inspired algorithm also calculates the linear bound of the output logits and is thus different from IBP [18], whose complexity is $\mathcal{O}(Nn^2)$, i.e., the same as a forward propagation. MIP is a complete certifier based on mixed integer programming. It solves an NP-hard problem and its complexity is super-polynomial in general. To update the model parameters, KW needs a back-propagation which costs $\mathcal{O}(Nn^2)$ FLOPs. Therefore the complexity of KW is also $\mathcal{O}(N^2n^3)$, with the model linearization dominating the complexity. In CROWN-IBP, the bounds of all intermediate layers are estimated by IBP, which costs $\mathcal{O}(Nn^2)$ FLOPs. The last layer's bound is then estimated in the same way as CROWN, which costs $\mathcal{O}(Nn^3)$ FLOPs, dominating the complexity of CROWN-IBP. For MMR, the complexity to calculate the expression of the input's linear region is $\mathcal{O}(Nn^2m)$. MMR then calculates the distances between the input and $\mathcal{O}(Nn)$ hyper-planes, costing $\mathcal{O}(Nnm)$. Altogether, the complexity of MMR is $\mathcal{O}(Nn^2m)$. MMR+at has the same complexity as MMR, because the overhead of adversarial training can be ignored.

Among our methods, the complexity of the CROWN-style model linearization in C-PER is $\mathcal{O}(N^2n^3)$. Like MMR and KW, the overhead of distance calculation and back-propagation can be ignored. Similarly, the complexity of I-PER is dominated by the IBP-inspired model linearization, which is $\mathcal{O}(Nn^2m)$. Note that C-PER has the same complexity as Fast-Lin, CROWN and KW, and the complexity of I-PER is smaller than that of CROWN-IBP because $m < n$. C-PER+at and I-PER+at have the same complexity as C-PER and I-PER, respectively, since the overhead of adversarial training is negligible. Table V summarizes the complexity of all methods.

No matter which linearization method we use, the bounds of the output logits inevitably become looser for deeper networks, which can be a problem for large models. Furthermore, the linear approximation implicitly favors the l_∞ norm over other l_p norms because the intermediate bounds are calculated in an elementwise manner [26]. As a result, our method performs better in l_∞ cases than in l_2 cases. Designing a training algorithm with scalable and tight certified robustness is highly

non-trivial and worth further exploration.

VI. Conclusion

In this paper, we have studied the robustness of neural networks from a geometric perspective. In our framework, linear bounds are estimated for the model's output under an adversarial budget. Then, the polyhedral envelope resulting from the linear bounds allows us to obtain quantitative robustness guarantees. Our certification method can give non-trivial robustness guarantees to more data points than existing methods and thus speed up the search for the optimal adversarial budget's size. Furthermore, we have shown that our certified bounds can be turned into a geometry-inspired regularization scheme that enables training provably robust models. Compared with existing methods, our framework can be applied to neural networks with general activation functions. In addition to better performance, it can achieve provable robustness at very little loss in clean accuracy.

VII. Acknowledgement

We thankfully acknowledge the support of the Hasler Foundation (Grant No. 16076) for this work.

Appendix A
Model Linearization

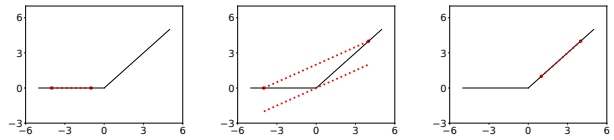
A. Linearization of Activation Functions

In this section, we discuss the choice of d, l and h in the linear approximation $dx + l \leq \sigma(x) \leq dx + h$ for activation function σ when $x \in [\underline{x}, \bar{x}]$. The method used here is slightly different from that of [55]. First, the slope of the linear upper and lower bound is the same, because this can save up to $3/4$ memory when calculating the slope of the linear bound. Second, all coefficients need to have an analytical form because we need to calculate the gradient based on them during training. Note that [55] use binary search to obtain the optimal d, l, h for general activation functions.

1) ReLU: As Figure 6 shows, the linear approximation for ReLU $\sigma(x) = \max(0, x)$, which is convex, is:

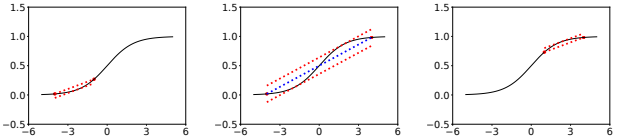
$$d = \begin{cases} 0 & \underline{x} \leq \bar{x} \leq 0 \\ \dfrac{\bar{x}}{\bar{x} - \underline{x}} & \underline{x} < 0 < \bar{x} \\ 1 & 0 \leq \underline{x} \leq \bar{x} \end{cases},$$
$$l = 0 \ \forall x \in \mathbb{R}, \tag{8}$$
$$h = \begin{cases} 0 & \underline{x} \leq \bar{x} \leq 0 \\ -\dfrac{\underline{x}\bar{x}}{\bar{x} - \underline{x}} & \underline{x} < 0 < \bar{x} \\ 0 & 0 \leq \underline{x} \leq \bar{x} \end{cases}.$$

