Scalable Verified Training for Provably Robust Image Classification

Sven Gowal* DeepMind sgowal@google.com
Krishnamurthy (Dj) Dvijotham* dvij@google.com
Robert Stanforth* stanforth@google.com
Rudy Bunel
Chongli Qin
Jonathan Uesato
Relja Arandjelović
Timothy Mann
Pushmeet Kohli

Abstract

Recent work has shown that it is possible to train deep neural networks that are provably robust to norm-bounded adversarial perturbations. Most of these methods are based on minimizing an upper bound on the worst-case loss over all possible adversarial perturbations. While these techniques show promise, they often result in difficult optimization procedures that remain hard to scale to larger networks. Through a comprehensive analysis, we show how a simple bounding technique, interval bound propagation (IBP), can be exploited to train large provably robust neural networks that beat the state-of-the-art in verified accuracy. While the upper bound computed by IBP can be quite weak for general networks, we demonstrate that an appropriate loss and clever hyper-parameter schedule allow the network to adapt such that the IBP bound is tight. This results in a fast and stable learning algorithm that outperforms more sophisticated methods and achieves state-of-the-art results on MNIST, CIFAR-10 *and* SVHN. *It also allows us to train the largest model to be verified beyond vacuous bounds on a downscaled version of* IMAGENET.

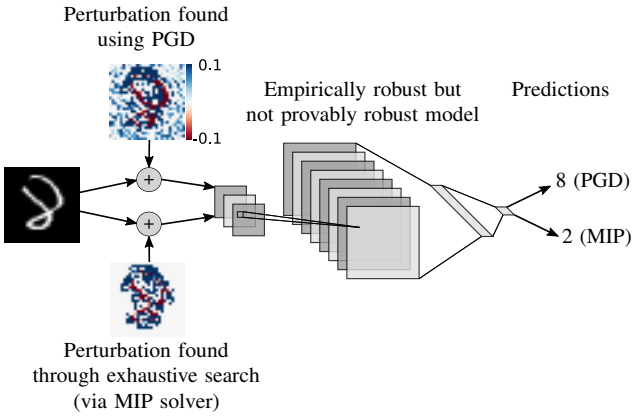


Figure 1: Example motivating why robustness to the projected gradient descent (PGD) attack is not a true measure of robustness (even for small convolutional neural networks). Given a seemingly robust neural network, the worst-case perturbation of size $\epsilon = 0.1$ found using 200 PGD iterations and 10 random restarts (shown at the top) is correctly classified as an "eight". However, a worst case perturbation classified as a "two" can be found through exhaustive search (shown at the bottom).

1. Introduction

Despite the successes of deep learning [1], it is well-known that neural networks are not robust. In particular, it has been shown that the addition of small but carefully chosen deviations to the input, called adversarial perturbations, can cause the neural network to make incorrect predictions with high confidence [2–6]. Starting with Szegedy et al. [6], there has been a lot of work on understanding and generating adversarial perturbations [3, 7], and on building models that are robust to such perturbations [4, 8–10]. Unfortunately, many of the defense strategies proposed in the literature are targeted towards a specific adversary (e.g., obfuscating gradients against projected gradient attacks), and as such they are easily broken by stronger adversaries [11, 12]. Robust optimization techniques, like the one developed by Madry et al. [9], overcome this problem by trying to find the worst-case adversarial examples at each training step and adding them to the training data. While the resulting models show strong empirical evidence that they are robust against many attacks, we cannot yet guarantee that a different adversary (for example, one that does brute-force enumeration to compute adversarial perturbations) cannot find inputs that cause the model to predict incorrectly. In fact, Figure 1 provides an example that motivates why projected gradient descent (PGD) – the technique at the core of Madry et al.'s method – does not always find the worst-case attack (a phenomenon also observed by Tjeng et al. [13]).

This has driven the need for *formal verification*: a provable guarantee that neural networks are consistent with a *specification* for all possible inputs to the network. Substantial progress has been made: from complete methods

that use Satisfiability Modulo Theory (SMT) [14–16] or Mixed-Integer Programming (MIP) [13, 17, 18] to incomplete methods that rely on solving a convex *relaxation* of the verification problem [19–26]. Complete methods, which provide exact robustness bounds, are expensive and difficult to scale (since they perform exhaustive enumeration in the worst case). Incomplete methods provide robustness bounds that can be loose. However, they scale to larger models than complete methods and, as such, can be used inside the training loop to build models that are not only robust, but also intrinsically easier to verify [20, 23, 24, 27].

In this paper, we study interval bound propagation (IBP), which is derived from interval arithmetic [14, 15, 28]: an incomplete method for training verifiably robust classifiers. IBP allows to define a loss to minimize an upper bound on the maximum difference between any pair of logits when the input can be perturbed within an ℓ_∞ norm-bounded ball. Compared to more sophisticated approaches [20, 23, 24, 27], IBP is very fast – its computational cost is comparable to two forward passes through the network. This enables us to have a much faster training step, allowing us to scale to larger models with larger batch sizes and perform more extensive hyper-parameter search. While the core approach behind IBP has been studied to some extent in previous papers [20, 23], blindly using it results in a difficult optimization problem with unstable performance. Most notably, we develop a training curriculum and show that this approach can achieve strong results, outperforming the state-of-the-art. The contributions of this paper are as follows:

