PRELIMINARY VERSION: DO NOT CITE
The AAAI Digital Library will contain the published
version some time after the conference

Adversarial Training and Provable Robustness: A Tale of Two Objectives

Jiameng Fan , Wenchao Li
Department of Electrical and Computer Engineering
Boston University, Boston
{jmfan, wenchao}@bu.edu

Abstract

We propose a principled framework that combines adversarial training and provable robustness verification for training certifiably robust neural networks. We formulate the training problem as a joint optimization problem with both empirical and provable robustness objectives and develop a novel gradient-descent technique that can eliminate bias in stochastic multi-gradients. We perform both theoretical analysis on the convergence of the proposed technique and experimental comparison with state-of-the-arts. Results on MNIST and CIFAR-10 show that our method can consistently match or outperform prior approaches for provable l_∞ robustness. Notably, we achieve 6.60% verified test error on MNIST at $\epsilon = 0.3$, and 66.57% on CIFAR-10 with $\epsilon = 8/255$.

Introduction

Vulnerability of deep neural networks to adversarial examples (Szegedy et al. 2014; Goodfellow, Shlens, and Szegedy 2015) has spurred the development of training methods for learning more robust models (Wong and Kolter 2018; Gowal et al. 2018; Zhang et al. 2020; Balunovic and Vechev 2020). Madry et al. (2018) show that adversarial training can be formulated as a minimax robust optimization problem as in (1). Given a model f_θ, loss function $\mathcal{L}$, and training data distribution $\mathcal{X}$, the training algorithm aims to minimize the loss whereas the adversary aims to maximize the loss within a neighborhood $\mathbb{S}(\boldsymbol{x}, \epsilon)$ of each input data $\boldsymbol{x}$ as follows:

$$\min_\theta E_{(\boldsymbol{x},y)\in\mathcal{X}} \left[\max_{\boldsymbol{x}'\in\mathbb{S}(\boldsymbol{x},\epsilon)} \mathcal{L}(f_\theta(\boldsymbol{x}'), y) \right] \tag{1}$$

In general, the inner maximization is intractable. Most existing techniques focus on finding an approximate solution. There are two main approaches to approximate the inner loss (henceforth referred to as *robust loss*). One direction is to generate adversarial examples to compute a lower bound of robust loss. The other is to compute an upper bound of robust loss by over-approximating the model outputs. We distinguish these two families of techniques below.

Adversarial training. To improve adversarial robustness, a natural idea is to augment the training set with adversarial examples (Kurakin, Goodfellow, and Bengio 2017). Using adversarial examples to compute the training loss yields a lower bound of *robust loss*, henceforth referred to as *adversarial loss*. Madry et al. (2018) propose to use projected gradient descent (PGD) to compute the adversarial loss and train the neural network by minimizing this loss. Networks trained using this method can achieve state-of-art test accuracy under strong adversaries (Carlini and Wagner 2017; Wang et al. 2018). More recently, Wong, Rice, and Kolter (2020) showed that fast gradient sign method (FGSM) (Goodfellow, Shlens, and Szegedy 2015) with random initialization can be used to learn robust models faster than PGD-based adversarial training. In term of efficiency, FGSM-based adversarial training is comparable to regular training. While adversarial training can produce networks robust against strong attacks, minimizing the adversarial loss alone cannot guarantee that (1) is minimized. In addition, it cannot provide rigorous guarantees on the robustness of the trained networks.

Provable robustness. Verification techniques (Katz et al. 2017; Dvijotham et al. 2018; Ruan, Huang, and Kwiatkowska 2018; Raghunathan, Steinhardt, and Liang 2018; Prabhakar and Afzal 2019), on the other hand, can be used to compute a certified upper bound of *robust loss* (henceforth referred to as *abstract loss*). Given a neural network, a simple way to obtain this upper bound is to propagate value bounds across the network, also known as interval bound propagation (IBP) (Mirman, Gehr, and Vechev 2018; Gowal et al. 2018). Techniques such as CROWN (Zhang et al. 2018), DeepZ (Singh et al. 2018), MIP (Tjeng, Xiao, and Tedrake 2019) and RefineZono (Singh et al. 2019), can compute more precise bounds, but also incur much higher computational costs. Building upon these upper bound verification techniques, approaches such as DIFFAI (Mirman, Gehr, and Vechev 2018) construct a differentiable *abstract loss* corresponding to the upper bound estimation and incorporate this loss function during training. However, Gowal et al. (2018) and Zhang et al. (2020) observe that a tighter approximation of the upper bound does not necessarily lead to a network with low robust loss. They show that IBP-based methods can produce networks with state-of-the-art certified robustness. More recently, COLT (Balunovic and Vechev 2020) proposed to combine adversarial training and zonotope propagation. Zonotopes are a collection of affine forms of the input variables and intermediate vector outputs in the neural network. The idea is to train the network with the so-called latent adversarial examples which are adversarial examples that lie inside these zonotopes.

Table 1: Comparison of different methods for training robust neural networks. We highlight the loss function used in each method. If there is an *abstract loss* used in training or post-training verification, we also list the corresponding verification method. We categorize the methods along five dimensions, with ✓ indicating a desirable property or an explicit consideration.

Method	Loss	*Abstract loss*	Efficiency[1]	Empirical Robustness	Provable Robustness	No weight[2] tuning/scheduling
Baseline	*regular loss*	n/a	✓			n/a
FGSM (2015)	*adversarial loss*	n/a	✓	✓		n/a
FGSM+random init (2020)	*adversarial loss*	n/a	✓	✓		n/a
PGD (2018)	*adversarial loss*	n/a		✓		n/a
COLT (2020)	*latent adversarial loss*	RefineZono[3]		✓	✓	n/a
DIFFAI (2018)	*abstract loss*[4]	DeepZ	✓[5]		✓	n/a
CROWN-IBP (2020)	*regular loss+abstract loss*	CROWN + IBP	✓		✓	
IBP method (2018)	*regular loss+abstract loss*	IBP	✓		✓	
AdvIBP	***adversarial loss+abstract loss***	**IBP**	✓	✓	✓	✓

[1] The efficiency baseline is the training time for each epoch during regular training. ✓ represents the training time is comparable to the baseline.
[2] The weights here represent the weights for the different losses if there are multiple of them.
[3] RefineZono is not used to construct an abstract loss. Instead, it is used to generate latent adversarial examples and for post-training verification.
[4] In their experiments, DIFFAI shows that adding regular loss with a fixed weight can achieve better performance.
[5] DIFFAI can also use IBP for training and verification for improved efficiency. However, the best robustness results are achieved using DeepZ.

This work: a principled framework for combining adversarial loss and abstract loss. We first start with the observation that there is a substantial gap between the provable robustness obtained from state-of-art verification tools and the empirical robustness of the same network against strong adversary in large-scale models. In this paper, we propose to bridge this gap by marrying the strengths of adversarial training and provable bound estimation techniques. Minimizing *adversarial loss* and minimizing *abstract loss* can be viewed as bounding the true *robust loss* from two ends. We argue that simultaneously reducing both losses is more likely to produce a network with good empirical and provable robustness. From an optimization perspective, this amounts to an optimization problem with two objectives and can be solved using gradient descent methods if both objectives are semi-smooth. The challenge is how to balance the minimization of these two objectives during training. In particular, computing the gradient based on a weighted-sum of the objectives can result in biased gradients. Inspired by the work on moment estimates (Kingma and Ba 2016), we propose a novel joint training scheme to compute the weights adaptively and minimize the joint objective with unbiased gradient estimates. For efficient training, we instantiate our framework in a tool called *AdvIBP*, which uses FGSM and random initialization for computing the adversarial loss and IBP for computing the abstract loss. We validate our approach on a set of commonly used benchmarks demonstrate and demonstrate that *AdvIBP* can learn provably robust neural networks that match or outperform state-of-art techniques. We summarize and compare the key features of prior methods and *AdvIBP* in Table 1.