2) Sigmoid, Tanh: Unlike the ReLU function, the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ and tanh function $\sigma(x) = \frac{e^{2x}-1}{e^{2x}+1}$ are not convex. However, these two functions are convex when $x < 0$ and concave when $x > 0$ (left and right sub-figures of Figure 7). Therefore, when $\underline{x} \leq \bar{x} \leq 0$ or $0 \leq \underline{x} \leq \bar{x}$, we can easily obtain a tight linear approximation. When $\underline{x} \leq 0 \leq \bar{x}$, we do not use the binary research to obtain a tight linear approximation as in [55], because the results would not have an analytical form in this way. Instead, we first calculate the slope between the two ends, i.e., $d = \frac{\sigma(\bar{x}) - \sigma(\underline{x})}{\bar{x} - \underline{x}}$. Then, we bound the function by two tangent lines of the same slope as d (middle sub-figure of Figure 7).



(a) $\underline{x} \leq \bar{x} \leq 0$ (b) $\underline{x} < 0 < \bar{x}$ (c) $0 \leq \underline{x} \leq \bar{x}$

Fig. 6: Linearization of the ReLU function in all scenarios.



(a) $\underline{x} \leq \bar{x} \leq 0$ (b) $\underline{x} < 0 < \bar{x}$ (c) $0 \leq \underline{x} \leq \bar{x}$

Fig. 7: Linearization of the sigmoid function in all scenarios.

For sigmoid and tanh, we can calculate the coefficients of the linear approximation as

$$d = \frac{\sigma(\bar{x}) - \sigma(\underline{x})}{\bar{x} - \underline{x}}.,$$
$$l = \begin{cases} \sigma(t_1) - t_1 d & \underline{x} < 0 \\ \dfrac{\bar{x}\sigma(\underline{x}) - \underline{x}\sigma(\bar{x})}{\bar{x} - \underline{x}} & 0 \leq \underline{x} \leq \bar{x} \end{cases}, \tag{9}$$
$$h = \begin{cases} \dfrac{\bar{x}\sigma(\underline{x}) - \underline{x}\sigma(\bar{x})}{\bar{x} - \underline{x}} & \underline{x} \leq \bar{x} \leq 0 \\ \sigma(t_2) - t_2 d & 0 < \bar{x} \end{cases}.$$

The coefficients $t_1 < 0 < t_2$ are the position of tangent points on both sides of the origin. The definitions of t_1 and t_2 for different activation functions are provided in Table VI.

σ	Sigmoid	Tanh
t_1	$-\log \frac{-(2d-1)+\sqrt{1-4d}}{2d}$	$\frac{1}{2}\log \frac{-(d-2)-2\sqrt{1-d}}{d}$
t_2	$-\log \frac{-(2d-1)-\sqrt{1-4d}}{2d}$	$\frac{1}{2}\log \frac{-(d-2)+2\sqrt{1-d}}{d}$

TABLE VI: Definition of t_1 and t_2 for different activation functions.

B. CROWN-style Bounds

Based on the linear approximation of activation functions in Section A-A, we have $\mathbf{D}^{(i)}\mathbf{z}'^{(i)} + \mathbf{l}^{(i)} \leq \sigma(\mathbf{z}'^{(i)}) \leq \mathbf{D}^{(i)}\mathbf{z}'^{(i)} + \mathbf{u}^{(i)}$ where $\mathbf{D}^{(i)}$ is a diagonal matrix and $\mathbf{l}^{(i)}$, $\mathbf{u}^{(i)}$ are vectors. We can rewrite this formulation as follows:

$$\begin{aligned} &\exists \mathbf{D}^{(i)}, \mathbf{l}^{(i)}, \mathbf{u}^{(i)} : \ \forall \mathbf{z}'^{(i)} \in [\underline{\mathbf{z}}^{(i)}, \ \bar{\mathbf{z}}^{(i)}], \\ &\text{then } \exists \mathbf{m}^{(i)} \in [\mathbf{l}^{(i)}, \mathbf{u}^{(i)}] \ s.t. \sigma(\mathbf{z}'^{(i)}) = \mathbf{D}^{(i)}\mathbf{z}'^{(i)} + \mathbf{m}^{(i)} . \end{aligned} \tag{10}$$

We plug (10) into (2), and the expression of $\mathbf{z}'^{(i)}$ can be rewritten as

$$
\begin{aligned}
&\mathbf{z}'^{(i)} = \mathbf{W}^{(i-1)}(\sigma(\mathbf{W}^{(i-2)}(...\sigma(\mathbf{W}^{(1)}\hat{\mathbf{z}}'^{(1)} + \mathbf{b}^{(1)})...) + \mathbf{b}^{(i-2)})) + \mathbf{b}^{(i-1)} \\
&= \mathbf{W}^{(i-1)}(\mathbf{D}^{(i-1)}(\mathbf{W}^{(i-2)}(...\mathbf{D}^{(2)}(\mathbf{W}^{(1)}\mathbf{x}' + \mathbf{b}^{(1)}) + \mathbf{m}^{(2)}...) \\
&+ \mathbf{b}^{(i-2)}) + \mathbf{m}^{(i-1)}) + \mathbf{b}^{(i-1)} \\
&= \left(\prod_{k=1}^{i-1} \mathbf{W}^{(k)}\mathbf{D}^{(k)}\right)\mathbf{W}^{(1)}\mathbf{x}' + \sum_{j=1}^{i-1}\left(\prod_{k=j+1}^{i-1} \mathbf{W}^{(k)}\mathbf{D}^{(k)}\right)\mathbf{b}^{(j)} \\
&+ \sum_{j=2}^{i-1}\left(\prod_{k=j+1}^{i-1} \mathbf{W}^{(k)}\mathbf{D}^{(k)}\right)\mathbf{W}^{(j)}\mathbf{m}^{(j)} \ .
\end{aligned} \tag{11}
$$