- We propose several enhancements that improve the performance of IBP for verified training. In particular, we differentiate ourselves from Mirman et al. [20] by using a different loss function, and by eliding the last linear layer of the neural network, thereby improving our estimate of the worst-case logits. We also develop a curriculum that stabilizes training and improves generalization.
- We compare our trained models to those from other approaches in terms of robustness to PGD attacks [3] and show that they are competitive against Madry et al. [9] and Wong et al. [25] across a wide range of ℓ_∞ perturbation radii (hereafter denoted by ϵ). We also compare IBP to Wong et al.'s method in terms of verified error rates.
- We demonstrate that IBP is not only computationally cheaper, but that it also achieves the state-of-the-art verified accuracy for single-model architecture.[1] We reduce the verified error rate from 3.67% to 2.23% on MNIST (with ℓ_∞ perturbations of $\epsilon = 0.1$[2]), from 19.32% to 8.05% on MNIST (at $\epsilon = 0.3$), and from 78.22% to 67.96% on CIFAR-10 (at $\epsilon = 8/255$). Thus, demonstrating the extent to which the model is able to adapt itself during training so that the simple relaxation induced by IBP is not too weak.
- We train the first provably robust model on IMAGENET (downscaled to 64×64 images) at $\epsilon = 1/255$. Using a WideResNet-10-10, we reach 93.87% top-1 verified error rate. This constitutes the largest model to be verified beyond vacuous bounds (a random or constant classifier would achieve a 99.9% verified error rate).
- Finally, the code for training provably robust neural networks using IBP is available at https://github.com/deepmind/interval-bound-propagation.

[1] The use of ensembles or cascades (as done by Wong et al. [25]) is orthogonal to the work presented here.

[2] ϵ is measured with respect to images normalized between 0 and 1.

2. Related Work

Work on training verifiably robust neural networks typically falls in one of two primary categories. First, there are empirical approaches exemplified perfectly by Xiao et al. [29]. This work takes advantage of the nature of MIP-based verification – the critical bottleneck being the number of integer variables the solver needs to branch over. The authors design a regularizer that aims to reduce the number of ambiguous ReLU activation units (units for which bound propagation is not able to determine whether they are on or off) so that verification after training using a MIP solver is efficient. This method, while not providing any meaningful measure of the underlying verified accuracy during training, is able to reach state-of-the-art performance once verified after training with a MIP solver.

Second, there are methods that compute a differentiable upper bound on the violation of the specification to verify. This upper bound, if fast to compute, can be used within a loss (e.g., hinge loss) to optimize models through regular Stochastic Gradient Descent (SGD). In this category, we highlight the works by Raghunathan et al. [27], Wong et al. [25], Dvijotham et al. [23] and Mirman et al. [20]. Raghunathan et al. [27] use a semi-definite relaxation that provides an adaptive regularizer that encourages robustness. Wong et al. [25] extend their previous work [24], which considers the dual formulation of the underlying LP. Critically, any feasible dual solution provides a guaranteed upper bound on the solution of the primal problem. This allows Wong and Kolter to fix the dual solution and focus on computing tight activation bounds that, in turn, yield a tight upper bound on the specification violation. Alternatively, Dvijotham et al. [23] fix the activation bounds and optimize the dual solution using an additional *verifier* network. Finally, Mirman et al. [20] introduce geometric abstractions that bound activations as they propagate through the network. To the contrary of the conclusions from these previous works, we demonstrate that tighter relaxations (such as the dual formulation from Dvijotham et al. [23], or the *zonotope domain* from Mirman

et al. [20]) are not necessary to reach tight verified bounds.

IBP, which often leads to loose upper bounds for arbitrary networks, has a significant computational advantage, since computing IBP bounds only requires two forward passes through the network. This enables us to apply IBP to significantly larger models and train with extensive hyperparameter tuning. We show that thanks to this capability, a carefully tuned verified training process using IBP is able to achieve state-of-the-art verified accuracy. Perhaps surprisingly, our results show that neural networks can easily adapt to make the rather loose bound provided by IBP much tighter – this is in contrast to previous results that seemed to indicate that more expensive verification procedures are needed to improve the verified accuracy of neural networks in image classification tasks.

3. Methodology

Neural network. We focus on feed-forward neural networks trained for classification tasks. The input to the network is denoted x_0 and its output is a vector of raw unnormalized predictions (hereafter logits) corresponding to its beliefs about which class x_0 belongs to. During training, the network is fed pairs of input x_0 and correct output label y_{true}, and trained to minimize a misclassification loss, such as cross-entropy.

For clarity of presentation, we assume that the neural network is defined by a sequence of transformations h_k for each of its K layers. That is, for an input z_0 (which we define formally in the next paragraph), we have

$$z_k = h_k(z_{k-1}) \quad k = 1, \ldots, K \tag{1}$$

The output $z_K \in \mathbb{R}^N$ has N logits corresponding to N classes.

Verification problem. We are interested in verifying that neural networks satisfy a specification by generating a proof that this specification holds. We consider specifications that require that for all inputs in some set $\mathcal{X}(x_0)$ around x_0, the network output satisfies a linear relationship

$$c^\mathsf{T} z_K + d \leq 0 \quad \forall z_0 \in \mathcal{X}(x_0) \tag{2}$$

where c and d are a vector and a scalar that may depend on the nominal input x_0 and label y_{true}. As shown by Dvijotham et al. [22], many useful verification problems fit this definition. In this paper, we focus on the robustness to adversarial perturbations within some ℓ_∞ norm-bounded ball around the nominal input x_0.

A network is adversarially robust at a point x_0 if there is no choice of adversarial perturbation that changes the classification outcome away from the true label y_{true}, i.e., $\operatorname{argmax}_i z_{K,i} = y_{\text{true}}$ for all elements $z_0 \in \mathcal{X}(x_0)$. Formally, we want to verify that for each class y:

$$(e_y - e_{y_{\text{true}}})^\mathsf{T} z_K \leq 0 \quad \forall z_0 \in \mathcal{X}(x_0) = \{x \mid \|x - x_0\|_\infty < \epsilon\} \tag{3}$$

where e_i is the standard i^{th} basis vector and ϵ is the perturbation radius.