Main contributions. In short, our key contributions are:

- A novel framework for training provably robust deep neural networks. The framework marries the strengths of adversarial training and provable upper bound estimation in a principled way.
- A novel gradient descent method for two-objective optimization that uses moment estimates to address the issue of bias in stochastic multi-gradients. We also perform theoretical analysis of the proposed method.
- Experiments on the MNIST and CIFAR-10 datasets show the proposed method can achieve state-of-the-art performance for networks with provable robustness guarantees.

Background

In this paper, we consider an adversary who can perturb an input $\boldsymbol{x}\in\mathcal{X}$ from a data distribution $\mathcal{X}$ arbitrarily within a small ϵ neighborhood of the input. In the case of l_∞ perturbation, which we experiment with later, we define the allowable adversarial input set as $\mathbb{S}(\boldsymbol{x},\epsilon)=\{\boldsymbol{x}'|\|\boldsymbol{x}'-\boldsymbol{x}\|_\infty\leq\epsilon\}$.

We define a L-layer neural network parameterized by θ as a function f_θ recursively as:

$$f_\theta(\boldsymbol{x})=\boldsymbol{z}^{(L)},\ \ \boldsymbol{z}^{(l)}=\boldsymbol{W}^{(l)}\boldsymbol{h}^{(l-1)}+\boldsymbol{b}^{(l)},\ \ \boldsymbol{h}^{(l)}=\sigma^{(l)}(\boldsymbol{z}^{(l)})$$

where $l\in\{1,\cdots,L-1\}$, $\boldsymbol{z}$ represent the pre-activation neuron values, $\boldsymbol{h}$ represent post-activation neuron values and σ is an element-wise activation function. We denote $h^{(l)}_{\theta_l}$ the mapping applied at layer l with parameter θ_l and the network can be represented as $f_\theta=h^{(L)}_{\theta_L}\circ h^{(L-1)}_{\theta_{L-1}}\cdots\circ h^{(1)}_{\theta_1}$.

In classification, the provable robustness seeks for the lower bounds of the margins between the ground-truth logit and all other classes. Let vector $\boldsymbol{m}$ be the margins between the ground-truth class and all other classes. Each element in $\boldsymbol{m}$ is a linear combination of the output (Wong and Kolter 2018): $\boldsymbol{c}^T f_\theta(\boldsymbol{x})$, where $\boldsymbol{c}$ is set to compute the margin. We define the lower bound of $\boldsymbol{m}$ in $\mathbb{S}(\boldsymbol{x},\epsilon)$ as $\underline{\boldsymbol{m}}(\boldsymbol{x},\epsilon;\theta)$. When all elements of $\underline{\boldsymbol{m}}(\boldsymbol{x},\epsilon;\theta)>0$, $\boldsymbol{x}$ is verifiably robust for any perturbation with l_∞-norm less than ϵ.

Interval bound propagation (IBP). Interval bound propagation uses a simple bound propagation rule. For the input layer we define element-wise upper and lower bound for $\boldsymbol{x}$, $z^{(l)}$ and $\boldsymbol{h}^{(l)}$ as $\boldsymbol{x}_L\leq\boldsymbol{x}\leq\boldsymbol{x}_U$, $\underline{\boldsymbol{z}}^{(l)}\leq\boldsymbol{z}^{(l)}\leq\overline{\boldsymbol{z}}^{(l)}$ and $\underline{\boldsymbol{h}}^{(l)}\leq\boldsymbol{h}^{(l)}\leq\overline{\boldsymbol{h}}^{(l)}$. For affine layers, we have:

$$\overline{\boldsymbol{z}}^{(l)}=\boldsymbol{W}^{(l)-}\cdot\underline{\boldsymbol{h}}^{(l-1)}+\boldsymbol{W}^{(l)+}\cdot\overline{\boldsymbol{h}}^{(l-1)}+\boldsymbol{b}^{(l)},$$
$$\underline{\boldsymbol{z}}^{(l)}=\boldsymbol{W}^{(l)-}\cdot\overline{\boldsymbol{h}}^{(l-1)}+\boldsymbol{W}^{(l)+}\cdot\underline{\boldsymbol{h}}^{(l-1)}+\boldsymbol{b}^{(l)}$$

where $\boldsymbol{W}^{(l)-}=\min(0,\boldsymbol{W}^{(l)})$ and $\boldsymbol{W}^{(l)+}=\max(0,\boldsymbol{W}^{(l)})$. Note that $\overline{\boldsymbol{h}}^{(0)}=\boldsymbol{x}_U$ and $\underline{\boldsymbol{h}}^{(0)}=\boldsymbol{x}_L$. For monotonic increasing activation functions σ, we have $\overline{\boldsymbol{h}}^{(l)}=\sigma(\overline{\boldsymbol{z}}^{(l)})$ and $\underline{\boldsymbol{h}}^{(l)}=\sigma(\underline{\boldsymbol{z}}^{(l)})$.

We define $\underline{\boldsymbol{m}}_{\text{IBP}}(\boldsymbol{x},\epsilon;\theta)$ as the lower bound of the margin obtained by IBP which is an underapproximation of $\underline{\boldsymbol{m}}(\boldsymbol{x},\epsilon;\theta)$. More generally, we use $\underline{\boldsymbol{m}}_{\textit{abstract}}(\boldsymbol{x},\epsilon;\theta)$ as the lower bound of the margin obtained by abstract methods. When $\underline{\boldsymbol{m}}_{\textit{abstract}}(\boldsymbol{x},\epsilon;\theta)\geq 0$, $\boldsymbol{x}$ is verifiably robust by the abstract method for any perturbation with l_∞-norm less than ϵ. Additionally, Wong and Kolter (2018) showed that for cross-entropy (CE) loss:

$$\max_{\boldsymbol{x}'\in\mathbb{S}(\boldsymbol{x},\epsilon)}\mathcal{L}(f_\theta(\boldsymbol{x}'),y)\leq\mathcal{L}(-\underline{\boldsymbol{m}}_{\textit{abstract}}(\boldsymbol{x},\epsilon;\theta),y;\theta) \quad (2)$$

IBP or other abstract methods gives a tractable upper bound of the inner-max in (1) and we refer it as *abstract loss*. In practice, solely minimizing *abstract loss* can be unstable and hard to tune (Mirman, Gehr, and Vechev 2018; Gowal et al. 2018). To mitigate this instability, prior works (Mirman, Gehr, and Vechev 2018; Gowal et al. 2018; Zhang et al. 2020) propose to stabilize the minimization of the abstract loss by adding normal regular loss in the objective. More specifically, the new objective can be formed as follows:

$$\mathcal{L}(\theta)=\kappa_1\mathcal{L}(f_\theta(\boldsymbol{x}),y)+\kappa_2\mathcal{L}(-\underline{\boldsymbol{m}}_{\textit{abstract}};y;\theta) \quad (3)$$

The coefficients κ_1 and κ_2 are hand-tuned to balance the minimization between regular loss and abstract loss. The goal is to improve the robustness of the trained model while avoiding the instability caused by loose abstract loss with respect to the true robust loss. Among different abstract methods, computing IBP bounds only requires two simple forward passes through the network and is thus computationally efficient. The downside of IBP, however, is that it can lead to loose upper bounds. Mirman, Gehr, and Vechev (2018); Gowal et al. (2018) propose to combine regular loss and IBP abstract loss as (3). CROWN-IBP (Zhang et al. 2020) uses a mixture of linear relaxation and IBP to compute the abstract loss and jointly minimize it with the regular loss. While the approaches based on (3) produce state-of-the-art results on a set of benchmarks, this type of works rely on an *ad hoc* scheduler to tune the weights between the regular loss and the abstract loss during training. In addition, regular loss is a loose lower bound of robust loss and minimizing the regular loss does not directly guide the training to a robust model. In this paper, we show that it is better to combine adversarial loss and abstract loss while leveraging the efficiency of IBP. Moreover, we can eliminate weight tuning and scheduling in a principled manner.

