Training Provably Robust Models by Polyhedral Envelope Regularization

Chen Liu, Mathieu Salzmann, Sabine Süsstrunk *

February 15, 2020

Abstract

Training certifiable neural networks enables one to obtain models with robustness guarantees against adversarial attacks. In this work, we introduce a framework to bound the adversary-free region in the neighborhood of the input data by a polyhedral envelope, which yields finer-grained certified robustness. We further introduce polyhedral envelope regularization (PER) to encourage larger polyhedral envelopes 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 general activation functions. Compared with the state-of-the-art methods, PER has very little computational overhead and better robustness guarantees without over-regularizing the model.

1 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 [8, 19, 28]. The existence of such adversarial examples indicates unsatisfactory properties of the deep learning models' decision boundary [12], 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 method (FGM) [8], the CW attack [3] and the projected gradient descent (PGD) algorithm [18], typically exploit the gradient of the loss w.r.t. the input to generate adversarial examples. To counteract such attacks, many defense algorithms have been proposed [2, 6, 11, 17, 24, 27, 35]. However, it was shown by [1] 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 only exception is adversarial training [18], which augments the training data with adversarial examples. Nevertheless, while adversarial training yields good empirical performance, 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. In this context, complete certifiers can either guarantee the absence of an adversary or construct an adversarial example given an adversarial budget. They are typically built on either Satisfiability Modulo Theories (SMT) [13] or mixed integer programming (MIP) [29, 34]. 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

*School of Computer and Communication Sciences, EPFL

context, techniques such as linear approximation [15, 33, 37, 32], symbolic interval analysis [31], abstract transformers [7, 25, 26], semidefinite programming (SDP) [20, 21] and randomized smoothing [4, 22] have been exploited to offer better certified robustness. Some of these methods enable training provably robust models [20, 15, 33, 4, 22] by optimizing the model parameters so as to maximize the area of the certified regions.

While effective, all the above-mentioned certification methods, in their vanilla version, only provide binary results given a *fixed* adversarial budget. 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 adversarial budget that can be certified, [15, 32, 37] use either Newton's method or binary search. By contrast, [5] take advantage of the geometric property of ReLU networks and gives finer-grained robustness guarantees. Given the piecewise linear ReLU function, any input is located in a polytope where the ReLU network can be considered linear. 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, [5] introduce a regularization scheme based on these bounds to effectively train provably robust models.

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 ReLU networks, we estimate a linear bound of 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. Similarly to [5], the intersection of the polyhedral envelope and the adversarial budget is guaranteed to be adversary-free. However, in contrast to [5], our method can be based on any model linearization method and thus applicable to general network architecture and activation functions. We then further introduce a hinge-loss-like regularization term based on our certified bound to train provably robust neural network models.

The computational overhead of our method is negligible and thus it has the same complexity as the model linearization method used. Compared with [5], it is more generally applicable and yields a more accurate estimation of the decision boundary. Compared with [15, 33], which have been found to over-regularize the model [36], our method achieves better certified robustness without sacrificing the clean accuracy too much. In the remainder of the paper, we call our certification method *Polyhedral Envelope Certifier (PEC)* and our regularization scheme *Polyhedral Envelope Regularizer (PER)*.

2 Preliminaries

2.1 Notation and Terminology

For simplicity, we discuss our approach using a standard N-layer fully-connected network. Note, however, that it straightforwardly extends to any network topology that can be represented by a directed acyclic graph (DAG) in the same way as in [16]. A fully-connected network parameterized by $\{\mathbf{W}^{(i)}, \mathbf{b}^{(i)}\}_{i=1}^{N-1}$ can be expressed as

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

where $\mathbf{z}^{(i)}$ and $\hat{\mathbf{z}}^{(i)}$ are the pre- & post-activations of the i-th layer, respectively, and $\hat{\mathbf{z}}^{(1)} \stackrel{\text{def}}{=\!=} \mathbf{x}$ 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 $\hat{\mathbf{z}}'^{(i)}$ 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 $\mathbf{b}^{(i)} \in \mathbb{R}^{n_{i+1}}$.

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

2.2 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

$$\mathbf{U}^{(N)}\mathbf{x}' + \mathbf{p}^{(N)} \leq \mathbf{z}'^{(N)} \leq \mathbf{V}^{(N)}\mathbf{x}' + \mathbf{q}^{(N)} \ . \tag{2}$$

The linear coefficients introduced above can be calculated by iteratively estimating the bounds of intermediate layers and linearizing the activation functions. In Appendix A.1, we discuss this for several activation functions, including ReLU, sigmoid, tanh and arctan. Note that our method differs from [37] because we need the analytical form of the linear coefficients for training, which removes numerical methods such as binary search. The bounding algorithm trades off computational complexity and bound tightness. In this work, we study two such algorithms. One is based on Fast-Lin / CROWN [32, 37]. It yields tighter bounds but has high computational complexity. Another is inspired by the interval bound propagation (IBP) [10], which instead is faster but leads to looser bounds. The details of both algorithms are provided in Appendices A.2 and A.3. We briefly discuss the complexity of both algorithms in Section 5.

3 Algorithms

3.1 Robustness Guarantees by Polyhedral Envelope

For an input point $\mathbf{x}$ with label $c \in [K]$, a sufficient condition to guarantee robustness is that the lower bounds of $\mathbf{z}_c'^{(N)} - \mathbf{z}_i'^{(N)}$ are positive for all $i \in [K]$. Here, we use the *elision of the last layer* introduced in [10] to merge the subtraction of $\mathbf{z}_c'^{(N)}$ and $\mathbf{z}_i'^{(N)}$ with the last linear layer and obtain the lower bound of $\mathbf{z}_c'^{(N)} - \mathbf{z}_i'^{(N)}$: $\underline{\mathbf{z}_c'^{(N)} - \mathbf{z}_i'^{(N)}} \overset{\text{def}}{=\!=} \mathbf{U}_i\mathbf{x}' + \mathbf{p}_i$. Then, the sufficient condition to ensure robustness within a budget $\mathcal{S}_\epsilon^{(p)}(\mathbf{x})$ can be written as

$$\underline{\mathbf{z}_c'^{(N)} - \mathbf{z}_i'^{(N)}} = \mathbf{U}_i\mathbf{x}' + \mathbf{p}_i \geq 0 \quad \forall i \in [K] \ . \tag{3}$$

The constraint is trivial when $i = c$, 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_{ic} to represent the distance between the input and the hyperplane defined in (3) and define $d_c = \min_{i \in [K], i \neq c} d_{ic}$ as the distance between the input and the polyhedral envelope boundary. The distance can be based on different l_p norms, and $d_{ic} = 0$ when the input itself does not satisfy the inequality (3). Since (3) is a sufficient condition for robustness given the adversarial budget $\mathcal{S}_\epsilon^{(p)}(\mathbf{x})$, there is no adversarial examples in the intersection of $\mathcal{S}_\epsilon^{(p)}(\mathbf{x})$ and the polytope defined in (3).