Verifying a specification like (2) can be done by searching for a counter-example that violates the specification constraint:

$$\begin{aligned} \max_{z_0 \in \mathcal{X}(x_0)} \quad & c^\mathsf{T} z_K + d \\ \text{subject to} \quad & z_k = h_k(z_{k-1}) \quad k = 1, \ldots, K \end{aligned} \tag{4}$$

If the optimal value of the above optimization problem is smaller than 0, the specification (2) is satisfied.

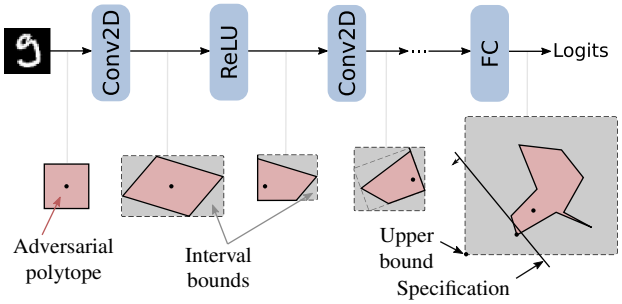


Figure 2: Illustration of interval bound propagation. From the left, the adversarial polytope (illustrated in 2D for clarity) of the nominal image of a "nine" (in red) is propagated through a convolutional network. At each layer, the polytope deforms itself until the last layer where it takes a complicated and non-convex shape in logit space. Interval bounds (in gray) can be propagated similarly: after each layer the bounds are reshaped to be axis-aligned bounding boxes that always encompass the adversarial polytope. In logit space, it becomes easy to compute an upper bound on the worst-case violation of the specification to verify.

Interval bound propagation. IBP's goal is to find an upper bound on the optimal value of the problem (4). The simplest approach is to bound the activation z_k of each layer by an axis-aligned bounding box (i.e., $\underline{z}_k(\epsilon) \leq z_k \leq \overline{z}_k(\epsilon)$ [3]) using interval arithmetic. For ℓ_∞ adversarial perturbations of size ϵ, we have for each coordinate $z_{k,i}$ of z_k:

$$\begin{aligned} \underline{z}_{k,i}(\epsilon) &= \min_{\underline{z}_{k-1}(\epsilon) \leq z_{k-1} \leq \overline{z}_{k-1}(\epsilon)} e_i^\mathsf{T} h_k(z_{k-1}) \\ \overline{z}_{k,i}(\epsilon) &= \max_{\underline{z}_{k-1}(\epsilon) \leq z_{k-1} \leq \overline{z}_{k-1}(\epsilon)} e_i^\mathsf{T} h_k(z_{k-1}) \end{aligned} \tag{5}$$

where $\underline{z}_0(\epsilon) = x_0 - \epsilon \mathbf{1}$ and $\overline{z}_0(\epsilon) = x_0 + \epsilon \mathbf{1}$. The above optimization problems can be solved quickly and in closed

[3] For simplicity, we abuse the notation $\leq$ to mean that all coordinates from the left-hand side need to be smaller than the corresponding coordinates from the right-hand side.

form for affine layers and monotonic activation functions. An illustration of IBP is shown in Figure 2.

For the **affine layers** (e.g., fully connected layers, convolutions) that can be represented in the form $h_k(z_{k-1}) = Wz_{k-1} + b$, solving the optimization problems (5) can be done efficiently with only two matrix multiplies:

$$\begin{aligned} \mu_{k-1} &= \frac{\overline{z}_{k-1} + \underline{z}_{k-1}}{2} \\ r_{k-1} &= \frac{\overline{z}_{k-1} - \underline{z}_{k-1}}{2} \\ \mu_k &= W\mu_{k-1} + b \\ r_k &= |W| r_{k-1} \\ \underline{z}_k &= \mu_k - r_k \\ \overline{z}_k &= \mu_k + r_k \end{aligned} \tag{6}$$

where $|\cdot|$ is the element-wise absolute value operator. Propagating bounds through any element-wise **monotonic activation function** (e.g., ReLU, tanh, sigmoid) is trivial. Concretely, if h_k is an element-wise increasing function, we have:

$$\begin{aligned} \underline{z}_k &= h_k(\underline{z}_{k-1}) \\ \overline{z}_k &= h_k(\overline{z}_{k-1}) \end{aligned} \tag{7}$$

Notice how for element-wise non-linearities the $(\underline{z}_k, \overline{z}_k)$ formulation is better, while for affine transformations (μ_k, r_k) is more efficient (requiring two matrix multiplies instead of four). Switching between parametrizations depending on h_k incurs a slight computational overhead, but since affine layers are typically more computationally intensive, the formulation (6) is worth it.

Finally, the upper and lower bounds of the output logits z_K can be used to construct an upper bound on the solution of (4):

$$\max_{\underline{z}_K(\epsilon) \leq z_K \leq \overline{z}_K(\epsilon)} c^\mathsf{T} z_K + d \tag{8}$$

Overall, the adversarial specification (3) is upper-bounded by $\overline{z}_{K,y}(\epsilon) - \underline{z}_{K,y_{\text{true}}}(\epsilon)$. It corresponds to an upper bound on the worst-case logit difference between the true class y_{true} and any other class y.

Elision of the last layer. Bound propagation is not necessary for the last linear layer of the network. Indeed, we can find an upper bound to the solution of (4) that is tighter than proposed by (8) by eliding the final linear layer with the specification. Assuming $h_K(z_{K-1}) = Wz_{K-1} + b$, we have:

$$\begin{aligned} &\max_{\substack{\underline{z}_K \leq z_K \leq \overline{z}_K \\ z_K = h_K(z_{K-1})}} c^\mathsf{T} z_K + d \\ &\geq \max_{\underline{z}_{K-1} \leq z_{K-1} \leq \overline{z}_{K-1}} c^\mathsf{T} h_K(z_{K-1}) + d \\ &= \max_{\underline{z}_{K-1} \leq z_{K-1} \leq \overline{z}_{K-1}} c^\mathsf{T} W z_{K-1} + c^\mathsf{T} b + d \\ &= \max_{\underline{z}_{K-1} \leq z_{K-1} \leq \overline{z}_{K-1}} c'^\mathsf{T} z_{K-1} + d' \end{aligned} \tag{9}$$

with $c' = W^\mathsf{T} c$ and $d' = c^\mathsf{T} b + d$, which bypasses the additional relaxation induced by the last linear layer.