This is a linear function w.r.t. $\mathbf{x}'$ and $\{\mathbf{m}^{(j)}\}_{j=2}^{i-1}$. Once given the perturbation budget $\mathcal{S}_\epsilon^{(p)}(\mathbf{x})$ and the bounds of $\{\mathbf{m}^{(j)}\}_{j=2}^{i-1}$, we can calculate the slope and the bias term in the linear bound of $\mathbf{z}'^{(i)}$ in Equation (11). This process can be repeated until we obtain the bound of the output logits in Equation (3). The derivation here is the same as in [26], [47], we encourage interested readers to check these works for details.

C. IBP-inspired Bounds

Interval Bound Propagation (IBP), introduced in [18], is a simple and scalable method to estimate the bounds of each layer in neural networks. IBP is much faster than the algorithm introduced in Appendix A-B because the bounds of any intermediate layer are calculated only based on the information of its immediate previous layer. Therefore, the bounds are propagated just like inference in network models, which costs only $\mathcal{O}(N)$ matrix-vector multiplications for an N-layer network defined in (2).

In our work, we need linear bounds of the output logits in addition to general numeric bounds, so the linearization of activation functions defined in (10) is necessary. We define linear bounds $\mathbf{U}^{(i)}\mathbf{x}' + \mathbf{p}^{(i)} \leq \mathbf{z}'^{(i)} \leq \mathbf{U}^{(i)}\mathbf{x}' + \mathbf{q}^{(i)}$, $\widehat{\mathbf{U}}^{(i)}\mathbf{x}' + \hat{\mathbf{p}}^{(i)} \leq \hat{\mathbf{z}}'^{(i)} \leq \widehat{\mathbf{U}}^{(i)}\mathbf{x}' + \hat{\mathbf{q}}^{(i)}$. We use the same slope as in Section A-A to linearize the activation functions, so the slopes of both bounds are the same. Plugging (10) into this formulation, we have

$$
\begin{aligned}
\widehat{\mathbf{U}}^{(i)} &= \mathbf{D}^{(i)}\mathbf{U}^{(i)}, \\
\hat{\mathbf{p}}^{(i)} &= \mathbf{D}^{(i)}\mathbf{p}^{(i)} + \mathbf{l}^{(i)}, \\
\hat{\mathbf{q}}^{(i)} &= \mathbf{D}^{(i)}\mathbf{q}^{(i)} + \mathbf{u}^{(i)} \ .
\end{aligned} \tag{12}
$$

Here, we assume that the activation functions are monotonically increasing, so the elements in $\mathbf{D}^{(i)}$ are non-negative. Similarly, by comparing the linear bounds of $\hat{\mathbf{z}}'^{(i)}$ and $\mathbf{z}'^{(i+1)}$, we have

$$
\begin{aligned}
\mathbf{U}^{(i+1)} &= \mathbf{W}^{(i)}\widehat{\mathbf{U}}^{(i)}, \\
\mathbf{p}^{(i+1)} &= \mathbf{W}_+^{(i)}\hat{\mathbf{p}}^{(i)} + \mathbf{W}_-^{(i)}\hat{\mathbf{q}}^{(i)} + \mathbf{b}^{(i)}, \\
\mathbf{q}^{(i+1)} &= \mathbf{W}_+^{(i)}\hat{\mathbf{q}}^{(i)} + \mathbf{W}_-^{(i)}\hat{\mathbf{p}}^{(i)} + \mathbf{b}^{(i)} \ .
\end{aligned} \tag{13}
$$

By definition, we have $\widehat{\mathbf{U}}^{(1)} = \mathbf{I}$ and $\hat{\mathbf{p}}^{(1)} = \hat{\mathbf{q}}^{(1)} = \mathbf{0}$. Applying Equation (12) and (13) iteratively allows us to obtain the values of the coefficients $\mathbf{U}^{(N)}$, $\mathbf{V}^{(N)}$, $\mathbf{p}^{(N)}$ and $\mathbf{q}^{(N)}$ in Equation (3).

APPENDIX B
ALGORITHMS

A. Algorithms for Searching the Optimal Value of ϵ

The pseudo code for finding the optimal ϵ is provided as Algorithm 3 below. $\mathcal{M}$, $\mathbf{x}$, ϵ_Δ, $\underline{\epsilon}$, $\bar{\epsilon}$ represent the classification model, the input point, the precision requirement, the predefined estimate of the lower bound and of the upper bound, respectively. Typically, $\underline{\epsilon}$ is set to 0 and $\bar{\epsilon}$ is set to a large value corresponding to a perceptible the image perturbation. f is a function mapping a model, an input point and a value of ϵ to a certified bound. f is a generalized form of the Fast-Lin, CROWN and PEC algorithms.