The theorem below formalizes our robustness certification. We defer its proof to Appendix C.1.

Theorem 1 (PEC in Unconstrained Cases)**.** *Given a model $f : \mathbb{R}^{n_1} \to [K]$ and an input point $\mathbf{x}$ with label c, let $\mathbf{U}$ and $\mathbf{p}$ in (3) 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 $d = \min\{\epsilon, d_c\}$ where $d_{ic} = \max\left\{0, \frac{\mathbf{U}_i\mathbf{x}+\mathbf{p}_i}{\|\mathbf{U}_i\|_q}\right\}$. l_q is the dual norm of the l_p norm, i.e., $\frac{1}{p} + \frac{1}{q} = 1$.*

Based on Theorem 1, when $\epsilon < d_c$, PEC has the same robustness guarantees as KW [15], Fast-Lin [32] and CROWN [37] if we use the same model linearization method. When $0 < d_c < \epsilon$,

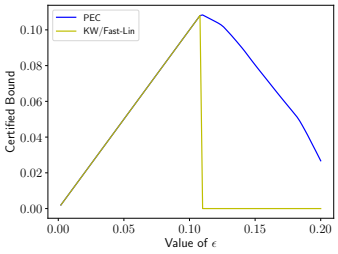


Figure 1: Certified l_∞-based bound of a randomly picked input by PEC and KW / Fast-Lin for different values of ϵ. The model is the 'FC1' on MNIST trained by 'MMR+at' in [5]

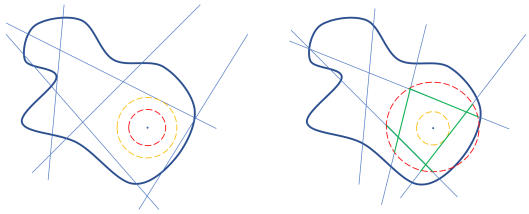


Figure 2: 2D sketch of decision boundary (dark blue), hyperplane defined by (3) (light blue), adversarial budget (red), polyhedral envelope (green) in PEC. The distance between the input data and the hyperplanes is depicted by a yellow circle. The left and right half correspond to the cases when d_c is bigger and smaller than ϵ, respectively.

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 and PEC on a randomly picked input for different values of ϵ.

Figure 2 shows a 2D sketch of the two cases mentioned above. If ϵ is too small, as in the left half of the figure, the linear bounds in (3) are tight but only valid in a small region $\mathcal{S}_\epsilon^{(p)}(\mathbf{x})$. Therefore, the certified robustness is ϵ at most. If ϵ is too large, the linear bounds are valid in a larger region but more pessimistic. This is because the value of d_{ic} monotonically decreases with the increase of ϵ for all model linearization methods. This is depicted by the right half of the figure, where the distances between the input and the hyperplanes are smaller and the certified robustness is then d_c. The hyperplane segments inside the adversarial budget (green lines) never exceed the decision boundary (dark blue lines), by definition of the polyhedral envelope.

To search for the optimal ϵ, i.e., the peak in Figure 1, [15] use Newton's method, which is an expensive second-order method. By contrast, [32, 37] use binary search, and our PEC can accelerate their strategy. The acceleration comes from the partial credit when $0 < d_c < \epsilon$, in which case we can rule out all values of ϵ below d_c and obtain a better lower bound of the optimal ϵ. A more detailed discussion is provided in Appendix B.1.

In many applications, the input is constrained in a hypercube $[r^{(min)}, r^{(max)}]^{n_1}$. For example, for images with normalized pixel values, an attacker will not perturb the image out of the hypercube $[0, 1]^{n_1}$. Because of this constraint for the attacker, the certified regions become larger.

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 (3) within the hypercube. The value of d_{ic} then is the minimum of the optimization problem

$$\begin{aligned} &\min_{\Delta} \|\Delta\|_p \\ s.t.\ \ &\mathbf{a}\Delta + b \leq 0, \quad \Delta^{(min)} \leq \Delta \leq \Delta^{(max)} \end{aligned} \tag{4}$$

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

$$\widehat{\Delta}_i = -\frac{b}{\|\mathbf{a}\|_q^q} \text{sign}(\mathbf{a}_i) \, |\mathbf{a}_i|^{\frac{q}{p}} \ , \tag{5}$$

where $\text{sign}(\cdot)$ 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 (5) 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.

Corollary 1. *If the maximum number of iterations $I^{(max)}$ in Algorithm 1 is large enough to make $\Delta^{(min)} \leq \widehat{\Delta} \leq \Delta^{(max)}$ satisfied in Problem* (4)*, then the output $\|\widehat{\Delta}\|_p$ is the optimum of Problem (4), i.e., d_{ic}.*

We can use the primal-dual method to prove Corollary 1, which we defer to Appendix C.2. Once we have the value of d_{ic} and thus d_c, the certified bound in this constrained case is then $\min\{\epsilon, d_c\}$, as demonstrated by Theorem 1.

Algorithm 1: Greedy algorithm to solve Problem (4).

Input: $\mathbf{x}$, $\mathbf{a}$, b, $\Delta^{(min)}$, $\Delta^{(max)}$ in (4) 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 (5)
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 (5) with elements in $\mathcal{S}^{(f)}$ fixed
  Update $i = i + 1$
end while
Output: $\|\widehat{\Delta}\|_p$

We observed $I^{(max)} = 20$to be sufficient to satisfy the condition of Corollary 1. In practice, the while-loop breaks within 5 iterations in most cases, which means Algorithm 1 introduces little overhead. 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 (4), and thus we eventually get a suboptimal but still valid robustness guarantee.

3.2 Geometry-Inspired Regularization

As in [5], we can incorporate our certification bounds in the training process so as to obtain more 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 (3) and $\tilde{d}_{ic} = d_{ic}$; when $d_{ic} = 0$, the clean input does not satisfy (3) 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 (3). 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 (4), $\tilde{d}_{ic} = \text{sign}(b)\|\widehat{\Delta}\|_p$, 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 3.1.

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

$$\text{PER}(\mathbf{x}, \alpha, \gamma, T) = \gamma \sum_{i=0}^{T-1} \max\left(0, 1 - \frac{\tilde{d}_{j_ic}}{\alpha}\right) . \tag{6}$$

Note that, following [5], 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 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 [9].