Loss. In the context of classification under adversarial perturbation, solving the optimization problem (8) for each target class $y \neq y_{\text{true}}$ results in a set of worst-case logits where the logit of the true class is equal to its lower bound and the other logits are equal to their upper bound:

$$\hat{z}_{K,y}(\epsilon) = \begin{cases} \overline{z}_{K,y}(\epsilon) & \text{if } y \neq y_{\text{true}} \\ \underline{z}_{K,y_{\text{true}}}(\epsilon) & \text{otherwise} \end{cases} \tag{10}$$

That is for all $y \neq y_{\text{true}}$, we have

$$(e_y - e_{y_{\text{true}}})^\mathsf{T} \hat{z}_K(\epsilon) = \max_{\underline{z}_K(\epsilon) \leq z_K \leq \overline{z}_K(\epsilon)} (e_y - e_{y_{\text{true}}})^\mathsf{T} z_K \tag{11}$$

We can then formulate our training loss as

$$L = \kappa \underbrace{\ell(z_K, y_{\text{true}})}_{L_{\text{fit}}} + (1-\kappa) \underbrace{\ell(\hat{z}_K(\epsilon), y_{\text{true}})}_{L_{\text{spec}}} \tag{12}$$

where ℓ is the cross-entropy loss and κ is a hyperparameter that governs the relative weight of satisfying the specification (L_{spec}) versus fitting the data (L_{fit}). If $\epsilon = 0$ then $z_K = \hat{z}_K(\epsilon)$, and thus (12) becomes equivalent to a standard classification loss.

Training procedure. To stabilize the training process and get a good trade-off between nominal and verified accuracy under adversarial perturbation, we create a learning curriculum by scheduling the values of κ and ϵ:

- κ controls the relative weight of satisfying the specification versus fitting the data. Hence, we found that starting with $\kappa = 1$ and slowly reducing it throughout training helps get more balanced models with higher nominal accuracy. In practice, we found that using a final value of $\kappa = 1/2$ works well on MNIST, CIFAR-10, SVHN and IMAGENET.
- More importantly, we found that starting with $\epsilon = 0$ and slowly raising it up to a target perturbation radius ϵ_{train} is necessary. We note that ϵ_{train} does not need to be equal to the perturbation radius used during testing, using higher values creates robust models that generalize better.

Additional details that relate to specific datasets are available in the supplementary material in Appendix A.

4. Results

We demonstrate that IBP can train verifiably robust networks and compare its performance to state-of-the-art methods on MNIST, CIFAR-10 and SVHN. Highlights include an improvement of the verified error rate from 3.67% to 2.23% on MNIST at $\epsilon = 0.1$, from 19.32% to 8.05% on MNIST at $\epsilon = 0.3$, and from 78.22% to 67.96% on CIFAR-10 at $\epsilon = 8/255$. We also show that IBP can scale to larger networks by training a model on downscaled IMAGENET that reaches a non-vacuous verified error rate of 93.87% at $\epsilon = 1/255$. Finally, Section 4.3 illustrates how training with the loss function and curriculum from Section 3 allows the training process to adapt the model to ensure that the bound computed by IBP is tight.

Unless stated otherwise, we compute the empirical adversarial accuracy (or error rate) on the test set using 200 untargeted PGD steps and 10 random restarts. As the verified error rate computed for a network varies greatly with the verification method, we calculate it using an exact solver. Several previous works have shown that training a network with a loss function derived from a specific verification procedure renders the network amenable to verification using that specific procedure only [23, 24, 27]. In order to circumvent this issue and present a fair comparison, we use a complete verification algorithm based on solving a MIP – such an algorithm is expensive as it performs a brute force enumeration in the worst case. However, in practice, we find that commercial MIP solvers like Gurobi can handle verification problems from moderately sized networks. In particular, we use the MIP formulation from Tjeng et al. [13]. For each example of the test set, a MIP is solved using Gurobi with a timeout of 10 minutes. Upon timeout, we fallback to solving a relaxation of the verification problem with a LP [15] using Gurobi again. When both approaches fail to provide a solution within the imparted time, we count the example as attackable. Thus, the verified error rate reported may be over-estimating the exact verified error rate.[4] We always report results with respect to the complete test set of 10K images for both MNIST and CIFAR-10, and 26K images for SVHN. For downscaled IMAGENET, we report results on the validation set of 10K images.

4.1. MNIST, CIFAR-10 and SVHN

We compare IBP to three alternative approaches: the nominal method, which corresponds to standard training with cross-entropy loss; adversarial training, following Madry et al. [9], which generates adversarial examples on the fly during training; and Wong et al. [25], which trains models that are provably robust. We train three different model architectures for each of the four methods (see Table 1). The first two models (i.e., **small** and **medium**) are equivalent to the small and large models in Wong et al. [25]. [5] The third model (i.e., **large**) is significantly larger (in terms of number of hidden units) than any other verified model presented in the literature. On MNIST, for each model and each method, we trained models that are robust to a wide range of perturbation radii by setting ϵ_{train} to $0.1, 0.2, 0.3$ or 0.4. During testing, we test each of these 12 models against $\epsilon \in [0, 0.45]$. On CIFAR-10, we train the same models and methods with $\epsilon_{\text{train}} \in \{2/255, 8/255\}$ and test on the same $\epsilon = \epsilon_{\text{train}}$ value. On SVHN we used $\epsilon_{\text{train}} = 0.01$ and only test on $\epsilon = \epsilon_{\text{train}}$.