Methodology

Overview. Let the perturbed input be $\boldsymbol{x}_{\text{adv}}$. The relations among *adversarial loss*, *robust loss* and *IBP abstract loss* are as follows.

$$\mathcal{L}(f_\theta(\boldsymbol{x}_{\text{adv}}),y)\leq\max_{\boldsymbol{x}'\in\mathbb{S}(\boldsymbol{x},\epsilon)}\mathcal{L}(f_\theta(\boldsymbol{x}'),y)\leq\mathcal{L}(-\underline{\boldsymbol{m}}_{\text{IBP}}(\boldsymbol{x},\epsilon);y;\theta) \quad (4)$$

We note that (4) holds for general adversarial training and provable robustness methods. Specifically adversarial loss provides a lower bound of robust loss and minimizing this loss can result in good empirical robustness. Latent adversarial examples (Balunovic and Vechev 2020), for instance, can be used to construct a different adversarial loss. However, a smaller latent adversarial loss does not necessarily indicate better certified robustness. COLT (Balunovic and Vechev 2020) uses multiple regularizers to mitigate this issue. On the other hand, minimizing the abstract loss can help to train a network with certified robustness. In this case, the choice of verification methods used in computing the abstract loss can significantly influence the final training outcome. For instance, training with the IBP abstract loss can result in a network that is amenable to IBP verification. The true robustness of the network or the robustness attainable under the given neural network architecture, however, could still be far away from this bound. In fact, a small gap between empirical robustness and provable robustness does not necessarily indicate the attainment of good robustness (the extreme case would be a ReLU network with only positive weights). Thus, the tightness of both losses relative to *robust loss* is critical to improving the model's true robustness.

We consider the joint minimization of adversarial loss and abstract loss as a two-objective optimization problem. A straightforward way to solve this joint optimization problem is to optimize a weighted sum of the objectives. This leads to the following objective similar to (3):

$$\mathcal{L}(\theta)=\kappa_1\mathcal{L}(f_\theta(\boldsymbol{x}_{\text{adv}}),y)+\kappa_2\mathcal{L}(-\underline{\boldsymbol{m}}_{\text{IBP}};y;\theta) \quad (5)$$

However, this simple linear-combination formulation is only sensible when the two objectives are not competing, which is rarely the case. The conflicting objectives require modeling the trade-off between objectives, and are generally handled by adaptive weight updates (Sener and Koltun 2018). This approach, however, faces the issue that even though the stochastic gradients for each objective are unbiased estimates of the corresponding full gradients, the weighted sum of the stochastic gradients is a biased estimate if the weights are associated with the sampled gradients. This bias can cause instability and local optima issues (Liu and Vicente 2019). In this paper, we leverage moment estimates to compute the weights adaptively and ensures their independence from the corresponding sampled gradients to eliminate the bias. Minimizing the two objectives jointly tightens the approximation of robust loss from both ends. For efficient training, we develop *AdvIBP* using FGSM+random init to compute *adversarial loss* and IBP to compute *abstract loss*.

Joint Training as Two-Objective Optimization

We propose a two-objective optimization method inspired by (Fan et al. 2019; Zhang, Yu, and Turk 2019) to choose the gradient descent direction that reduces *adversarial loss and abstract loss* simultaneously. Let the adversarial loss be $\mathcal{L}_{\text{adv}}(\theta)$ and IBP abstract loss be $\overline{\mathcal{L}}_{\text{IBP}}(\theta)$. Their gradients with respect to θ are denoted by

$$g_{\text{adv}}=\nabla_\theta\mathcal{L}_{\text{adv}}(\theta),\quad g_{\text{IBP}}=\nabla_\theta\overline{\mathcal{L}}_{\text{IBP}}(\theta)$$

To balance between the two objectives, we update the network parameters in the direction of the angular bisector of the two

gradients. Then, we average the projected vectors of the two gradients on this direction. If $\langle g_{\text{adv}}, g_{\text{IBP}}\rangle > 0$, this results in an update that is expected to reduce both losses to improve the adversarial accuracy and tighten IBP. If $\langle g_{\text{adv}}, g_{\text{IBP}}\rangle \leq 0$, taking the angular bisector direction results in an update that improves the objective functions little or not at all for either objective. In this case, we project one of the gradients onto the hyperplane that is perpendicular to the other gradient. The idea is that when two gradients disagree with each other, we prioritize the minimization of one of the objectives. The final gradient guides the search in the direction that reduces the prioritized objective while avoiding increasing the other objective. We use Figure 1 to illustrate this computation.

To decide which direction to prioritize, the tightness of adversarial loss and abstract loss relative to the ground-truth robust loss can be the determining factor. Wang et al. (2019) propose the First-Order Stationary Condition (FOSC) to quantitatively evaluate the adversarial strength of adversarial examples. In general, the adversarial loss is closer to robust loss with stronger adversarial examples. Let $c(\boldsymbol{x}_{\text{adv}})$ be FOSC value of $\boldsymbol{x}_{\text{adv}}$ and c_t be the threshold that indicates the desired adversarial strength at the t-th epoch. Smaller FOSC values would indicate stronger adversarial examples. With strong attacks ($c(\boldsymbol{x}_{\text{adv}}) \leq c_t$), adversarial training leads to robust models. Thus, we prioritize the gradient of adversarial loss in this case. The idea is to drive the search to the region of robust models with high accuracy and stabilize the minimization of abstract loss. With weak attacks ($c(\boldsymbol{x}_{\text{adv}}) > c_t$), minimizing adversarial loss does not necessarily imply better robustness. However, minimizing abstract loss makes solving (1) tractably. We prioritize the gradient of abstract loss in this case. Figure 1 provides a visualization of the final gradient computation in different cases.

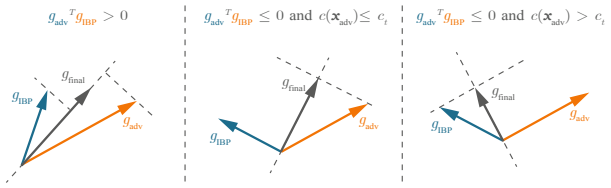


Figure 1: Three cases of computing g_{final} from g_{adv} and g_{adv}.