We can further incorporate PER with adversarial training in a similar way to [5]. Here, the distance $\tilde{d}_{j_{ic}}$ in (6) is calculated between the polyhedral envelope and the adversarial example generated by PGD [18] instead of the clean input. Note that, the polyhedral envelope is the same in both cases because it depends on the adversarial budget. 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 5. To prevent a prohibitive computational and memory overhead, we use the stochastic robust approximation of [30]. For a mini-batch of size B, we only calculate the PER or PER+at regularization term for $B' < B$ instances sub-sampled from this mini-batch. [19] empirically observed the geometric correlation of high-dimensional decision boundaries near the data manifold. Thus, in practice, a B' much smaller than B provides a good approximation of the full-batch regularization.

4 Experiments

To validate the theory 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 GPU machine within hours. We will make the code and checkpoints publicly available.

4.1 Training and Certifying ReLU Networks

As the main experiment, we first demonstrate the benefits of our approach over the state-of-the-art training and certification methods. To this end, we use the same model architectures as in [5, 15]: **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) [18], KW [15], IBP [10], CROWN-IBP [36], MMR, MMR plus adversarial training (MMR + at) [5]. 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-style model linearization. To evaluate robustness, we report the results of Fast-Lin [32], IBP [10] and our proposed PEC, in addition to the clean test error (CTE) and the empirical robust error against PGD (PGD). Note that KW [15] is algorithmically equivalent to Fast-Lin with *elision of the last layer* [23], so we consider them as one certification method, denoted as Lin. Based on the discussions in Section 3.1, KW, Fast-Lin and PEC have the same certified robust error, which is the proportion of the inputs whose certified regions are smaller than the adversarial budget. Therefore, for these three methods, we report the certified robust error as CRE Lin, and also report the IBP one (CRE IBP). In addition, we calculate the average certified

	CTE (%)	PGD (%)	CRE Lin (%)	CRE IBP (%)	ACB Lin	ACB IBP	ACB PEC	CTE (%)	PGD (%)	CRE Lin (%)	CRE IBP (%)	ACB Lin	ACB IBP	ACB PEC
	MNIST - FC1, ReLU, l_∞, $\epsilon = 0.1$							**MNIST - FC1, ReLU, l_2, $\epsilon = 0.3$**						
plain	1.99	98.37	100.00	100.00	0.0000	0.0000	0.0000	1.99	9.81	40.97	99.30	0.1771	0.0021	0.2300
at	1.42	9.00	97.94	100.00	0.0021	0.0000	0.0099	1.35	2.99	14.85	99.23	0.2555	0.0023	0.2684
KW	2.26	8.59	12.91	69.20	0.0871	0.0308	0.0928	1.23	2.70	4.91	41.55	0.2853	0.1754	0.2892
IBP	1.65	9.67	87.27	15.20	0.0127	0.0848	0.0705	1.36	2.90	6.87	9.01	0.2794	0.2730	0.2876
C-IBP	1.98	9.50	67.39	14.45	0.0326	0.0855	0.0800	1.26	2.80	6.36	8.73	0.2809	0.2738	0.2884
MMR	2.11	17.82	33.75	99.88	0.0663	0.0001	0.0832	2.40	5.88	7.76	99.55	0.2767	0.0013	0.2845
MMR+at	2.04	10.39	17.64	95.09	0.0824	0.0049	0.0905	1.77	3.76	5.68	99.86	0.2830	0.0004	0.2880
C-PER	1.60	7.45	11.71	92.89	0.0883	0.0071	0.0935	1.26	2.44	5.35	59.17	0.2840	0.1225	0.2888
C-PER+at	1.81	7.73	12.90	99.90	0.0871	0.0001	0.0925	0.67	1.40	4.84	64.79	0.2855	0.1056	0.2910
I-PER	1.60	6.28	11.96	93.33	0.0880	0.0067	0.0934	1.21	2.59	5.34	54.13	0.2840	0.1376	0.2888
I-PER+at	1.54	7.15	13.96	98.55	0.0868	0.0014	0.0927	0.74	1.46	7.81	72.85	0.2766	0.0814	0.2860
	MNIST - CNN, ReLU, l_∞, $\epsilon = 0.1$							**MNIST - CNN, ReLU, l_2, $\epsilon = 0.3$**						
plain	1.28	85.75	100.00	100.00	0.0000	0.0000	0.0000	1.28	4.93	100.00	100.00	0.0000	0.0000	0.0000
at	1.02	4.75	91.91	100.00	0.0081	0.0000	0.0189	1.12	2.50	100.00	100.00	0.0000	0.0000	0.0000
KW	1.21	3.03	4.44	100.00	0.0956	0.0000	0.0971	1.11	2.05	5.84	100.00	0.2825	0.0000	0.2861
IBP	1.51	4.43	23.89	8.13	0.0761	0.0919	0.0872	2.37	3.85	51.12	11.73	0.1534	0.2648	0.1669
C-IBP	1.85	4.28	10.72	6.91	0.0893	0.0931	0.0928	2.89	4.44	31.62	12.29	0.2051	0.2631	0.2178
MMR	1.65	6.09	11.56	100.00	0.0884	0.0000	0.0928	2.57	5.49	10.03	100.00	0.2699	0.0000	0.2788
MMR+at	1.19	3.35	9.49	100.00	0.0905	0.0000	0.0939	1.73	3.22	9.46	100.00	0.2716	0.0000	0.2780
C-PER	1.44	3.44	5.13	100.00	0.0949	0.0000	0.0965	1.02	1.87	5.04	100.00	0.2849	0.0000	0.2882
C-PER+at	0.50	2.02	4.85	100.00	0.0952	0.0000	0.0969	0.43	0.91	5.43	100.00	0.2837	0.0000	0.2878
I-PER	1.03	2.40	4.64	99.55	0.0954	0.0004	0.0967	1.11	2.16	6.37	100.00	0.2809	0.0000	0.2851
I-PER+at	0.48	1.29	4.61	99.94	0.0954	0.0001	0.0971	0.52	1.12	7.89	100.00	0.2763	0.0000	0.2812
	CIFAR10 - CNN, ReLU, l_∞, $\epsilon = 2/255$							**CIFAR10 - CNN, ReLU, l_2, $\epsilon = 0.1$**						
plain	24.62	86.29	100.00	100.00	0.0000	0.0000	0.0000	23.29	47.39	100.00	100.00	0.0000	0.0000	0.0000
at	27.04	48.53	85.36	100.00	0.0011	0.0000	0.0015	25.84	35.81	99.96	100.00	0.0000	0.0000	0.0000
KW	39.27	48.16	53.81	99.98	0.0036	0.0000	0.0040	40.24	43.87	48.98	100.00	0.0510	0.0000	0.0533
IBP	46.74	56.38	61.81	67.58	0.0030	0.0025	0.0034	57.90	60.03	64.78	78.13	0.0352	0.0219	0.0366
C-IBP	58.32	63.56	66.28	69.10	0.0026	0.0024	0.0029	71.21	72.51	76.23	80.97	0.0238	0.0190	0.0256
MMR	34.59	57.17	69.28	100.00	0.0024	0.0000	0.0032	40.93	50.57	57.07	100.00	0.0429	0.0000	0.0480
MMR+at	35.36	49.27	59.91	100.00	0.0031	0.0000	0.0037	37.78	43.98	53.33	100.00	0.0467	0.0000	0.0502
C-PER	39.21	50.98	57.45	99.98	0.0033	0.0000	0.0038	34.10	52.54	63.42	100.00	0.0369	0.0000	0.0465
C-PER+at	28.87	43.55	56.59	100.00	0.0034	0.0000	0.0040	25.76	33.47	46.74	100.00	0.0533	0.0000	0.0580
I-PER	29.34	51.54	64.34	99.98	0.0028	0.0000	0.0036	33.94	43.06	56.80	100.00	0.0432	0.0000	0.0484
I-PER+at	26.66	43.35	57.72	100.00	0.0033	0.0000	0.0040	24.85	31.32	47.28	100.00	0.0528	0.0000	0.0572