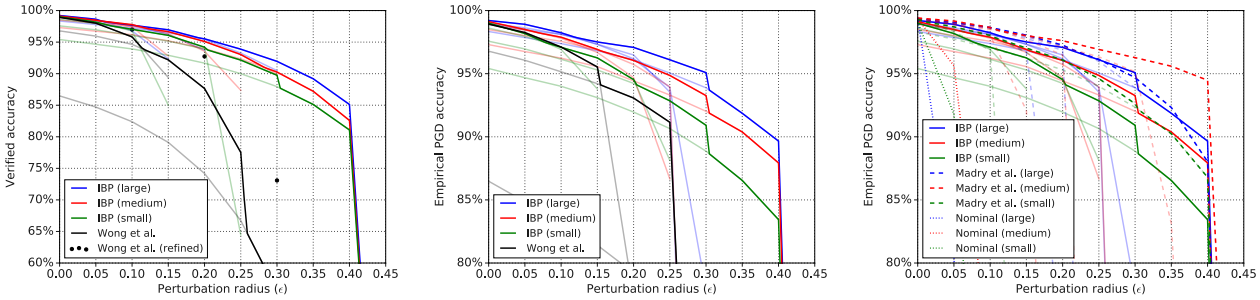
small	medium	large
CONV 16 4×4+2	CONV 32 3×3+1	CONV 64 3×3+1
CONV 32 4×4+1	CONV 32 4×4+2	CONV 64 3×3+1
FC 100	CONV 64 3×3+1	CONV 128 3×3+2
	CONV 64 4×4+2	CONV 128 3×3+1
	FC 512	CONV 128 3×3+1
	FC 512	FC 512
# hidden: 8.3K	47K	230K
# params: 471K	1.2M	17M

Table 1: Architecture of the three models used on MNIST, CIFAR-10 and SVHN. All layers are followed by RELU activations. The last fully connected layer is omitted. "CONV k w×h+s" corresponds to a 2D convolutional layer with k filters of size $w \times h$ using a stride of s in both dimensions. "FC n" corresponds to a fully connected layer with n outputs. The last two rows are the number of hidden units (counting activation units only) and the number of parameters when training on CIFAR-10.

Figures 3a and b compare IBP to Wong et al. on MNIST for all perturbation radii between 0 and 0.45 across all models. Remember that we trained each model architecture against many ϵ_{train} values. The bold lines show for each model architecture, the model trained with the perturbation radius ϵ_{train} that performed best for a given ϵ (i.e., x-axis) The faded lines show all individual models. Across the full spectrum, IBP achieves good accuracy under PGD attacks and higher provable accuracy (computed by an exact verifier). We observe that while Wong et al. is competitive at small perturbation radii (when both ϵ and ϵ_{train} are small), it quickly degrades as the perturbation radius increases (when ϵ_{train} is large). For completeness, Figure 3c also compares IBP to Madry et al. with respect to the empirical accuracy against PGD attacks of varying intensities. We observe that IBP tends to be slightly worse than Madry et al. for similar network sizes – except for the large model where Madry et al.

[4] As an example, for the small model trained using Wong et al., there are 3 timeouts at $\epsilon = 0.1$, 18 timeouts at $\epsilon = 0.2$ and 58 timeouts at $\epsilon = 0.3$ for the 10K examples of the MNIST test set. These timeouts would amount to a maximal over-estimation of 0.03%, 0.18% and 0.58% in verified error rate, respectively.

[5] We do not train our large model with Wong et al. as we could not scale this method beyond the medium sized model.



(a) (b) (c)

Figure 3: Accuracy against different adversarial perturbations: (a) shows the verified/provable worst-case accuracy compared to Wong et al., (b) shows the empirical adversarial accuracy computed by running PGD compared to Wong et al., and (c) shows the empirical adversarial accuracy computed by running PGD compared to Madry et al.. Faded lines show individual models of a given size (i.e., small, medium and large) trained with $\epsilon_{\text{train}} = \{0.1, 0.2, 0.3, 0.4\}$, while bold lines show the best accuracy across across all ϵ_{train} values for each model size. In (a), for Wong et al., the dots correspond to exact bounds computed using a MIP solver, while the black bold line corresponds to a lower bound computed using [25] without random projections.

is likely overfitting (as it performs worse than the medium-sized model).

Table 4 provides additional results and also includes results from the literature. The test error corresponds to the test set error rate when there is no adversarial perturbation. For models that we trained ourselves, the PGD error rate is calculated using 200 iterations of PGD and 10 random restarts. The verified bound on the error rate is obtained using the MIP/LP cascade described earlier. A dash is used to indicate when we could not verify models beyond trivial bounds within the imparted time limits. For such cases, it is useful to consider the PGD error rate as a lower bound on the verified error rate. All methods use the same model architectures (except results from the literature). For clarity, we do not report the results for all ϵ_{train} values and all model architectures (Table 6 in the appendix reports additional results). Figure 3 already shows the effect of ϵ_{train} and model size in a condensed form. Compared to Wong et al., IBP achieves lower error rates under normal and adversarial conditions, as well as better verifiable bounds, setting the state-of-the-art in verified robustness to ℓ_∞-bounded adversarial attacks on most pairs of dataset and perturbation radius. Additionally, IBP remains competitive against Madry et al. by achieving a lower PGD error rate on CIFAR-10 with $\epsilon = 8/255$ (albeit at the cost of an increased nominal error rate).[6] CIFAR-10 with $\epsilon = 2/255$ is the only combination where IBP is worse than Wong et al. [25]. From our experience, the method from Wong et al. is more effective when the perturbation radius is small (as visible on Figure 3a), thus giving a marginally better feedback when training on CIFAR-10 at $\epsilon = 2/255$. Additional results are available in Appendix D. Appendix C also details an ablative study that demonstrates that *(i)* using cross-entropy (rather than a hinge or softplus loss on each specification) improves verified accuracy across all datasets and model sizes, that *(ii)* eliding the last linear layer also provides a small but consistent improvement (especially for models with limited capacity), and that *(iii)* the schedule on ϵ is necessary.