During the search for the optimal ϵ, the lower bound is updated by the current certified bound, while the upper bound is updated when the current certified bound is smaller than the choice of ϵ. In Fast-Lin and CROWN, we update either the lower or the upper bound in one iteration since the certified bound is either 0 or the current choice of ϵ. However, it is possible for PEC to update both the lower and the upper bounds in one iteration, and this leads to a faster convergence of ϵ.

Algorithm 3: Search for optimal value of ϵ

Input: $\mathbf{x}$, $\underline{\epsilon}$, $\bar{\epsilon}$, ϵ_Δ, f, $\mathcal{M}$
Set the bounds of ϵ: $\epsilon_{up} = \bar{\epsilon}$, $\epsilon_{low} = \underline{\epsilon}$
while $\epsilon_{up} - \epsilon_{low} > \epsilon_\Delta$ **do**
  $\epsilon_{try} = \frac{1}{2}(\epsilon_{low} + \epsilon_{up})$
  $\epsilon_{cert} = f(\mathcal{M}, \mathbf{x}, \epsilon_{try})$
  Update lower bound: $\epsilon_{low} = \max\{\epsilon_{low}, \epsilon_{cert}\}$
  if $\epsilon_{try} > \epsilon_{cert}$ **then**
    Update upper bound: $\epsilon_{up} = \epsilon_{try}$
  end if
end while
Output: $\frac{1}{2}(\epsilon_{low} + \epsilon_{up})$

APPENDIX C
PROOFS

A. Proof of Lemma 1

Proof. Let $\mathbf{x}' = \mathbf{x} + \Delta$ be a point that breaks condition (4). Then,

$$
\begin{aligned}
& \mathbf{U}_i(\mathbf{x} + \Delta) + \mathbf{p}_i && < 0 \\
\iff\ & \mathbf{U}_i\Delta && < -\mathbf{U}_i\mathbf{x} - \mathbf{p}_i \\
\implies\ & -\|\mathbf{U}_i\|_q\|\Delta\|_p && < -\mathbf{U}_i\mathbf{x} - \mathbf{p}_i \\
\iff\ & \|\Delta\|_p && > \frac{\mathbf{U}_i\mathbf{x} + \mathbf{p}_i}{\|\mathbf{U}_i\|_q}
\end{aligned} \tag{14}
$$

The $\implies$ comes from Hölder's inequality. (14) indicates that a perturbation of l_p norm over $d_{ic} = \max\left\{0, \frac{\mathbf{U}_i\mathbf{x}+\mathbf{p}_i}{\|\mathbf{U}_i\|_q}\right\}$ is needed to break the sufficient condition of $\mathbf{z}_c'^{(N)} - \mathbf{z}_i'^{(N)} \geq 0$. Based on the assumption of adversarial budget $\mathcal{S}_\epsilon^{(p)}(\mathbf{x})$ when linearizing the model, the l_p norm of a perturbation to produce an adversarial example is at least $\min\{\epsilon, d_c\}$. □

B. Proof of Theorem 1

Proof. We use the primal-dual method to solve the optimization problem (5), which is a convex optimization problem with linear constraints.

It is clear that there exists an image inside the allowable pixel space for which the model predicts the wrong label. That is, the constrained problem (5) is strictly feasible:

$$\exists \Delta \ s.t. \ \mathbf{a}\Delta + b < 0, \Delta^{(min)} < \Delta < \Delta^{(max)} \ . \tag{15}$$

Thus, this convex optimization problem satisfies *Slater's Condition*, i.e., strong duality holds. We then rewrite the primal problem as

$$\begin{aligned} &\min_{\Delta^{(min)} \leq \Delta \leq \Delta^{(max)}} \|\Delta\|_p^p \\ &s.t. \ \ \mathbf{a}\Delta + b \leq 0 \end{aligned} \tag{16}$$

We minimize $\|\Delta\|_p^p$ instead of directly $\|\Delta\|_p$ in order to decouple all elements in vector Δ. In addition, we consider $\Delta^{(min)} \leq \Delta \leq \Delta^{(max)}$ as the domain of Δ instead of constraints for simplicity. We write the dual problem of (16) by introducing a coefficient of relaxation $\lambda \in \mathbb{R}_+$:

$$\max_{\lambda \geq 0} \min_{\Delta^{(min)} \leq \Delta \leq \Delta^{(max)}} g(\Delta, \lambda) \stackrel{\text{def}}{=\!=} \|\Delta\|_p^p + \lambda(\mathbf{a}\Delta + b) \tag{17}$$

To solve the inner minimization problem, we set the gradient $\frac{\partial g(\Delta,\lambda)}{\partial \Delta_i} = \text{sign}(\Delta_i) p |\Delta_i|^{p-1} + \lambda \mathbf{a}_i$ to zero and obtain $\Delta_i = -\text{sign}(\mathbf{a}_i) \left| \frac{\lambda \mathbf{a}_i}{p} \right|^{\frac{1}{p-1}}$. Based on the convexity of function $g(\Delta, \lambda)$ w.r.t. Δ, we can obtain the optimal $\tilde{\Delta}_i$ in the domain:

$$\tilde{\Delta}_i = \text{clip}\left(-\text{sign}(\mathbf{a}_i) \left| \frac{\lambda \mathbf{a}_i}{p} \right|^{\frac{1}{p-1}}, \text{min} = \Delta_i^{(min)}, \text{max} = \Delta_i^{(max)} \right) . \tag{18}$$

Based on strong duality, we can say that the optimal $\tilde{\Delta}$ is chosen by setting a proper value of λ. Fortunately, $\|\tilde{\Delta}\|_p$ increases monotonically with λ, so the smallest λ corresponds to the optimum.