Table 1: Full results of 11 training schemes and 7 evaluation schemes for ReLU networks. The best and second best results among provably robust training methods (plain and at excluded) are marked in grey and light grey, respectively.

bound obtained by Fast-Lin / KW (ACB Lin), IBP (ACB IBP) and PEC (ACB PEC). Note that the average certified bound here is from *one-shot* Fast-Lin / KW / IBP / PEC , i.e., without searching for the optimal adversarial budget. We do not report the certified bound obtained by MMR [5], because, in practice, it only gives trivial results. As a matter of fact, [5] emphasize their training method and report certification results using only KW and MIP [29].

We use the same adversarial budgets as [5] and thus directly download the KW, MMR and MMR+at models from the checkpoints provided by them.[1] 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.1.

We constrain the attacker to perturb the images within $[0, 1]^{n_1}$, and the full results for both the l_∞ and l_2 cases are summarized in Table 1. Among the training methods, our (C/I)-PER or (C/I)-PER+at achieve the best certified accuracy in most cases, followed by KW, and significantly outperform other baselines. The performance of I-PER and I-PER+at is comparable with the one 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. We observe that IBP is only able to certify IBP-trained models and has worse certification results than methods based on model linearization. Consistently with Section 3.1, 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. Figure 10 of Appendix D.2.2 shows the distribution of the certified bounds on the test data.

When compared with KW, our methods, especially PER+at, have much better clean test

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

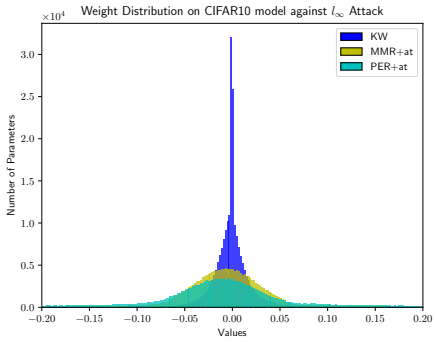


Figure 3: Parameter value distributions of CIFAR10 models trained against l_∞ attacks. The Euclidean norms of KW, MMR+at, PER+at models against l_∞ attack are 18.08, 38.36 and 94.63 respectively, which evidences that the KW model is over-regularized while our PER model best preserves the model capacity.

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_∞ attack. The results of CIFAR10 models against l_2 attack are shown in Figure 8 of Appendix D.2.1. The parameters of C-PER+at models have the largest norms and thus better preserve the model capacity.

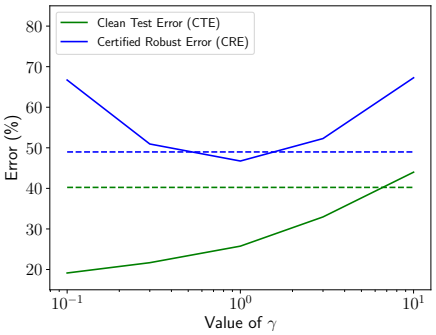


Figure 4: CTE and CRE for different values of γ in C-PER+at to show their trade-off. The results of KW, for reference, are the horizontal dashed lines. The optimal value of γ for C-PER+at is 1.0, with both CTE and CRE better than KW.

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 adversarial examples. However, this trick 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 like KW. If we simply add a loss term on adversarial examples to KW, we will reach a performance between KW and adversarial training (at), which is 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 (6). Figure 4 shows the results of C-PER+at in the l_2 case for CIFAR10. When γ is small, C-PER+at becomes similar to adversarial training (at) and has low clean test error but 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,

	CTE (%)	PGD (%)	CRE CRO (%)	CRE IBP (%)	ACB CRO	ACB IBP	ACB PEC	CTE (%)	PGD (%)	CRE CRO (%)	CRE IBP (%)	ACB CRO	ACB IBP	ACB PEC
	MNIST - FC1, Sigmoid, l_∞, $\epsilon = 0.1$							**MNIST - FC1, Sigmoid, l_2, $\epsilon = 0.3$**						
plain	2.04	97.80	100.00	100.00	0.0000	0.0000	0.0000	2.01	10.25	30.78	94.82	0.2077	0.0155	0.2539
at	1.78	10.05	98.52	100.00	0.0015	0.0000	0.0055	1.65	3.48	7.50	85.84	0.2775	0.0422	0.2839
IBP	2.06	10.58	44.14	13.65	0.0559	0.0863	0.0846	1.40	3.07	6.43	9.13	0.2807	0.2726	0.2873
C-IBP	2.88	9.83	26.04	12.51	0.0740	0.0875	0.0886	1.51	3.24	6.36	8.73	0.2709	0.2738	0.2872
C-PER	1.97	7.55	12.15	84.76	0.0879	0.0152	0.0930	1.36	2.58	6.12	73.71	0.2816	0.0789	0.2867
C-PER+at	2.16	7.12	11.87	88.06	0.0881	0.0119	0.0927	0.46	1.03	5.26	68.94	0.2842	0.0932	0.2905
I-PER	2.15	8.35	12.79	86.99	0.0872	0.0130	0.0926	1.19	2.59	6.05	70.18	0.2818	0.0895	0.2871
I-PER+at	2.45	8.05	12.36	88.94	0.0876	0.0111	0.0923	0.49	1.16	5.03	65.79	0.2849	0.1026	0.2907
	MNIST - FC1, Tanh, l_∞, $\epsilon = 0.1$							**MNIST - FC1, Tanh, l_2, $\epsilon = 0.3$**						
plain	2.00	97.80	100.00	100.00	0.0000	0.0000	0.0000	1.94	16.46	61.66	99.64	0.1150	0.0011	0.1789
at	1.28	8.89	99.98	100.00	0.0000	0.0000	0.0001	1.36	3.02	12.35	97.66	0.2630	0.0070	0.2735
IBP	2.04	9.84	31.81	13.02	0.0682	0.0870	0.0864	1.57	3.17	7.21	10.44	0.2784	0.2688	0.2851
C-IBP	2.75	9.57	20.10	11.80	0.0799	0.0882	0.0894	1.50	3.14	6.64	9.53	0.2801	0.2714	0.2861
C-PER	2.19	7.71	11.55	57.81	0.0885	0.0422	0.0934	1.31	2.47	5.53	55.17	0.2834	0.1345	0.2880
C-PER+at	2.30	7.45	11.39	56.74	0.0886	0.0433	0.0930	0.58	1.30	5.89	54.88	0.2823	0.1354	0.2885
I-PER	2.21	8.51	12.23	55.53	0.0878	0.0445	0.0929	1.38	2.85	5.90	45.31	0.2823	0.1641	0.2874
I-PER+at	2.46	7.87	12.04	66.04	0.0880	0.0340	0.0929	0.55	1.17	5.57	53.73	0.2833	0.1388	0.2890