Stochastic gradients. Since the data distribution $\mathcal{X}$ is unknown in practice, it is impossible to get the full gradients, g_{adv} and g_{IBP}. We denote the realizations of the stochastic objectives at subsequent training epochs $0, \ldots, T-1$ as $\mathcal{L}_{\text{adv},0}(\theta^0), \ldots, \mathcal{L}_{\text{adv},T-1}(\theta^{T-1})$ and $\overline{\mathcal{L}}_{\text{IBP},0}(\theta^0), \ldots, \overline{\mathcal{L}}_{\text{IBP},T-1}(\theta^{T-1})$. The stochastic gradients $g_{\text{adv},t}$ and $g_{\text{IBP},t}$ are the evaluations of data points from mini-batches and provide unbiased estimation of the full gradients. However, the stochastic gradient of the weighted-sum objective at the t-th epoch becomes a biased estimate of the final gradient, g_{final}. The bias is the result of dependence between the weights and the corresponding stochastic gradients.

Unbiased weights computation. To eliminate this bias, we propose to compute the weights from the estimates of the first and norm moments of the gradients instead of the stochastic gradients. The goal is to ensure the independence of stochastic gradients and the corresponding weights. Let $\boldsymbol{m}_{1,t}$, $\boldsymbol{m}_{2,t}$, $\boldsymbol{v}_{1,t}$ and $\boldsymbol{v}_{2,t}$ represent the moment estimates for $g_{\text{adv},t}$, $g_{\text{IBP},t}$, $\|g_{\text{adv},t}\|_2$ and $\|g_{\text{IBP},t}\|_2$ respectively. We modify the moment estimate in (Kingma and Ba 2016) to meet the independence requirement. In Algorithm 1, the t-th moment estimates are the exponential moving averages of the past stochastic gradients from epoch 0 to epoch $t-1$, where the hyper-parameters $\beta_1, \beta_2 \in [0, 1)$ control the exponential decay rates. The moving averages themselves, $\boldsymbol{m}_{1,t}, \boldsymbol{m}_{2,t}, \boldsymbol{v}_{1,t}, \boldsymbol{v}_{2,t}$, are estimate of the first moment and the norm moment of the true gradients. The independent mini-batch sampling guarantees the independence of stochastic gradients. Thus, the moment estimates are independent from the current sampled stochastic gradient. Then, we calculate the weights using the moment estimates in Algorithm 1 and update the model parameters with unbiased gradient estimates.

Algorithm 1 Weight Updates

1: **Input** Exponential decay rates of the moving averages $\beta_1, \beta_2 \in [0, 1)$
2: **Init** $\boldsymbol{m}_{1,0} \leftarrow 0$, $\boldsymbol{m}_{2,0} \leftarrow 0$, $\boldsymbol{v}_{1,0} \leftarrow 0$ and $\boldsymbol{v}_{2,0} \leftarrow 0$
3: **procedure** COMPUTE_WEIGHTS($\boldsymbol{x}_{\text{adv}}, t, c_t$)
4: $\quad \hat{\boldsymbol{m}}_{1,t} \leftarrow \beta_1 \cdot \hat{\boldsymbol{m}}_{1,t-1} + (1-\beta_1) \cdot g_{\text{adv},t-1}$
5: $\quad \hat{\boldsymbol{m}}_{2,t} \leftarrow \beta_1 \cdot \hat{\boldsymbol{m}}_{2,t-1} + (1-\beta_1) \cdot g_{\text{IBP},t-1}$
6: $\quad \hat{\boldsymbol{v}}_{1,t} \leftarrow \beta_2 \cdot \hat{\boldsymbol{v}}_{1,t-1} + (1-\beta_2) \cdot \|g_{\text{adv},t-1}\|_2$
7: $\quad \hat{\boldsymbol{v}}_{2,t} \leftarrow \beta_2 \cdot \hat{\boldsymbol{v}}_{2,t-1} + (1-\beta_2) \cdot \|g_{\text{IBP},t-1}\|_2$
8: $\quad \boldsymbol{m}_{1,t} = \hat{\boldsymbol{m}}_{1,t} / (1-\beta_1^t)$ ▷ Bias-corrected 1st moment
9: $\quad \boldsymbol{m}_{2,t} = \hat{\boldsymbol{m}}_{2,t} / (1-\beta_1^t)$
10: $\quad \boldsymbol{v}_{1,t} = \hat{\boldsymbol{v}}_{1,t} / (1-\beta_2^t)$ ▷ Bias-corrected norm moment
11: $\quad \boldsymbol{v}_{2,t} = \hat{\boldsymbol{v}}_{2,t} / (1-\beta_2^t)$
12: $\quad$ **if** $\langle \boldsymbol{m}_{1,t}, \boldsymbol{m}_{2,t} \rangle > 0$ **then**
13: $\quad\quad \gamma = \frac{1}{2} \langle \boldsymbol{m}_{1,t} + \boldsymbol{m}_{2,t}, \frac{\boldsymbol{m}_{1,t}}{\boldsymbol{v}_{1,t}} + \frac{\boldsymbol{m}_{2,t}}{\boldsymbol{v}_{2,t}} \rangle / \| \frac{\boldsymbol{m}_{1,t}}{\boldsymbol{v}_{1,t}} + \frac{\boldsymbol{m}_{2,t}}{\boldsymbol{v}_{2,t}} \|_2^2$
14: $\quad\quad \kappa_{\text{adv}} = \frac{\gamma}{\boldsymbol{v}_{1,t}}, \kappa_{\text{IBP}} = \frac{\gamma}{\boldsymbol{v}_{2,t}}, \kappa_{\text{reg}} = 0$
15: $\quad$ **else**
16: $\quad\quad$ **if** $c(\boldsymbol{x}_{\text{adv}}) \leq c_t$ **then** ▷ check FOSC value
17: $\quad\quad\quad \kappa_{\text{adv}} = 1, \kappa_{\text{IBP}} = -\frac{\langle \boldsymbol{m}_{1,t} \cdot \boldsymbol{m}_{2,t} \rangle}{\boldsymbol{v}_{2,t}^2}, \kappa_{\text{reg}} = 0$
18: $\quad\quad$ **else**
19: $\quad\quad\quad \kappa_{\text{adv}} = -\frac{\langle \boldsymbol{m}_{1,t} \cdot \boldsymbol{m}_{2,t} \rangle}{\boldsymbol{v}_{1,t}^2}, \kappa_{\text{IBP}} = 1, \kappa_{\text{reg}} = 1/2$
20: $\quad\quad$ **end if**
21: $\quad$ **end if**
22: $\quad$ **return** $\kappa_{\text{adv}}, \kappa_{\text{IBP}}, \kappa_{\text{reg}}$
23: **end procedure**

The overall joint training algorithm is shown in Algorithm 2. The regularization term κ_{reg} in line 11 is only used when prioritizing the minimization of abstract loss. The regularizer helps to bound the convergence rate of training.