As we can see, the expression of $\widehat{\Delta}$ in (6) is consistent with $\tilde{\Delta}_i$ in (18) if λ is set properly.[10] The greedy algorithm in Algorithm 1 describes the process of gradually increasing λ to find the smallest value satisfying the constraint $\mathbf{a}\Delta + b \leq 0$. With the increase of λ, the elements in vector Δ remain unchanged when they reach either $\Delta^{(min)}$ or $\Delta^{(max)}$, so we keep such elements fixed and optimize the others.

□

APPENDIX D
ADDITIONAL EXPERIMENTS

A. Details of the Experiments

1) Model Architecture: The FC1 and CNN networks used in this paper are identical to the ones used in [10]. The FC1 network is a fully-connected network with one hidden layer of 1024 neurons. The CNN network has two convolutional layers and one additional hidden layer before the output layer. Both convolutional layers have a kernel size of 4, a stride of 2 and a padding of 1 on both sides, so the height and width of the feature maps are halved after each convolutional layer. The first convolutional layer has 32 channels while the second one has 16. The hidden layer following them has 100 neurons.

2) Hyper-parameter Settings: In all experiments, we use the Adam optimizer [23] with an initial learning rate of 10^{-3} and train all models for 100 epochs with a mini-batch of 100 instances. For CNN models, we decrease the learning rate to 10^{-4} for the last 10 epochs. When we train CNN models on MNIST, we only calculate the polyhedral envelope of 20 instances subsampled from each mini-batch. When we train CNN models on CIFAR10, this subsampling number is 10. These settings make our algorithm possible to be trained on a GPU with 12 GB memory. For PER and PER+at, the value of T is always 4. We search in the logarithmic scale for the value of γ and in the linear scale for the value of α. For ϵ, we ensure that its values in the end of training are close to the ones used in the adversarial budget $\mathcal{S}_\epsilon^{(p)}(\mathbf{x})$. We compare constant values with an exponential growth scheme for ϵ but always use constant values for α and γ. The optimal values we found for different settings are provided in Table VII.

Task	α	ϵ	γ
MNIST FC1, l_∞	0.15	initial value 0.0064 $\times 2$ every 20 epochs	0.1
MNIST CNN, l_∞	0.15	0.1	PER: 0.3 PER+at: 0.03
CIFAR10 CNN, l_∞	0.1	0.008	PER: 0.0003 PER+at: 0.001
MNIST FC1, l_2	0.45	initial value 0.02 $\times 2$ every 20 epochs	1.0
MNIST CNN, l_2	0.45	0.3	1.0
CIFAR10 CNN, l_2	0.15	0.1	PER: 0.3 PER+at: 1.0

TABLE VII: Values of α, ϵ and γ for different experiments.

B. Additional Experimental Results

1) l_2 Robustness on ReLU Networks: The results of 11 training methods and 7 evaluation metrics on l_2 robustness are provided in Table VIII. In all three cases studied, our proposed methods, either PER or PER+at, achieves the best performance.

2) l_2 Robustness on non-ReLU Networks: The results of 8 training methods and 7 evaluation metrics on l_2 robustness in the case of Non-ReLU networks are provided in Table IX. Consider the best certified robust accuracy for each model, we can clearly see that our proposed PER and PER+at achieve the best performance. Meanwhile, its clean accuracy is also better than the baselines. In addition, IBP can only give good performance on IBP-trained models. These observations are the same as the l_∞ cases.

3) Parameter Value Distribution: The parameter value distributions of CIFAR10 models against l_2 attacks are provided in Figure 8. Same as the l_∞ cases, the parameters of the PER+at model have significantly larger norms, indicating it better utilize the model's capacity.

[10] The power term $\frac{q}{p} = \frac{1}{p-1}$ when $\frac{1}{p} + \frac{1}{q} = 1$