Table 2: Full results of 8 training schemes and 7 evaluation schemes for sigmoid and tanh networks. The best results among provably robust training methods (plain and at excluded) are marked in grey.

which evidences that C-PER+at is less over-regularized in general than KW, with much lower clean test error for the same certified robust error.

4.2 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 function. MMR and MMR+at are no longer applicable here, because they only support piece-wise linear activation functions. While [33] claim 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[2] does not support non-ReLU activations. For evaluation, we replace Fast-Lin and KW with CROWN [37] and thus report its certified robust error (CRE CRO) and average certified bound (ACB CRO). We use the model linearization method in Appendix A.1 for C-PER and C-PER+at, 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 of [37] is used because it is tighter.

Table 2 leads to similar conclusions to the ReLU networks in Section 4.1. 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.

4.3 The Optimal Adversarial Budget

To obtain the biggest certified bound, we need to search for the optimal value of ϵ, i.e., the peak in Figure 1. While KW [15] uses Newton's method to solve a constrained optimization problem, which is expensive, Fast-Lin and CROWN [32, 37] apply a binary search strategy to find the optimal ϵ. The optimal adversarial budget for a data point is also its optimal certified bound.

To validate the claim in Section 3.1 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 required precision ϵ_Δ. Using $\underline{\epsilon}$ and $\bar{\epsilon}$ to define the initial lower and upper estimates of the optimal value, then we need $\lceil \log_2 \frac{\bar{\epsilon}-\underline{\epsilon}}{\epsilon_\Delta} \rceil$ steps of bound calculation to obtain the optimal value by binary search in Fast-Lin / CROWN. By contrast, the number of iterations needed by PEC depends on the model to certify. The number of iterations by PEC is guaranteed to be smaller than for Fast-Lin / CROWN, because the partial certified bounds obtained by

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

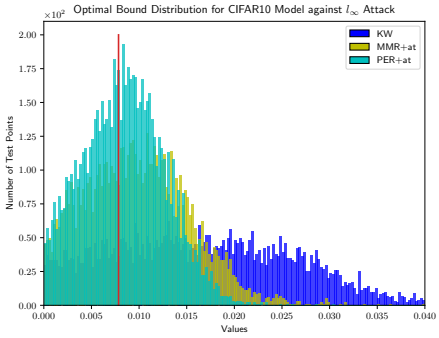


Figure 5: Distribution of optimal certified bounds of CIFAR10 models trained against l_∞ attacks. The target bound (2/255) is indicated by a red vertical line.

PEC indicate tighter lower bounds of the optimal adversarial budget. The pseudo-code of the search algorithm is provided in Appendix B.1.

We briefly discuss the experimental results here, and defer their details to Appendix D.2.3. Note that, because PEC has almost no computational overhead compared with Fast-Lin and CROWN, the number of iterations reflects the running time to obtain the optimal certified bounds. Altogether, our results show that PEC can save on average 25% of the running time for FC1 models with ReLU, 15% for FC1 models with non-ReLU activations and 10% for CNN models.

Figure 5 shows the distribution of 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.2.1. We use vertical red lines to represent the target bounds, so the area on the right of this line is the certified robust accuracy. Compared with KW, the mass of C-PER+at is more concentrated on a 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 (6) 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.

5 Discussion

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 linearization algorithm based on Fast-Lin / CROWN needs $\mathcal{O}(N^2n^3)$ FLOPs to obtain 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 MMR [5], the complexity to calculate the expression of the input's linear region is also $\mathcal{O}(Nn^2m)$. On the training side, KW needs a back-propagation to calculate the bound, which costs $\mathcal{O}(Nn^2)$. MMR needs to calculate the distance between the input and $\mathcal{O}(Nn)$ hyper-planes, costing $\mathcal{O}(Nnm)$ FLOPs, while PER only calculates $\mathcal{O}(k)$ hyperplanes, thus requiring $\mathcal{O}(km)$ FLOPs. Overall, we can see that the estimation of the output logits or decision boundary dominates the complexity of all training algorithms and is the main barrier towards scalable provably robust training.

We can conclude that the complexity of C-PER is $\mathcal{O}(N^2n^3)$, similar to that of Fast-Lin /

CROWN. The overhead to calculate the distances d_{ic} can be ignored here. With IBP-inspired model linearization, the complexity can be reduced to $\mathcal{O}(Nn^2m)$ in I-PER. Note that the FLOP complexity of PGD with h iterations is $\mathcal{O}(Nn^2h)$ and typically $h \ll \min\{m, n\}$, so the overhead of (C/I)-PER+at over (C/I)-PER and the adversarial training warm up phase is also negligible.

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 [16]. 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.

6 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. 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. Unlike many over-regularized methods, it can achieve provable robustness at very little loss in clean accuracy. Extending this framework to larger networks will be the focus of our future research.

7 Acknowledgements

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

References

[1] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

[2] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. 2018.

[3] 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.

[4] Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. Certified adversarial robustness via randomized smoothing. *arXiv preprint arXiv:1902.02918*, 2019.