Finally, we note that when training the small network on MNIST with a Titan Xp GPU (where standard training takes 1.5 seconds per epoch), IBP only takes 3.5 seconds per epoch compared to 8.5 seconds for Madry et al. and 2 minutes for Wong et al. (using random projection of 50 dimensions). Indeed, as detailed in Section 3 (under the paragraph interval bound propagation), IBP creates only two additional passes through the network compared to Madry et al. for which we used seven PGD steps.

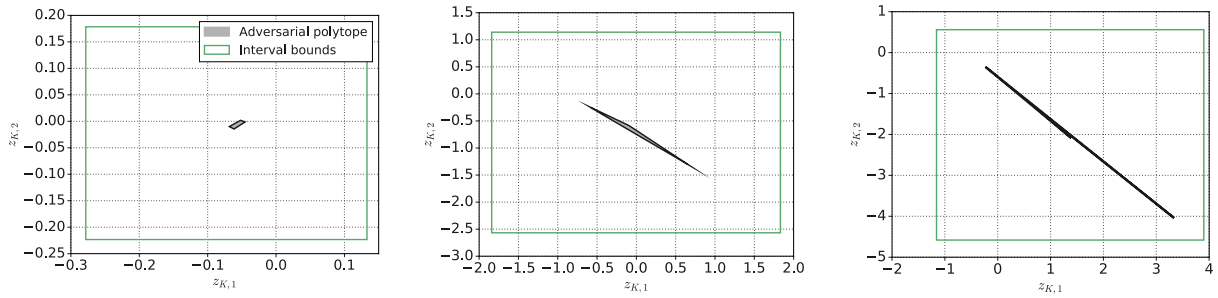
[6] This result only holds for our constrained set of network sizes. The best known empirical adversarial error rate for CIFAR-10 at $\epsilon = 8/255$ using Madry et al. is 52.96% when using 20 PGD steps and no restarts. As a comparison, our large model on CIFAR-10 achieves an empirical adversarial error rate of 60.1% when using 20 PGD steps and no restarts.

ϵ	Method	Test error	PGD	Verified
$1/255$	Nominal	**48.84%**	100.00%	–
	Madry et al.	51.52%	**70.03%**	–
	IBP	84.04%	90.88%	**93.87%**

Table 2: **Downscaled IMAGENET results.** Comparison of the nominal test error (under no perturbation), empirical PGD error rate, and verified bound on the error rate. The verified error rate is computed using IBP bounds only, as running a complete solver is too slow for this model.

4.2. Downscaled IMAGENET

This section demonstrates the scalability of IBP by training, to the best of our knowledge, the first model with non-vacuous verifiable bounds on IMAGENET. We train



(a) Training step 0 (b) Training step 200 (c) Training step 800

Figure 4: Evolution of the adversarial polytope (in gray) around the same input during training. The outer approximation computed using IBP is shown in green.

a WideResNet-10-10 with 8M parameters and 1.8M hidden units, almost an order of magnitude greater than the number of hidden units in our large network. The results in Table 2 are obtained through standard non-robust training, adversarial training, and robust training using IBP on downscaled images (i.e., 64×64). We use all 1000 classes and measure robustness (either empirical or verifiably) using the same one-vs-all scheme used for MNIST, CIFAR-10 and SVHN. Additional details are available in the supplementary material in Appendix A. We realize that these results are pale in comparison to the nominal accuracy obtained by larger non-robust models (i.e., Real et al. [30] achieving 16.1% top-1 error rate). However, we emphasize that no other work has formally demonstrated robustness to norm-bounded perturbation on IMAGENET, even for small perturbations like $\epsilon = 1/255$.

4.3. Tightness

Figure 4 shows the evolution of an adversarial polytope and its outer approximation during training. In this setup (similar to Wong and Kolter [24]), we train a 2-100-100-100-2 network composed of fully-connected layers with ReLU activations on a toy two-dimensional dataset. This dataset consists of 13 randomly drawn 2-dimensional points in $[0, 1]^2$, five of which are from the positive class. The ℓ_∞ distance between each pair of points is at least 0.08, which corresponds to the ϵ and ϵ_{train} values used during testing and training, respectively. The adversarial polytope at the last layer (shown in gray) is computed by densely sampling inputs within an ℓ_∞-norm bounded ball around a nominal input (corresponding to one of the positive training examples). The outer bounds (in green) correspond to the interval bounds at the last layer computed using (5). We observe that, while initially the bounds are very loose, they do become tighter as training progresses.

To judge the tightness of IBP quantitatively, we compare the final verified error rate obtained using the MIP/LP cascade describe earlier with the upper bound estimates from IBP only. Table 3 shows the differences. We observe that IBP itself is a good estimate of the verified error rate and provides estimates that are competitive with more sophisticated solvers (when models are trained using IBP). While intuitive, it is surprising to see that the IBP bounds are so close to the MIP bounds. This highlights that verification becomes easier when models are trained to be verifiable as a simple method like IBP can verify a large proportion of the MIP verified samples. This phenomenon was already observed by Dvijotham et al. [23] and Xiao et al. [29] and explains why some methods cannot be verified beyond trivial bounds within a reasonable computational budget.