Leveraging FOSC in joint training. In Algorithm 2, we use similar dynamic criterion FOSC as in (Wang et al. 2019). In the early stages of training, c_t is close to the maximum FOSC value c_{max}, which can be satisfied with weak adversarial examples. Thus, the early stages of training will mostly prioritize the minimization of adversarial loss. This helps to avoid the instability caused by a loose abstract loss. However, prioritizing the adversarial loss does not necessarily improve

Algorithm 2 Joint Training

Input Warm-up epochs T_{nat} and T_{adv}, ϵ_{train} ramp-up epochs R, maximum FOSC value $c_{\max}$
$f_{\theta^0} \leftarrow$ WARM-UP$(f_{\theta^0}, T_{\text{nat}}, T_{\text{adv}})$
for $t = 0$ to $T-1$ **do**
  c_t=clip$(c_{\max}-(t-R)\cdot c_{\max}/T', 0, c_{\max})$
  Sample $\mathcal{B}=\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_{\mathcal{B}}, y_{\mathcal{B}})\} \sim (\mathcal{X}, \mathcal{Y})$
  for $i = 0$ to $|\mathcal{B}|-1$ **do**
    $\epsilon_t \leftarrow$RAMPUP_SCHEDULER$(t, \epsilon_{\text{train}}, R)$
    $\boldsymbol{x}_{\text{adv},i} \leftarrow$FGSM+RANDOM_INIT$(\boldsymbol{x}_i, y_i, \epsilon_t)$
  end for
  $\kappa_{\text{adv}}, \kappa_{\text{IBP}}, \kappa_{\text{reg}}$=COMPUTE_WEIGHTS$(\boldsymbol{x}_{\text{adv}}, t, c_t)$
  $loss=\kappa_{\text{adv}}\mathcal{L}_{\text{adv}}(\theta^t)+\kappa_{\text{IBP}}\overline{\mathcal{L}}_{\text{IBP}}(\theta^t)+\kappa_{\text{reg}}\|\overline{\mathcal{L}}_{\text{IBP}}(\theta^t)\|_2^2$
  $\theta^{t+1}=\theta^t-\eta_t g_{\text{final}}(\theta^t)$ ▷ $g_{\text{final}}(\theta^t)$: stochastic gradient
end for

procedure WARMUP$(f_{\theta^0}, T_{\text{nat}}, T_{\text{adv}})$ ▷ Warm-up phase
  for $t = 0$ to $T_{\text{nat}}-1$ **do**
    Train on the *regular loss* $\mathcal{L}(f_{\theta^t}(\boldsymbol{x}), y)$
  end for
  for $t = T_{\text{nat}}$ to $T_{\text{nat}}+T_{\text{adv}}-1$ **do**
    Train on the *adversarial loss* $\mathcal{L}(f_{\theta^t}(\boldsymbol{x}_{\text{adv}}), y)$
  end for
  return f_θ
end procedure

the verified robustness of the models. Thus, we design the FOSC value c_t so that it decreases linearly towards zero as training progresses. As a result, in the later training stages, the joint training scheme will mostly prioritize the minimization of the abstract loss to improve provable robustness.

Theoretical Analysis

We provide a theoretical analysis of our proposed joint training scheme to train IBP certified robust networks. It aims to provide insights on how the ground-truth robust loss changes during training by our joint training scheme. The gradient update and the prioritization scheme provide an approximate maximizer for the inner maximization. Below, we provide theoretical analyses on how *robust loss* changes when two gradients agree with each other and how *abstract loss* changes when two gradients disagree with each other.

In detail, let $\boldsymbol{x}^*(\theta)=\arg\max_{\boldsymbol{x}'\in\mathbb{S}(\boldsymbol{x},\epsilon)}\mathcal{L}(f_\theta(\boldsymbol{x}'), y)$. $\hat{\boldsymbol{x}}(\theta)$ is a δ-approximation solution to $\boldsymbol{x}^*$, if it satisfies that (Wang et al. 2019)

$$c(\hat{\boldsymbol{x}}(\theta))=\max_{\boldsymbol{x}'\in\mathbb{S}(\boldsymbol{x},\epsilon)}\langle \boldsymbol{x}'-\hat{\boldsymbol{x}}(\theta), \nabla_{\boldsymbol{x}'}\mathcal{L}(f_\theta(\hat{\boldsymbol{x}}(\theta)), y)\rangle \le \delta \quad (6)$$

Let the robust loss in (1) be $\mathcal{L}(\theta)$, and its gradient be $\nabla\mathcal{L}(\theta)=\mathbb{E}[\nabla_\theta\mathcal{L}(f_\theta(\boldsymbol{x}^*(\theta)), y)]$. We denote the stochastic gradient of $\mathcal{L}(\theta)$ as $g(\theta) = 1/|\mathcal{B}|\sum_{i\in\mathcal{B}}\nabla_\theta\mathcal{L}(f_\theta(\boldsymbol{x}_i^*(\theta)), y_i)$, where $\mathcal{B}$ is the mini-batch. Similarly, we denote the abstract loss as $\overline{\mathcal{L}}(\theta)$, and its gradient as $\nabla\overline{\mathcal{L}}(\theta) = \mathbb{E}[\nabla_\theta\mathcal{L}(-\underline{\boldsymbol{m}}(\boldsymbol{x},\epsilon); y; \theta)]$. We denote the stochastic gradient of $\overline{\mathcal{L}}(\theta)$ as $\overline{g}(\theta)=1/|\mathcal{B}|\sum_{i\in\mathcal{B}}\nabla_\theta\mathcal{L}(-\underline{\boldsymbol{m}}(\boldsymbol{x}_i,\epsilon); y_i; \theta)$. Note that $\mathbb{E}[g(\theta)]=\nabla\mathcal{L}(\theta)$ and $\mathbb{E}[\overline{g}(\theta)]=\nabla\overline{\mathcal{L}}(\theta)$. The adversarial loss, $\mathcal{L}_{\text{adv}}(\theta)$, is $\mathbb{E}[\mathcal{L}(f_\theta(\hat{\boldsymbol{x}}(\theta)), y)]$ and its stochastic gradient is $\hat{g}(\theta)=1/|\mathcal{B}|\sum_{i\in\mathcal{B}}\nabla_\theta\mathcal{L}(f_\theta(\hat{x}_i(\theta)), y_i)$. We make assumptions similar to those in Wang et al. (2019) and present the theoretical analysis of our method below.

Assumption 1. *The function $\mathcal{L}(\theta;\boldsymbol{x})$ and $\overline{\mathcal{L}}(\theta;\boldsymbol{x})$ satisfies the gradient Lipschitz conditions s.t.*

$$\sup_{\boldsymbol{x}}\|\nabla_\theta\mathcal{L}(\theta;\boldsymbol{x})-\nabla_\theta\mathcal{L}(\theta';\boldsymbol{x})\|_2\le L_{\theta\theta}\|\theta-\theta'\|_2$$
$$\sup_{\boldsymbol{x}}\|\nabla_\theta\overline{\mathcal{L}}(\theta;\boldsymbol{x})-\nabla_\theta\overline{\mathcal{L}}(\theta';\boldsymbol{x})\|_2\le \overline{L}_{\theta\theta}\|\theta-\theta'\|_2$$
$$\sup_{\theta}\|\nabla_\theta\mathcal{L}(\theta;\boldsymbol{x})-\nabla_\theta\mathcal{L}(\theta;\boldsymbol{x}')\|_2\le L_{\theta\boldsymbol{x}}\|\boldsymbol{x}-\boldsymbol{x}'\|_2$$
$$\sup_{\theta}\|\nabla_\theta\overline{\mathcal{L}}(\theta;\boldsymbol{x})-\nabla_\theta\overline{\mathcal{L}}(\theta;\boldsymbol{x}')\|_2\le \overline{L}_{\theta\boldsymbol{x}}\|\boldsymbol{x}-\boldsymbol{x}'\|_2$$
$$\sup_{\boldsymbol{x}}\|\nabla_{\boldsymbol{x}}\mathcal{L}(\theta;\boldsymbol{x})-\nabla_{\boldsymbol{x}}\mathcal{L}(\theta';\boldsymbol{x})\|_2\le L_{\boldsymbol{x}\theta}\|\theta-\theta'\|_2$$
$$\sup_{\boldsymbol{x}}\|\nabla_{\boldsymbol{x}}\overline{\mathcal{L}}(\theta;\boldsymbol{x})-\nabla_{\boldsymbol{x}}\overline{\mathcal{L}}(\theta';\boldsymbol{x})\|_2\le \overline{L}_{\boldsymbol{x}\theta}\|\theta-\theta'\|_2$$

where $L_{\theta\theta}$, $L_{\theta\boldsymbol{x}}$, $L_{\boldsymbol{x},\theta}$, $\overline{L}_{\theta\theta}$, $\overline{L}_{\theta\boldsymbol{x}}$, $\overline{L}_{\boldsymbol{x},\theta}$ are positive scalars.