[5] Francesco Croce, Maksym Andriushchenko, and Matthias Hein. Provable robustness of relu networks via maximization of linear regions. *arXiv preprint arXiv:1810.07481*, 2018.

[6] Guneet S Dhillon, Kamyar Azizzadenesheli, Zachary C Lipton, Jeremy Bernstein, Jean Kossaifi, Aran Khanna, and Anima Anandkumar. Stochastic activation pruning for robust adversarial defense. *arXiv preprint arXiv:1803.01442*, 2018.

[7] 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.

[8] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[9] Akhilesh Gotmare, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. A closer look at deep learning heuristics: Learning rate restarts, warmup and distillation. *arXiv preprint arXiv:1810.13243*, 2018.

[10] 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.

[11] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*, 2017.

[12] Warren He, Bo Li, and Dawn Song. Decision boundary analysis of adversarial examples. 2018.

[13] 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.

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

[15] J Zico Kolter and Eric Wong. Provable defenses against adversarial examples via the convex outer adversarial polytope. *arXiv preprint arXiv:1711.00851*, 1(2):3, 2017.

[16] Chen Liu, Ryota Tomioka, and Volkan Cevher. On certifying non-uniform bound against adversarial attacks. *arXiv preprint arXiv:1903.06603*, 2019.

[17] Xingjun Ma, Bo Li, Yisen Wang, Sarah M Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E Houle, and James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. *arXiv preprint arXiv:1801.02613*, 2018.

[18] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[19] 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.

[20] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018.

[21] 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.

[22] Hadi Salman, Greg Yang, Jerry Li, Pengchuan Zhang, Huan Zhang, Ilya Razenshteyn, and Sebastien Bubeck. Provably robust deep learning via adversarially trained smoothed classifiers. *arXiv preprint arXiv:1906.04584*, 2019.

[23] Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. A convex relaxation barrier to tight robust verification of neural networks. *arXiv preprint arXiv:1902.08722*, 2019.

[24] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *arXiv preprint arXiv:1805.06605*, 2018.

[25] 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.

[26] 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.

[27] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*, 2017.

[28] 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.

[29] Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.

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

[31] 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.

[32] 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. *arXiv preprint arXiv:1804.09699*, 2018.

[33] 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.

[34] Kai Y Xiao, Vincent Tjeng, Nur Muhammad Shafiullah, and Aleksander Madry. Training for faster adversarial robustness verification via inducing relu stability. *arXiv preprint arXiv:1809.03008*, 2018.

[35] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. *arXiv preprint arXiv:1711.01991*, 2017.

[36] Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Duane Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. *arXiv preprint arXiv:1906.06316*, 2019.

[37] 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.

A Model Linearization

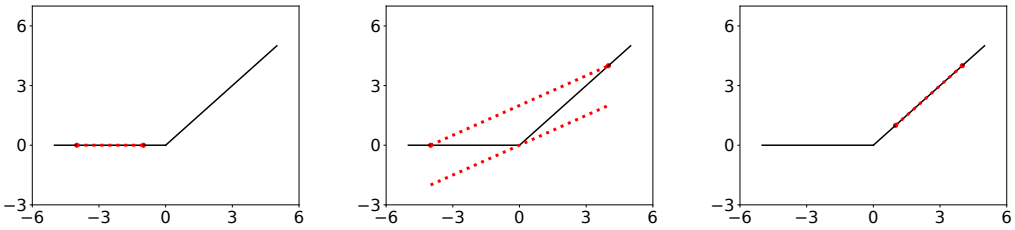
A.1 Linearization of Activation Functions

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 [37]. First, the slope of the linear upper and lower bound is the same for simplicity. Second, all coefficients need to have an analytical form because we need to calculate the gradient based on them during training. Note that [37] use binary search to obtain the optimal d, l, h.

A.1.1 ReLU

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

$$d = \begin{cases} 0 & \underline{x} \leq \bar{x} \leq 0 \\ \dfrac{\bar{x}}{\bar{x} - \underline{x}} & \underline{x} < 0 < \bar{x}, l \\ 1 & 0 \leq \underline{x} \leq \bar{x} \end{cases} = 0, h = \begin{cases} 0 & \underline{x} \leq \bar{x} \leq 0 \\ -\dfrac{\underline{x}\bar{x}}{\bar{x} - \underline{x}} & \underline{x} < 0 < \bar{x} \\ 0 & 0 \leq \underline{x} \leq \bar{x} \end{cases} \tag{7}$$

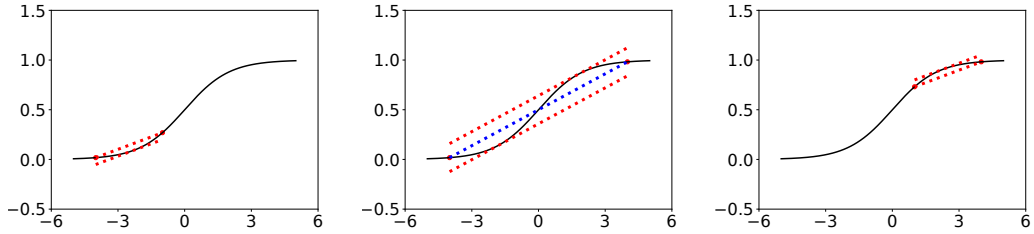


(a) $\underline{x} \leq \bar{x} \leq 0$ (b) $\underline{x} < 0 < \bar{x}$ (c) $0 \leq \underline{x} \leq \bar{x}$

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

A.1.2 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 [37], 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) $0 \leq \underline{x} \leq \bar{x}$

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

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

$$d = \frac{\sigma(\bar{x}) - \sigma(\underline{x})}{\bar{x} - \underline{x}}, l = \begin{cases} \sigma(t_1) - t_1 d & \underline{x} < 0 \\ \dfrac{\bar{x}\sigma(\underline{x}) - \underline{x}\sigma(\bar{x})}{\bar{x} - \underline{x}} & 0 \leq \underline{x} \leq \bar{x} \end{cases}, h = \begin{cases} \dfrac{\bar{x}\sigma(\underline{x}) - \underline{x}\sigma(\bar{x})}{\bar{x} - \underline{x}} & \underline{x} \leq \bar{x} \leq 0 \\ \sigma(t_2) - t_2 d & 0 < \bar{x} \end{cases} \tag{8}$$

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 3.

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 3: Definition of t_1 and t_2 for different activation functions.