Dataset	Epsilon	IBP bound	MIP bound
MNIST	$\epsilon = 0.1$	2.92%	2.23%
	$\epsilon = 0.2$	4.53%	4.48%
	$\epsilon = 0.3$	8.21%	8.05%
	$\epsilon = 0.4$	15.01%	14.88%
CIFAR-10	$\epsilon = 2/255$	55.88%	49.98%
	$\epsilon = 8/255$	68.44%	67.96%
SVHN	$\epsilon = 0.01$	39.35%	37.60%

Table 3: **Tightness of IBP verified bounds on the error rate.** This table compares the verified bound on the error rate obtained using the MIP/LP cascade with the estimates from IBP only (obtained using the worst-case logits from (10)). The models are the ones reported in Table 4.

5. Conclusion

We have presented an approach for training verifiable models and provided strong baseline results for MNIST, CIFAR-10, SVHN and downscaled IMAGENET. Our experiments have shown that the proposed approach outperforms competing techniques in terms of verified bounds on adversarial error rates in image classification problems, while also training faster. In the future, we hope that these results can serve as a useful baseline. We believe that this is an important step towards the vision of specification-driven ML.

Dataset	Epsilon	Method	Test error	PGD	Verified
MNIST	$\epsilon = 0.1$	Nominal	0.65%	27.72%	–
		Madry et al. ($\epsilon_{\text{train}} = 0.2$)	**0.59%**	**1.34%**	–
		Wong et al. ($\epsilon_{\text{train}} = 0.1$)	1.08%	2.89%	3.01%
		IBP ($\epsilon_{\text{train}} = 0.2$)	1.06%	2.11%	**2.23%**
		Reported in literature*			
		Xiao et al. [29]**	1.05%	3.42%	4.40%
		Wong et al. [25]	1.08%	–	3.67%
		Dvijotham et al. [23]	1.20%	2.87%	4.44%
MNIST	$\epsilon = 0.2$	Nominal	**0.65%**	99.57%	–
		Madry et al. ($\epsilon_{\text{train}} = 0.4$)	0.70%	**2.39%**	–
		Wong et al. ($\epsilon_{\text{train}} = 0.2$)	3.22%	6.93%	7.27%
		IBP ($\epsilon_{\text{train}} = 0.4$)	1.66%	3.90%	**4.48%**
		Reported in literature			
		Xiao et al. [29]	1.90%	6.86%	10.21%
MNIST	$\epsilon = 0.3$	Nominal	**0.65%**	99.63%	–
		Madry et al. ($\epsilon_{\text{train}} = 0.4$)	0.70%	**3.73%**	–
		Wong et al. ($\epsilon_{\text{train}} = 0.3$)	13.52%	26.16%	26.92%
		IBP ($\epsilon_{\text{train}} = 0.4$)	1.66%	6.12%	**8.05%**
		Reported in literature			
		Madry et al. [9]	1.20%	6.96%	–
		Xiao et al. [29]	2.67%	7.95%	19.32%
		Wong et al. [25]	14.87%	–	43.10%
MNIST	$\epsilon = 0.4$	Nominal	**0.65%**	99.64%	–
		Madry et al. ($\epsilon_{\text{train}} = 0.4$)	0.70%	**5.52%**	–
		IBP ($\epsilon_{\text{train}} = 0.4$)	1.66%	10.34%	**14.88%**
CIFAR-10	$\epsilon = 2/255$	Nominal	16.66%	87.24%	–
		Madry et al. ($\epsilon_{\text{train}} = 2/255$)	**15.54%**	**42.01%**	–
		Wong et al. ($\epsilon_{\text{train}} = 2/255$)	36.01%	45.11%	49.96%
		IBP ($\epsilon_{\text{train}} = 2/255$)	29.84%	45.09%	49.98%
		Reported in literature			
		Xiao et al. [29]	38.88%	50.08%	54.07%
		Wong et al. [25]	31.72%	–	**46.11%**
CIFAR-10	$\epsilon = 8/255$	Nominal	16.66%	100.00%	100.00%
		Madry et al. ($\epsilon_{\text{train}} = 8/255$)	20.33%	75.95%	–
		Wong et al. ($\epsilon_{\text{train}} = 8/255$)	71.03%	78.14%	79.21%
		IBP ($\epsilon_{\text{train}} = 8/255$)	50.51%	65.23%	**67.96%**
		Reported in literature			
		Madry et al. [9]	**12.70%**	**52.96%**	–
		Xiao et al. [29]	59.55%	73.22%	79.73%
		Wong et al. [25]	71.33%	–	78.22%
		Dvijotham et al. [23]***	51.36%	67.28%	73.33%
SVHN	$\epsilon = 0.01$	Nominal	5.13%	94.14%	–
		Madry et al. ($\epsilon_{\text{train}} = 0.01$)	6.18%	**29.06%**	–
		Wong et al. ($\epsilon_{\text{train}} = 0.01$)	18.10%	32.41%	37.96%
		IBP ($\epsilon_{\text{train}} = 0.01$)	14.82%	32.46%	**37.60%**
		Reported in literature			
		Wong and Kolter [24]	20.38%	33.74%	40.67%
		Dvijotham et al. [23]	16.59%	33.14%	**37.56%**

Table 4: **Comparison with the state-of-the-art.** Comparison of the nominal test error (no adversarial perturbation), error rate under PGD attacks, and verified bound on the error rate. The PGD error rate is calculated using 200 iterations of PGD and 10 random restarts. Dashes "–" indicate that we were unable to verify these networks beyond the trivial 100% error rate bound within the imparted time limit; for such cases we know that the verified error rate must be at least as large as the PGD error rate. For the models we trained ourselves, we indicate the ϵ_{train} that lead to the lowest verified (or – when not available – empirical) error rate. Results from Mirman et al. [20] are only included in Appendix D as, due to differences in image normalization, different ϵ were used; we confirmed with the authors that our IBP results are significantly better.