Assumption 1 was made in (Wang et al. 2019) to assume the smoothness of the loss function. Recent studies (Du et al. 2019a,b) help justify it by showing that the loss function of overparameterized neural networks is semi-smooth.

Let $L=(L_{\theta\boldsymbol{x}}L_{\boldsymbol{x}\theta}/\mu+L_{\theta\theta})$ and $\overline{L}=(\overline{L}_{\theta\boldsymbol{x}}\overline{L}_{\boldsymbol{x}\theta}/\mu+\overline{L}_{\theta\theta})$. For stochastic gradient descent, we can assume that the variances of stochastic gradients $g(\theta)$ and $\overline{g}(\theta)$ are bounded by constants $\sigma, \overline{\sigma} > 0$. Let $\Delta=\mathcal{L}(\theta^0)-\min_\theta\mathcal{L}(\theta)$ and $\overline{\Delta}=\overline{\mathcal{L}}(\theta^0)-\min_\theta\overline{\mathcal{L}}(\theta)$. Under Assumption 1 , we have the following theoretical results.

Theorem 1. *If the dot product of the gradients of the two objectives is greater than 0 and the step size of the training is set to $\eta_t=\eta=\min(1/6L, \sqrt{\Delta/TL\sigma^2})$, then the expectation of the gradient of robust loss satisfies*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla\mathcal{L}(\theta^t)\|_2^2] \le 8\sigma\sqrt{\frac{L\Delta}{T}}+\frac{7L_{\theta\boldsymbol{x}}^2\delta}{3\mu}.$$

Theorem 2. *If the dot product of the gradients of the two objectives is smaller or equal to 0, adversarial loss is not tight enough ($c(\boldsymbol{x}_{adv})>c_t$), and the step size of training is set to $\eta_t=\eta=\min(2*\mathbb{E}[\overline{\mathcal{L}}_{IBP}(\theta^t)]/\overline{L}-1/\overline{L}, \sqrt{\overline{\Delta}/T\overline{L}\sigma^2})$ with $\mathbb{E}[\overline{\mathcal{L}}_{IBP}(\theta^t)]>1/2$, then the expectation of the gradient of IBP abstract loss satisfies*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla\overline{\mathcal{L}}_{IBP}(\theta^t)\|_2^2] \le 2\overline{\sigma}\sqrt{\frac{\overline{L}\overline{\Delta}}{T}}(1+\sum_{t=0}^{T-1}(1+\mathbb{E}[\overline{\mathcal{L}}_{IBP}(\theta^t)])^2).$$

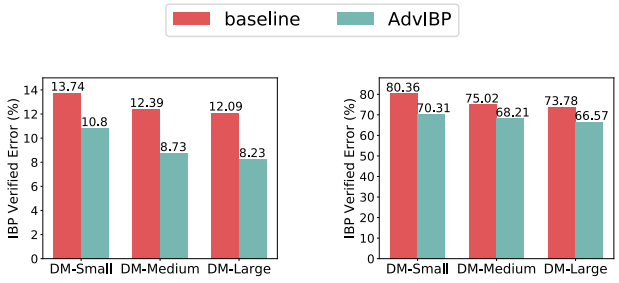
The complete proof can be found in the Appendix. If the two gradients agree with each other (i.e. their dot product is greater than 0), Theorem 1 suggests that the robust loss minimization can converge to a first-order stationary point at a sublinear rate with sufficiently small δ. Using FOSC ensures that the adversarial loss approximates the *robust loss* up to a precision less than δ as in (6). Note that it is difficult

for the perturbed input $\boldsymbol{x}_{\text{adv}}$ to reach the maximum adversarial strength (minimum FOSC value which is 0) as the model becomes more robust during training. Algorithm 2 will mostly prioritize the abstract loss minimization when the two gradients disagree with each other since c_t is decreasing to 0. In this case, Theorem 2 suggests that the abstract loss (as obtained by IBP) minimization can converge to a first-order stationary point at a sublinear rate. Although $\bar{\mathcal{L}}_{\text{IBP}}$ is not guaranteed to converge, our joint training scheme actively reduces the abstract loss to avoid its divergence. In practice, potential divergence of the $\bar{\mathcal{L}}_{\text{IBP}}$ is controlled with a stable training process in our method. Although Theorem 2 requires $\mathbb{E}[\bar{\mathcal{L}}_{\text{IBP}}(\theta^t)]>1/2$, the abstract loss will be sufficiently small if the condition does not hold. With Theorem 1 and 2, the *robust loss* or its upper bound *abstract loss* can be minimized at a sublinear convergence rate. These results provide theoretical support for our approach.

Experiment

Experiment setup. We evaluate *AdvIBP* on all the network model structures used in (Gowal et al. 2018; Zhang et al. 2020) on the MNIST and CIFAR-10 datasets with different l_∞ perturbation bounds, ϵ. We denote these models as **DM-Small**, **DM-Medium** and **DM-Large**. We perform all experiments on a desktop server using at most 4 GeForce GTX 1080 Ti GPUs. All models are trained using a single GPU except for **DM-Large** which requires all 4 GPUs.

Metrics. We use the following metrics to compare the trained neural networks: (i) IBP verified error, which is the percentage of test examples that are not verified by IBP, (ii) standard error, which is the test error evaluated on the clean test dataset, and (iii) PGD error, which is the test error under 200-step PGD attack. Verified errors provide the worst-case test error against l_∞ perturbations. PGD errors provide valid lower bounds of test errors against l_∞ perturbations.



(a) MNIST, $\epsilon = 0.3$ (b) CIFAR-10, $\epsilon = 8/255$

Figure 2: Comparison with the baseline.

Baseline comparison. We consider a baseline method that uses the same warm-up strategy in Algorithm 2 but fixes the coefficients to κ_{adv}=1.0 and κ_{IBP}=1.0 (effectively using the weighted sum method). As shown in Figure 2, *AdvIBP*, which automatically adapts the coefficients, reduces the IBP verified errors by 9.1% to 31.9% compared with the baseline.