A.2 Bounds based on Fast-Lin / CROWN

Based on the linear approximation of activation functions above, 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:

$$\exists \mathbf{D}^{(i)}, \mathbf{l}^{(i)}, \mathbf{u}^{(i)} : \ \forall \mathbf{z}'^{(i)} \in [\underline{\mathbf{z}}^{(i)}, \ \bar{\mathbf{z}}^{(i)}], \ \exists \mathbf{m}^{(i)} \in [\mathbf{l}^{(i)}, \mathbf{u}^{(i)}] \ s.t. \sigma(\mathbf{z}'^{(i)}) = \mathbf{D}^{(i)}\mathbf{z}'^{(i)} + \mathbf{m}^{(i)} \ . \tag{9}$$

We plug (9) into (1), and the expression of $\mathbf{z}'^{(i)}$ can be rewritten as

$$\begin{aligned} \mathbf{z}'^{(i)} &= \mathbf{W}^{(i-1)}(\sigma(\mathbf{W}^{(i-2)}(...\sigma(\mathbf{W}^{(1)}\hat{\mathbf{z}}'^{(1)} + \mathbf{b}^{(1)})...) + \mathbf{b}^{(i-2)})) + \mathbf{b}^{(i-1)} \\ &= \mathbf{W}^{(i-1)}(\mathbf{D}^{(i-1)}(\mathbf{W}^{(i-2)}(...\mathbf{D}^{(2)}(\mathbf{W}^{(1)}\mathbf{x}' + \mathbf{b}^{(1)}) + \mathbf{m}^{(2)}...) + \mathbf{b}^{(i-2)}) + \mathbf{m}^{(i-1)}) + \mathbf{b}^{(i-1)} \\ &= \left(\prod_{k=1}^{i-1} \mathbf{W}^{(k)}\mathbf{D}^{(k)}\right)\mathbf{W}^{(1)}\mathbf{x}' + \sum_{j=1}^{i-1}\left(\prod_{k=j+1}^{i-1} \mathbf{W}^{(k)}\mathbf{D}^{(k)}\right)\mathbf{b}^{(j)} + \sum_{j=2}^{i-1}\left(\prod_{k=j+1}^{i-1} \mathbf{W}^{(k)}\mathbf{D}^{(k)}\right)\mathbf{W}^{(j)}\mathbf{m}^{(j)} \ . \end{aligned} \tag{10}$$

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 bounds of $\mathbf{z}'^{(i)}$ and the bias term in (10). This process can be repeated until we obtain the bound of the output logits (2). The derivation here is the same as in [32, 16], we encourage interested reader to check these works for details.

A.3 Bounds Inspired by Interval Bound Propagation

Interval Bound Propagation (IBP), introduced in [10], 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.2 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 (1).

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 (9) 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.1 to linearize the activation functions, so the slopes of both bounds are the same. Plugging (9) into this formulation, we have

$$\widehat{\mathbf{U}}^{(i)} = \mathbf{D}^{(i)}\mathbf{U}^{(i)},\ \hat{\mathbf{p}}^{(i)} = \mathbf{D}^{(i)}\mathbf{p}^{(i)} + \mathbf{l}^{(i)},\ \hat{\mathbf{q}}^{(i)} = \mathbf{D}^{(i)}\mathbf{q}^{(i)} + \mathbf{u}^{(i)}\ . \tag{11}$$

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

$$\mathbf{U}^{(i+1)} = \mathbf{W}^{(i)}\widehat{\mathbf{U}}^{(i)},\ \mathbf{p}^{(i+1)} = \mathbf{W}_+^{(i)}\hat{\mathbf{p}}^{(i)} + \mathbf{W}_-^{(i)}\hat{\mathbf{q}}^{(i)} + \mathbf{b}^{(i)},\ \mathbf{q}^{(i+1)} = \mathbf{W}_+^{(i)}\hat{\mathbf{q}}^{(i)} + \mathbf{W}_-^{(i)}\hat{\mathbf{p}}^{(i)} + \mathbf{b}^{(i)}\ . \tag{12}$$

By definition, we have $\widehat{\mathbf{U}}^{(1)} = \mathbf{I}$ and $\hat{\mathbf{p}}^{(1)} = \hat{\mathbf{q}}^{(1)} = \mathbf{0}$. Applying (11) and (12) iteratively allows us to obtain the values of the coefficients $\mathbf{U}^{(N)}$, $\mathbf{V}^{(N)}$, $\mathbf{p}^{(N)}$ and $\mathbf{q}^{(N)}$ in (2).

B Algorithms

B.1 Algorithms for Searching the Optimal ϵ

The pseudo code for finding the optimal ϵ is provided as Algorithm 2 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, 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 only 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 bound in one iteration, which leads to a faster convergence of ϵ.

C Proofs

C.1 Proof of Theorem 1

Proof. Let $\mathbf{x}' = \mathbf{x} + \Delta$ be a point that breaks condition (3). Then,

$$\begin{aligned}
& \mathbf{U}_i(\mathbf{x}+\Delta) + \mathbf{p}_i && < 0 \\
\iff\ & \mathbf{U}_i\Delta && < -\mathbf{U}_i\mathbf{x} - \mathbf{p}_i \\
\implies\ & -\|\mathbf{U}_i\|_q\|\Delta\|_p && < -\mathbf{U}_i\mathbf{x} - \mathbf{p}_i \\
\iff\ & \|\Delta\|_p && > \frac{\mathbf{U}_i\mathbf{x} + \mathbf{p}_i}{\|\mathbf{U}_i\|_q}
\end{aligned} \tag{13}$$

Algorithm 2: 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: $\bar{\epsilon} = \epsilon_{try}$
  end if
end while
Output: $\frac{1}{2}(\epsilon_{low} + \epsilon_{up})$

The $\Longrightarrow$ comes from Hölder's inequality. (13) 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 $\min\{\epsilon, d_c\}$. □

C.2 Proof of Corollary 1

Proof. We use the primal-dual method to solve the optimization problem (4), 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 (4) is strictly feasible:

$$\exists \Delta \ s.t. \ \mathbf{a}\Delta + b < 0, \Delta^{(min)} < \Delta < \Delta^{(max)} \ . \tag{14}$$

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

$$\begin{aligned} &\min_{\Delta^{(min)} \leq \Delta \leq \Delta^{(max)}} \|\Delta\|_p^p \\ &s.t. \ \ \mathbf{a}\Delta + b \leq 0 \end{aligned} \tag{15}$$

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 (15) by introducing a coefficient of relaxation $\lambda \in \mathbb{R}_+$:

$$\max_{\lambda \geq 0} \min_{\Delta^{(min)} \leq \Delta \leq \Delta^{(max)}} g(\Delta, \lambda) \stackrel{\text{def}}{=\!=} \|\Delta\|_p^p + \lambda(\mathbf{a}\Delta + b) \tag{16}$$

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:

$$\tilde{\Delta}_i = \text{clip}\left(-\text{sign}(\mathbf{a}_i)\left|\frac{\lambda \mathbf{a}_i}{p}\right|^{\frac{1}{p-1}}, \text{min} = \Delta_i^{(min)}, \text{max} = \Delta_i^{(max)}\right) \ . \tag{17}$$

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 (5) is consistent with $\tilde{\Delta}_i$ in (17) if λ is set properly.[3] 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.