Methods	CTE (%)	PGD (%)	CRE Lin (%)	CRE IBP (%)	ACB Lin	ACB IBP	ACB PEC
MNIST - FC1, ReLU, l_2, $\epsilon = 0.3$							
plain	1.99	9.81	40.97	99.30	0.1771	0.0021	0.2300
at	1.35	2.99	14.85	99.23	0.2555	0.0023	0.2684
KW	1.23	2.70	**4.91**	41.55	**0.2853**	0.1754	**0**.2892
IBP	1.36	2.90	6.87	**9.01**	0.2794	**0.2730**	0.2876
C-IBP	1.26	2.80	6.36	**8.73**	0.2809	**0.2738**	0.2884
MMR	2.40	5.88	7.76	99.55	0.2767	0.0013	0.2845
MMR+at	1.77	3.76	5.68	99.86	0.2830	0.0004	0.2880
C-PER	1.26	2.44	5.35	59.17	0.2840	0.1225	0.2888
C-PER+at	**0.67**	**1.40**	**4.84**	64.79	**0.2855**	0.1056	**0.2910**
I-PER	1.21	2.59	5.34	54.13	0.2840	0.1376	0.2888
I-PER+at	**0.74**	**1.46**	7.81	72.85	0.2766	0.0814	0.2860
MNIST - CNN, ReLU, l_2, $\epsilon = 0.3$							
plain	1.28	4.93	100.00	100.00	0.0000	0.0000	0.0000
at	1.12	2.50	100.00	100.00	0.0000	0.0000	0.0000
KW	1.11	2.05	5.84	100.00	0.2825	0.0000	0.2861
IBP	2.37	3.85	51.12	**11.73**	0.1534	**0.2648**	0.1669
C-IBP	2.89	4.44	31.62	**12.29**	0.2051	**0.2631**	0.2178
MMR	2.57	5.49	10.03	100.00	0.2699	0.0000	0.2788
MMR+at	1.73	3.22	9.46	100.00	0.2716	0.0000	0.2780
C-PER	1.02	1.87	**5.04**	100.00	**0.2849**	0.0000	**0.2882**
C-PER+at	**0.43**	**0.91**	**5.43**	100.00	**0.2837**	0.0000	**0.2878**
I-PER	1.11	2.16	6.37	100.00	0.2809	0.0000	0.2851
I-PER+at	**0.52**	**1.12**	7.89	100.00	0.2763	0.0000	0.2812
CIFAR10 - CNN, ReLU, l_2, $\epsilon = 0.1$							
plain	23.29	47.39	100.00	100.00	0.0000	0.0000	0.0000
at	25.84	35.81	99.96	100.00	0.0000	0.0000	0.0000
KW	40.24	43.87	48.98	100.00	0.0510	0.0000	0.0533
IBP	57.90	60.03	64.78	**78.13**	0.0352	**0.0219**	0.0366
C-IBP	71.21	72.51	76.23	**80.97**	0.0238	**0.0190**	0.0256
MMR	40.93	50.57	57.07	100.00	0.0429	0.0000	0.0480
MMR+at	37.78	43.98	53.33	100.00	0.0467	0.0000	0.0502
C-PER	34.10	52.54	63.42	100.00	0.0369	0.0000	0.0465
C-PER+at	**25.76**	**33.47**	**46.74**	100.00	**0.0533**	0.0000	**0.0580**
I-PER	33.94	43.06	56.80	100.00	0.0432	0.0000	0.0484
I-PER+at	**24.85**	**31.32**	**47.28**	100.00	**0.0528**	0.0000	**0.0572**

TABLE VIII: Full results of 11 training schemes and 7 evaluation schemes for ReLU networks under l_2 attacks. The best and the second best results among provably robust training methods (plain and at excluded) are bold. In addition, the best results are underlined.

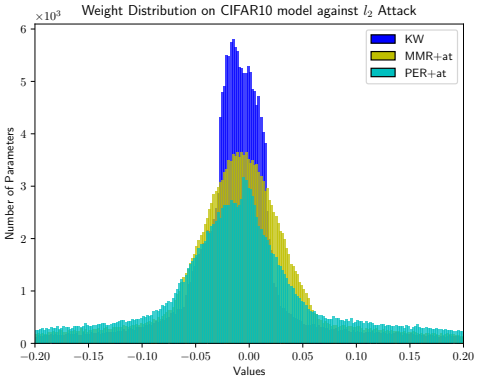


Fig. 8: Parameter value distributions of CIFAR10 models trained against l_2 attack. The Euclidean norms of KW, MMR+at, PER+at model against l_2 attacks are 71.34, 62.97 and 141.77, respectively.

4) Distribution of the Optimal Bounds: The distribution of optimal certified bounds of CIFAR10 models against l_2 attacks is shown in Figure 9. Compared with KW and MMR+at, the values of the optimal certified bounds of the PER+at model are more concentrated in a region slightly better than the required bounds (indicated by the red vertical line). On the contrary, the KW model usually has unnecessarily large certified bounds on some input instances, indicating over-regularization.

5) Searching for the Optimal Value of ϵ: Table X shows the number of bound calculations in the binary search for the optimal ϵ in PEC and Fast-Lin under l_2 attacks. The original interval $[\underline{\epsilon}, \bar{\epsilon}]$ is $[0, 1.2]$ for MNIST and $[0, 0.4]$ for CIFAR10. The bound of the number calculation does not depend on the model in Fast-Lin and is model-dependent in PEC as discussed in Section IV-C.

REFERENCES

[1] Jean-Baptiste Alayrac, Jonathan Uesato, Po-Sen Huang, Alhussein Fawzi, Robert Stanforth, and Pushmeet Kohli. Are labels required for improving adversarial robustness? In *Advances in Neural Information Processing Systems*, pages 12214–12223, 2019.

[2] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *ECCV*, 2020.