Comparison with prior works. Table 2 and 4 shows the standard, verified and PGD errors under different ϵ on CIFAR-10 and MNIST. On CIFAR-10, our method outperforms the state-of-art methods on verified errors obtained from IBP. In addition to CROWN-IBP, we also present the best errors reported by IBP method (Gowal et al. 2018), MIP (Xiao et al. 2019) and COLT (Balunovic and Vechev 2020). Note that MIP (Xiao et al. 2019) reports the verified error obtained by mixed integer programming, which is able to compute the *exact* value of *robust loss*. COLT (Balunovic and Vechev 2020) uses RefineZono to compute the verified errors and RefineZono is supposed to a much higher precision than IBP. On both MNIST and CIFAR-10, even though our method does not use *regular loss*, we still achieve lower standard errors across different models in most cases. The verified errors obtained by *AdvIBP* on MNIST can match the prior state-of-art results. The result of l_∞ perturbation $2/255$ outperforms existing approaches except for the results in (Zhang et al. 2020; Singh et al. 2019). However, we note here that both methods in (Zhang et al. 2020; Singh et al. 2019) use over-approximation methods with better precision in both training and verification, which may result in significant computation overhead and memory requirement. We hypothesize that the main reason for this performance gap is that with a relatively small l_∞ perturbation, the minimization of *IBP abstract loss* reduces the capacity of the models to learn well as reflected by the higher standard errors.

Additionally, we compare *AdvIBP* with CROWN-IBP *across a wide range of neural network models* (Table F) rather than on a few hand-selected models. In Table 3, we present the best, median and worst verified and standard test errors for models trained on MNIST and CIFAR-10 using CROWN-IBP (with default settings) and *AdvIBP* respectively. *AdvIBP*'s best, median and worst verified errors outperform those of CROWN-IBP in almost all cases.

AdvCROWN-IBP. In our joint training scheme, one can replace IBP with a more precise method for computing the abstract loss. We present here the results of *AdvCROWN-IBP* which uses CROWN-IBP to compute the abstract loss on the MNIST dataset. CROWN-IBP uses a linear combination of CROWN bounds and IBP bounds to compute the abstract loss during the warm-up period. After the warm-up period, the abstract loss is computed solely with IBP bounds. In Table 4, we can observe that with a more precise abstract loss, our joint training scheme outperforms CROWN-IBP and *AdvIBP* in IBP verified errors consistently across different model structures. In fact, to the best of our knowledge, *AdvCROWN-IBP* achieves the *best* verified error rates compared to those reported in existing literature on the MNIST dataset across different choices of network models for these ϵ bounds.

Conclusion

We propose a new certified adversarial training framework that bridges the gap between adversarial training and provable robustness from a joint training perspective. We formulate the joint training as a two-objective optimization problem, which facilitates the balance between *adversarial loss* and *abstract loss*. We show that our joint training framework outperforms prior certified adversarial training methods in both standard and verified errors, and achieves state-of-the-art verified test errors for l_∞ robustness.

Table 2: Evaluation on the CIFAR-10 dataset between models trained by *AdvIBP* and those by CROWN-IBP. *AdvIBP* outperforms the state-of-art, CROWN-IBP, and other best reported results under all perturbation and model settings if IBP is used to compute the verified errors. If different network architectures and more precise verification methods are also considered, our IBP verified errors still outperform the best prior results for both $\epsilon = \frac{8}{255}$ and $\epsilon = \frac{16}{255}$.

ϵ (l_∞ norm)	Training Method	DM-Small model's err. (%)			DM-Medium model's err. (%)			DM-Large model's err. (%)			Best errors reported in literature (%)[2]		
		Standard	Verified	PGD	Standard	Verified	PGD	Standard	Verified	PGD	Method	Standard	Verified
$\epsilon=\frac{2}{255}$[1]	CROWN-IBP	38.15	52.57	50.35	32.78	49.57	44.22	28.48	46.03	40.28	IBP method (Gowal et al. 2018)[3] MIP (Xiao et al. 2019) COLT (Balunovic and Vechev 2020)	39.22 38.88 21.60	55.19 54.07 39.50
	AdvIBP	42.33	56.00	50.08	35.36	52.27	43.75	40.61	51.66	46.97			
$\epsilon=\frac{8}{255}$	CROWN-IBP	59.94	70.76	69.65	58.19	68.94	67.72	54.02	66.94	65.42	IBP method (Gowal et al. 2018)[3] MIP (Xiao et al. 2019) COLT (Balunovic and Vechev 2020)	58.43 59.55 48.30	70.81 79.73 72.50
	AdvIBP	57.88	**70.31**	66.52	54.20	**68.21**	61.21	52.86	**66.57**	61.66			
$\epsilon=\frac{16}{255}$	CROWN-IBP	67.42	78.41	76.86	67.94	78.46	77.21	66.06	76.80	75.23	IBP method (Gowal et al. 2018)[3] MIP (Xiao et al. 2019) COLT (Balunovic and Vechev 2020)	68.97 n/a n/a	78.12 n/a n/a
	AdvIBP	67.32	**78.12**	73.44	66.26	**77.79**	73.52	64.40	**76.05**	71.78			

[1] The verified error of CROWN-IBP in this setting is computed using CROWN.
[2] Some of the best errors from literature are obtained from models with different architectures from ours. Some of the verified errors are also obtained using more precise verification methods.
[3] The results are reproduced by (Zhang et al. 2020) on the same perturbation settings and models used by our method and CROWN-IBP. The verified error is obtained from IBP.

Table 3: Standard, verified and PGD test errors *for a wide range of models* trained on MNIST and CIFAR-10 datasets using CROWN-IBP and *AdvIBP*. The purpose of this experiment is to compare model performance statistics on a wide range of models, rather than a few selected models. For each settings, we report 3 statistics, the smallest, median and largest verified errors. We also report the standard and PGD errors in the same way.

Dataset	ϵ (l_∞ norm)	Training Method	Standard Error (%)			Verified Error (%)			PGD Error (%)			Number of AdvIBP models with lower verified errors among all trained model structures
			best	median	worst	best	median	worst	best	median	worst	
MNIST	$\epsilon = 0.2$	CROWN-IBP	2.49	3.50	5.39	4.81	6.33	8.82	3.42	4.94	7.33	9/10
		AdvIBP	2.41	3.36	5.29	**4.76**	**6.13**	**8.52**	3.31	4.70	7.01	
	$\epsilon = 0.3$	CROWN-IBP	2.49	3.50	5.39	**7.19**	9.12	11.58	3.85	5.47	8.46	8/10
		AdvIBP	2.41	3.36	5.29	7.21	**8.86**	**11.32**	4.04	5.40	8.00	
CIFAR-10	$\epsilon = \frac{8}{255}$	CROWN-IBP	57.25	59.84	63.46	69.02	71.32	72.40	65.56	67.57	70.17	7/7
		AdvIBP	57.03	58.85	60.97	**68.50**	**69.36**	**71.40**	65.08	66.90	68.74	

Table 4: Evaluation on the MNIST dataset between models trained by *AdvIBP*, *AdvCROWN-IBP* and those by CROWN-IBP. The CROWN-IBP result is from Table C. in (Zhang et al. 2020). *AdvIBP* achieves competitive performance compared to CROWN-IBP on MNIST. *AdvCROWN-IBP* outperforms CROWN-IBP under all settings, and achieves state-of-the-art verified errors on MNIST dataset for l_∞ robustness.