□

D Additional Experiments

D.1 Details of the Experiments

D.1.1 Model Architecture

The FC1 and CNN networks used in this paper are identical to the ones used in [5]. 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 the convolutional layers has 100 neurons.

D.1.2 Hyper-parameter Settings

In all experiments, we use the Adam optimizer [14] 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. 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 final values 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 4.

	α	ϵ	γ
MNIST - FC1, l_∞	constant, 0.15	initial value 0.0064 $\times 2$ every 20 epochs	constant, 0.1
MNIST - CNN, l_∞	constant, 0.15	constant, 0.1	constant, 0.3 for PER constant, 0.03 for PER+at
CIFAR10 - CNN, l_∞	constant, 0.1	constant, 0.008	constant, 3×10^{-4} for PER constant, 1×10^{-3} for PER+at
MNIST - FC1, l_2	constant, 0.45	initial value 0.02 $\times 2$ every 20 epochs	constant, 1.0
MNIST - CNN, l_2	constant, 0.45	constant, 0.3	constant, 1.0
CIFAR10 - CNN, l_2	constant, 0.15	constant, 0.1	constant, 0.3 for PER constant, 1.0 for PER+at

Table 4: Values of α, ϵ and γ for different experiments.

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

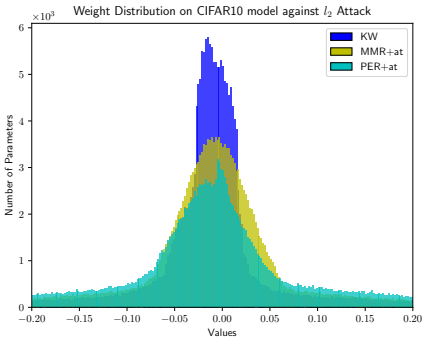


Figure 8: Parameter value distributions of CIFAR10 models trained against l_2 attack. The Euclidean norms of KW, MMR+at, PER+at model against l_2 attacks are 71.34, 62.97 and 141.77, respectively.

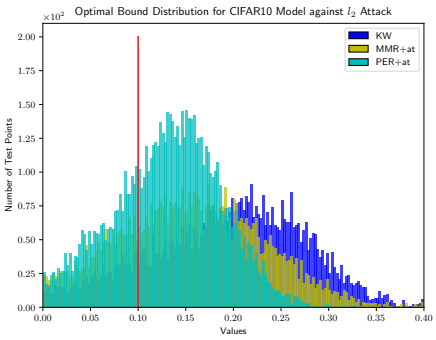


Figure 9: Distribution of optimal certified bounds of CIFAR10 models trained against l_2 attacks. The target bound (0.1) is marked as a red vertical line.

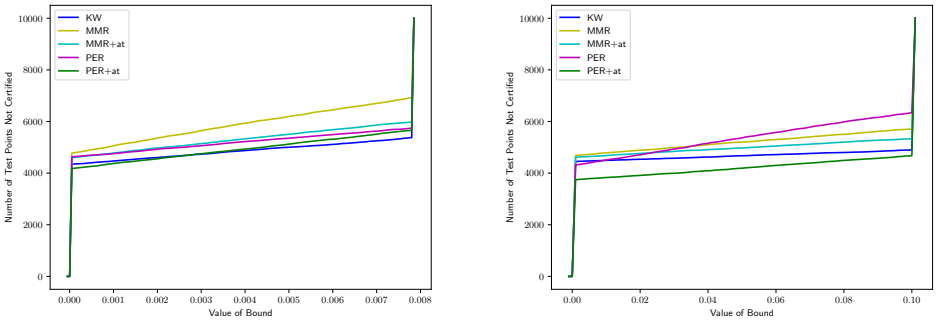


Figure 10: Distribution of certified bounds by one-shot PEC for different models on CIFAR10 in l_∞ (Left) and l_2 (Right) cases. The y-axis shows the number of points in the test set whose certified bounds are smaller than the corresponding value of the x-axis. Lower values on the y-axis mean more robust models.

D.2 Additional Experimental Results

D.2.1 Additional Figures

The parameter value distributions of CIFAR10 models against l_2 attacks are provided in Figure 8. The distribution of optimal certified bounds of CIFAR10 models against l_2 attacks is shown in Figure 9.

D.2.2 Distribution of One-Shot Certified Bounds

The distribution of certified bounds obtained by one-shot PEC on the CIFAR10 models of Table 1 is shown in Figure 10. We plot the number of points whose certified bounds are smaller than a given value, so the jump at $x = 0$ depicts the number of points totally uncertified, while the jump at $x = \epsilon$ depicts the number of fully certified points. The slope between both ends represents the points partially certified, which accounts for $10\% - 20\%$ of all the points. They are the points not certified by KW / Fast-Lin but that have non-trivial bounds by PEC. Among all training methods, our proposed PER+at has competitive performance with KW in the l_∞ case and the best performance in the l_2 case.

D.2.3 Searching for the Optimal Value of ϵ

Table 5 shows the number of bound calculations in the binary search for the optimal ϵ in PEC and Fast-Lin. In l_∞ cases, the original interval $[\underline{\epsilon}, \bar{\epsilon}]$ is $[0, 0.4]$ for MNIST and $[0, 0.1]$ for CIFAR10. In l_2 cases, 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 4.3.

	T_{Lin}	T_{PEC}	$\frac{T_{PEC}}{T_{Lin}}$	T_{Lin}	T_{PEC}	$\frac{T_{PEC}}{T_{Lin}}$	T_{Lin}	T_{PEC}	$\frac{T_{PEC}}{T_{Lin}}$
	MNIST-FC1, ReLU, l_∞			**MNIST-CNN, ReLU, l_∞**			**CIFAR10-CNN, ReLU, l_∞**		
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
	MNIST-FC1, ReLU, l_2			**MNIST-CNN, ReLU, l_2**			**CIFAR10-CNN, ReLU, l_2**		
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 5: Number of steps of bound calculation for the optimal ϵ in Fast-Lin (T_{Lin}) and PEC (T_{PEC}) for ReLU networks. Note that T_{Lin} is a constant for different models given the original interval $[\underline{\epsilon}, \bar{\epsilon}]$.