Methods	CTE (%)	PGD (%)	CRE CRO (%)	CRE IBP (%)	ACB CRO	ACB IBP	ACB PEC
MNIST - FC1, Sigmoid, l_2, $\epsilon = 0.3$							
plain	2.01	10.25	30.78	94.82	0.2077	0.0155	0.2539
at	1.65	3.48	7.50	85.84	0.2775	0.0422	0.2839
IBP	1.40	3.07	6.43	9.13	0.2807	0.2726	0.2873
C-IBP	1.51	3.24	6.36	**8.73**	0.2709	**0.2738**	0.2872
C-PER	1.36	2.58	6.12	73.71	0.2816	0.0789	0.2867
C-PER+at	**0.46**	**1.03**	5.26	68.94	0.2842	0.0932	0.2905
I-PER	1.19	2.59	6.05	70.18	0.2818	0.0895	0.2871
I-PER+at	0.49	1.16	**5.03**	65.79	**0.2849**	0.1026	**0.2907**
MNIST - FC1, Tanh, l_2, $\epsilon = 0.3$							
plain	1.94	16.46	61.66	99.64	0.1150	0.0011	0.1789
at	1.36	3.02	12.35	97.66	0.2630	0.0070	0.2735
IBP	1.57	3.17	7.21	10.44	0.2784	0.2688	0.2851
C-IBP	1.50	3.14	6.64	**9.53**	0.2801	**0.2714**	0.2861
C-PER	1.31	2.47	**5.53**	55.17	**0.2834**	0.1345	0.2880
C-PER+at	0.58	1.30	5.89	54.88	0.2823	0.1354	0.2885
I-PER	1.38	2.85	5.90	45.31	0.2823	0.1641	0.2874
I-PER+at	**0.55**	**1.17**	5.57	53.73	0.2833	0.1388	**0.2890**

TABLE IX: Full results of 8 training schemes and 7 evaluation schemes for sigmoid and tanh networks under l_2 attacks. The best results among provably robust training methods (plain and at excluded) are bold and underlined.

Methods	**MNIST-FC1, ReLU, l_2**			**MNIST-CNN, ReLU, l_2**			**CIFAR10-CNN, ReLU, l_2**		
	T_{Lin}	T_{PEC}	$\frac{T_{PEC}}{T_{Lin}}$	T_{Lin}	T_{PEC}	$\frac{T_{PEC}}{T_{Lin}}$	T_{Lin}	T_{PEC}	$\frac{T_{PEC}}{T_{Lin}}$
plain		9.68	0.6914		13.64	0.9742		11.73	0.9775
at		10.44	0.7457		13.76	0.9829		11.67	0.9725
KW		7.72	0.5514		12.63	0.9021		10.23	0.8525
MMR	14	5.86	0.4186	14	8.52	0.6086	12	9.05	0.7542
MMR+at		5.91	0.4221		12.13	0.8664		10.33	0.8608
C-PER		11.47	0.8194		13.75	0.9819		9.13	0.7609
C-PER+at		11.34	0.8100		13.72	0.9796		10.71	0.8926

TABLE X: Number of steps of bound calculation for the optimal ϵ in Fast-Lin (T_{Lin}) and PEC (T_{PEC}) for ReLU networks under l_2 attacks. Note that T_{Lin} is a constant for different models given the original interval $[\underline{\epsilon}, \bar{\epsilon}]$.

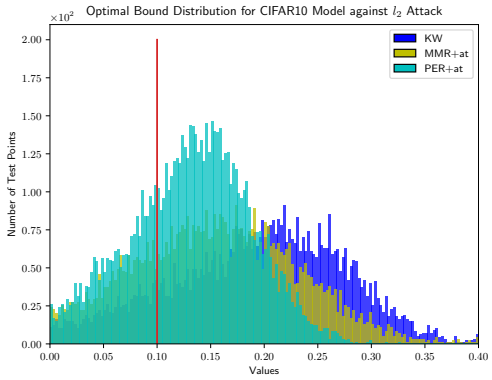


Fig. 9: Distribution of optimal certified bounds of CIFAR10 models trained against l_2 attacks. The target bound (0.1) is marked as a red vertical line.

[3] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning*, pages 274–283, 2018.

[4] Mislav Balunovic and Martin Vechev. Adversarial training and provable defenses: Bridging the gap. In *International Conference on Learning Representations*, 2020.

[5] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. In *International Conference on Learning Representations*, 2018.

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

[7] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S Liang. Unlabeled data improves adversarial robustness. In *Advances in Neural Information Processing Systems*, pages 11190–11201, 2019.

[8] Minhao Cheng, Thong Le, Pin-Yu Chen, Huan Zhang, JinFeng Yi, and Cho-Jui Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. In *International Conference on Learning Representations*, 2019.

[9] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pages 1310–1320, 2019.

[10] Francesco Croce, Maksym Andriushchenko, and Matthias Hein. Provable robustness of relu networks via maximization of linear regions. In *the 22nd International Conference on Artificial Intelligence and Statistics*, pages 2057–2066, 2019.

[11] Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning*, pages 2196–2205. PMLR, 2020.

[12] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, 2020.

[13] Guneet S. Dhillon, Kamyar Azizzadenesheli, Jeremy D. Bernstein, Jean Kossaifi, Aran Khanna, Zachary C. Lipton, and Animashree Anandkumar. Stochastic activation pruning for robust adversarial defense. In *International Conference on Learning Representations*, 2018.