ϵ (l_∞ norm)	Training Method	DM-Small model's err. (%)			DM-Medium model's err. (%)			DM-Large model's err. (%)		
		Standard	Verified[1]	PGD	Standard	Verified[1]	PGD	Standard	Verified[1]	PGD
$\epsilon = 0.1$	CROWN-IBP	1.67	3.44	3.09	1.14	**2.64**	2.23	0.97	2.25	1.81
	AdvIBP[2]	1.63	3.69	2.70	1.41	3.24	2.26	1.03	2.28	1.53
	AdvCROWN-IBP	1.52	**3.19**	2.39	1.23	2.88	2.18	1.22	**2.19**	1.57
$\epsilon = 0.2$	CROWN-IBP	2.96	6.11	5.74	2.37	5.35	4.90	1.62	3.87	3.81
	AdvIBP[2]	4.15	7.68	5.81	2.33	5.37	3.54	1.58	4.70	2.59
	AdvCROWN-IBP	3.22	**6.02**	4.50	2.45	**5.16**	3.27	1.51	**3.87**	1.98
$\epsilon = 0.3$	CROWN-IBP	3.55	9.40	8.50	2.37	8.54	7.74	1.62	6.68	5.85
	AdvIBP[2]	4.15	10.80	6.83	2.33	8.73	4.35	1.58	8.23	3.17
	AdvCROWN-IBP	3.22	**9.03**	5.42	2.45	**8.31**	3.81	1.90	**6.60**	2.87
$\epsilon = 0.4$	CROWN-IBP	3.78	15.21	13.34	3.16	14.19	11.31	1.62	12.46	9.47
	AdvIBP	4.15	17.57	8.48	2.72	16.18	5.58	1.88	16.57	3.23
	AdvCROWN-IBP	3.22	**14.42**	6.69	2.98	**13.88**	6.38	1.90	**12.30**	3.46

[1] To further probe the *true robustness* of the trained models, we verify the robustness of the *AdvIBP* trained models with a more precise method, RefineZono. The results are shown in Table D in the Appendix.
[2] We have also tested three model structures similar to DM-Small, DM-Medium and DM-Large. Results are reported in Table C in the Appendix. For these models, *AdvIBP* already outperforms CROWN-IBP in all settings.

Acknowledgement

We gratefully acknowledge the support from NSF grant 1646497 and ONR grant N00014-19-1-2496.

Ethics Statement

Broad acceptance and adoption of large-scale deployments of deep learning systems rely critically on their trustworthiness which, in turn, depends on the ability to assess and demonstrate the safety of such systems. Concerns like adversarial robustness already arise with today's deep learning systems and those that may be exacerbated in the future with more complex systems. Our research has the potential to enable the efficient training of robust deep learning systems. It can help unlock deep learning applications that are currently not deployable due to safety, robustness or resource concerns. These applications range from autonomous driving to mobile devices, and can benefit the society at large.

References

Balunovic, M.; and Vechev, M. 2020. Adversarial training and provable defenses: Bridging the gap. In *International Conference on Learning Representations*.

Carlini, N.; and Wagner, D. 2017. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, 39–57. IEEE.

Du, S.; Lee, J.; Li, H.; Wang, L.; and Zhai, X. 2019a. Gradient Descent Finds Global Minima of Deep Neural Networks. In *International Conference on Machine Learning*, 1675–1685.

Du, S. S.; Zhai, X.; Poczos, B.; and Singh, A. 2019b. Gradient descent provably optimizes over-parameterized neural networks. *International Conference on Learning Representations* .

Dvijotham, K.; Stanforth, R.; Gowal, S.; Mann, T. A.; and Kohli, P. 2018. A Dual Approach to Scalable Verification of Deep Networks. In *UAI*, volume 1, 2.

Fan, J.; Huang, C.; Li, W.; Chen, X.; and Zhu, Q. 2019. Towards Verification-Aware Knowledge Distillation for Neural-Network Controlled Systems. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 1–8. IEEE.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and harnessing adversarial examples. *International Conferences on Learning Representations* .

Gowal, S.; Dvijotham, K.; Stanforth, R.; Bunel, R.; Qin, C.; Uesato, J.; Arandjelovic, R.; Mann, T.; and Kohli, P. 2018. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715* .

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

Kingma, D. P.; and Ba, J. 2016. Adam: A method for stochastic optimization. *International Conference on Learning Representations* .

Kurakin, A.; Goodfellow, I.; and Bengio, S. 2017. Adversarial machine learning at scale. *International Conferences on Learning Representations* .

Lee, J.; and Raginsky, M. 2018. Minimax statistical learning with wasserstein distances. In *Advances in Neural Information Processing Systems*, 2687–2696.

Liu, S.; and Vicente, L. N. 2019. The stochastic multi-gradient algorithm for multi-objective optimization and its application to supervised machine learning. *arXiv preprint arXiv:1907.04472* .

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards deep learning models resistant to adversarial attacks. *International Conferences on Learning Representations* .

Mirman, M.; Gehr, T.; and Vechev, M. 2018. Differentiable abstract interpretation for provably robust neural networks. In *International Conference on Machine Learning*, 3578–3586.

Prabhakar, P.; and Afzal, Z. R. 2019. Abstraction based Output Range Analysis for Neural Networks. In *Advances in Neural Information Processing Systems*, 15762–15772.

Raghunathan, A.; Steinhardt, J.; and Liang, P. S. 2018. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems*, 10877–10887.

Ruan, W.; Huang, X.; and Kwiatkowska, M. 2018. Reachability analysis of deep neural networks with provable guarantees. In *International Joint Conferences on Artificial Intelligence*.

Sener, O.; and Koltun, V. 2018. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, 527–538.

Singh, G.; Gehr, T.; Mirman, M.; Püschel, M.; and Vechev, M. 2018. Fast and effective robustness certification. In *Advances in Neural Information Processing Systems*, 10802–10813.

Singh, G.; Gehr, T.; Püschel, M.; and Vechev, M. 2019. Boosting robustness certification of neural networks. *International Conference on Learning Representations* .

Sinha, A.; Namkoong, H.; and Duchi, J. 2018. Certifying some distributional robustness with principled adversarial training. *International Conferences on Learning Representations* .

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2014. Intriguing properties of neural networks. *International Conferences on Learning Representations* .

Tjeng, V.; Xiao, K.; and Tedrake, R. 2019. Evaluating robustness of neural networks with mixed integer programming. *International Conference on Learning Representations* .

Wang, S.; Chen, Y.; Abdou, A.; and Jana, S. 2018. Mixtrain: Scalable training of verifiably robust neural networks. *arXiv preprint arXiv:1811.02625* .

Wang, Y.; Ma, X.; Bailey, J.; Yi, J.; Zhou, B.; and Gu, Q. 2019. On the convergence and robustness of adversarial training. In *International Conference on Machine Learning*, 6586–6595.

Wong, E.; and Kolter, Z. 2018. Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope. In *International Conference on Machine Learning*, 5283–5292.

Wong, E.; Rice, L.; and Kolter, J. Z. 2020. Fast is better than free: Revisiting adversarial training. *International Conferences on Learning Representations* .

Xiao, K. Y.; Tjeng, V.; Shafiullah, N. M.; and Madry, A. 2019. Training for faster adversarial robustness verification via inducing relu stability. *International Conference on Learning Representations* .

Zhang, H.; Chen, H.; Xiao, C.; Li, B.; Boning, D.; and Hsieh, C.-J. 2020. Towards Stable and Efficient Training of Verifiably Robust Neural Networks. *International Conference on Learning Representations* .

Zhang, H.; Weng, T.-W.; Chen, P.-Y.; Hsieh, C.-J.; and Daniel, L. 2018. Efficient neural network robustness certification with general activation functions. In *Advances in neural information processing systems*, 4939–4948.

Zhang, Y.; Yu, W.; and Turk, G. 2019. Learning Novel Policies For Tasks. In *International Conference on Machine Learning*, 7483–7492.