[14] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.

[15] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2018.

[16] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[17] Akhilesh Gotmare, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. A closer look at deep learning heuristics: Learning rate restarts, warmup and distillation. In *International Conference on Learning Representations*, 2019.

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

[19] Sven Gowal, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv preprint arXiv:2010.03593*, 2020.

[20] Warren He, Bo Li, and Dawn Song. Decision boundary analysis of adversarial examples. In *International Conference on Learning Representations*, 2018.

[21] Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. In *International Conference on Machine Learning*, pages 2712–2721, 2019.

[22] 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*, pages 97–117. Springer, 2017.

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[24] Sungyoon Lee, Woojin Lee, Jinseong Park, and Jaewook Lee. Loss landscape matters: Training certifiably robust models with favorable loss landscape, 2021.

[25] Chen Liu, Mathieu Salzmann, Tao Lin, Ryota Tomioka, and Sabine Süsstrunk. On the loss landscape of adversarial training: Identifying challenges and how to overcome them. *Advances in Neural Information Processing Systems*, 33, 2020.

[26] Chen Liu, Ryota Tomioka, and Volkan Cevher. On certifying non-uniform bounds against adversarial attacks. In *International Conference on Machine Learning*, pages 4072–4081, 2019.

[27] Xingjun Ma, Bo Li, Yisen Wang, Sarah M. Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Michael E. Houle, Dawn Song, and James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. In *International Conference on Learning Representations*, 2018.

[28] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[29] Chengzhi Mao, Ziyuan Zhong, Junfeng Yang, Carl Vondrick, and Baishakhi Ray. Metric learning for adversarial robustness. In *Advances in Neural Information Processing Systems*, pages 478–489, 2019.

[30] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773, 2017.

[31] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 5947–5956, 2017.

[32] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614*, 2014.

[33] Tianyu Pang, Kun Xu, Yinpeng Dong, Chao Du, Ning Chen, and Jun Zhu. Rethinking softmax cross-entropy loss for adversarial robustness. In *International Conference on Learning Representations*, 2020.

[34] Tianyu Pang, Kun Xu, Chao Du, Ning Chen, and Jun Zhu. Improving adversarial robustness via promoting ensemble diversity. In *International Conference on Machine Learning*, pages 4970–4979, 2019.

[35] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. In *International Conference on Learning Representations*, 2018.

[36] Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems*, pages 10877–10887, 2018.

[37] Hadi Salman, Jerry Li, Ilya Razenshteyn, Pengchuan Zhang, Huan Zhang, Sebastien Bubeck, and Greg Yang. Provably robust deep learning via adversarially trained smoothed classifiers. In *Advances in Neural Information Processing Systems*, pages 11292–11303, 2019.

[38] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*, 2018.

[39] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. Fast and effective robustness certification. In *Advances in Neural Information Processing Systems*, pages 10825–10836, 2018.

[40] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL):41, 2019.

[41] Abhishek Sinha, Mayank Singh, Nupur Kumari, Balaji Krishnamurthy, Harshitha Machiraju, and Vineeth N Balasubramanian. Harnessing the vulnerability of latent layers in adversarially trained models. *arXiv preprint arXiv:1905.05186*, 2019.

[42] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

[43] Vincent Tjeng, Kai Y. Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *International Conference on Learning Representations*, 2019.

[44] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. *Advances in Neural Information Processing Systems*, 33, 2020.

[45] Shiqi Wang, Yizheng Chen, Ahmed Abdou, and Suman Jana. Mixtrain: Scalable training of formally robust neural networks. *arXiv preprint arXiv:1811.02625*, 2018.

[46] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Efficient formal safety analysis of neural networks. In *Advances in Neural Information Processing Systems*, pages 6367–6377, 2018.

[47] Lily Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning, and Inderjit Dhillon. Towards fast computation of certified robustness for relu networks. In *International Conference on Machine Learning*, pages 5276–5285, 2018.

[48] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5286–5295. PMLR, 2018.

[49] Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J Zico Kolter. Scaling provable adversarial defenses. In *Advances in Neural Information Processing Systems*, pages 8410–8419, 2018.

[50] Dongxian Wu, Shu-Tao Xia, and Yisen Wang. Adversarial weight perturbation helps robust generalization. *Advances in Neural Information Processing Systems*, 33, 2020.

[51] Chang Xiao, Peilin Zhong, and Changxi Zheng. Enhancing adversarial defense by k-winners-take-all. In *International Conference on Learning Representations*, 2020.

[52] Kai Y. Xiao, Vincent Tjeng, Nur Muhammad (Mahi) Shafiullah, and Aleksander Madry. Training for faster adversarial robustness verification via inducing reLU stability. In *International Conference on Learning Representations*, 2019.

[53] Dinghuai Zhang, Tianyuan Zhang, Yiping Lu, Zhanxing Zhu, and Bin Dong. You only propagate once: Accelerating adversarial training via maximal principle. In *Advances in Neural Information Processing Systems*, pages 227–238, 2019.

[54] Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. In *International Conference on Learning Representations*, 2020.

[55] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In *Advances in Neural Information Processing Systems*, pages 4944–4953, 2018.