* Results reported from the literature may use different network architectures. Their empirical PGD error rate may have been computed with a different number of PGD steps and a different number of restarts (when possible we chose the closest setting to ours). Except for the results from Xiao et al. [29], the reported verified bound on the error rate is not computed with an exact solver and may be over-estimated.

** For this model, Xiao et al. [29] only provides estimates computed from 1000 samples (rather than the full 10K images).

*** Dvijotham et al. [23] use a slightly smaller $\epsilon = 0.03 = 7.65/255$ for CIFAR-10.

References

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*. MIT Press, 2016. 1

[2] Nicholas Carlini and David Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM, 2017, pp. 3–14. 1

[3] ——, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy*. IEEE, 2017, pp. 39–57. 1, 2

[4] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014. 1

[5] Alexey Kurakin, Ian Goodfellow, and Samy Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[6] 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. 1

[7] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok, "Synthesizing robust adversarial examples," in *International Conference on Machine Learning*, 2018, pp. 284–293. 1

[8] Nicolas Papernot, Patrick Drew McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE Symposium on Security and Privacy, SP 2016*. Institute of Electrical and Electronics Engineers Inc., 2016, pp. 582–597. 1

[9] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018. 1, 2, 5, 8, 11

[10] Harini Kannan, Alexey Kurakin, and Ian Goodfellow, "Adversarial logit pairing," *arXiv preprint arXiv:1803.06373*, 2018. 1

[11] Jonathan Uesato, Brendan ODonoghue, Pushmeet Kohli, and Aaron Oord, "Adversarial risk and the dangers of evaluating against weak attacks," in *International Conference on Machine Learning*, 2018, pp. 5032–5041. 1

[12] Anish Athalye, Nicholas Carlini, and David Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *International Conference on Machine Learning*, 2018, pp. 274–283. 1

[13] Vincent Tjeng, Kai Y. Xiao, and Russ Tedrake, "Evaluating robustness of neural networks with mixed integer programming," in *International Conference on Learning Representations*, 2019. 1, 2, 5

[14] 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*. Springer, 2017, pp. 97–117. 2

[15] Ruediger Ehlers, "Formal verification of piece-wise linear feed-forward neural networks," in *International Symposium on Automated Technology for Verification and Analysis*. Springer, 2017, pp. 269–286. 2, 5

[16] Nicholas Carlini, Guy Katz, Clark Barrett, and David L Dill, "Ground-truth adversarial examples," *arXiv preprint arXiv:1709.10207*, 2017. 2

[17] Rudy Bunel, Ilker Turkaslan, Philip HS Torr, Pushmeet Kohli, and M Pawan Kumar, "Piecewise linear neural network verification: a comparative study," *arXiv preprint arXiv:1711.00455*, 2017. 2

[18] Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess, "Maximum resilience of artificial neural networks," in *International Symposium on Automated Technology for Verification and Analysis*. Springer, 2017, pp. 251–268. 2

[19] Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning, and Inderjit Dhillon, "Towards fast computation of certified robustness for relu networks," in *International Conference on Machine Learning*, 2018, pp. 5273–5282. 2

[20] Matthew Mirman, Timon Gehr, and Martin Vechev, "Differentiable abstract interpretation for provably robust neural networks," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, 2018, pp. 3578–3586. 2, 3, 8, 13, 14

[21] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev, "Ai 2: Safety and robustness certification of neural networks with abstract interpretation," in *IEEE Symposium on Security and Privacy*, 2018.

[22] Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy A Mann, and Pushmeet Kohli, "A dual approach to scalable verification of deep networks." in *UAI*, 2018, pp. 550–559. 3

[23] Krishnamurthy Dvijotham, Sven Gowal, Robert Stanforth, Relja Arandjelovic, Brendan O'Donoghue, Jonathan Uesato, and Pushmeet Kohli, "Training verified learners with learned verifiers," *arXiv preprint arXiv:1805.10265*, 2018. 2, 5, 7, 8, 13

[24] Eric Wong and Zico Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *International Conference on Machine Learning*, 2018, pp. 5283–5292. 2, 5, 7, 8

[25] Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J Zico Kolter, "Scaling provable adversarial defenses," in *Advances in Neural Information Processing Systems*, 2018, pp. 8400–8409. 2, 5, 6, 8, 11, 12, 14

[26] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana, "Formal security analysis of neural networks using symbolic intervals," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 2018, pp. 1599–1614. 2

[27] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang, "Certified defenses against adversarial examples," in *International Conference on Learning Representations*, 2018. 2, 5

[28] Teruo Sunaga, "Theory of interval algebra and its application to numerical analysis," *RAAG memoirs*, vol. 2, no. 29-46, p. 209, 1958. 2

[29] Kai Y. Xiao, Vincent Tjeng, Nur Muhammad (Mahi) Shafiullah, and Aleksander Madry, "Training for faster adversarial robustness verification via inducing reLU stability," in *Inter-*

national Conference on Learning Representations, 2019. 2, 7, 8

[30] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 4780–4789. 7

[31] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014. 11

[32] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard *et al.*, "Tensorflow: a system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283. 11

[33] Shiqi Wang, Yizheng Chen, Ahmed Abdou, and Suman Jana, "Mixtrain: Scalable training of formally robust neural networks," *arXiv preprint arXiv:1811.02625*, 2018. 14

Scalable Verified Training for Provably Robust Image Classification (Supplementary Material)

A. Training parameters

For IBP, across all datasets, the networks were trained using the Adam [31] algorithm with an initial learning rate of 10^{-3}. We linearly ramp-down the value of κ between 1 and κ_{final} after a fixed warm-up period (κ_{final} is set to both 0 or 0.5 and the best result is used). Simultaneously, we lineary ramp-up the value of ϵ between 0 and ϵ_{train} (for CIFAR-10 and SVHN, we use a value of ϵ_{train} that is 10% higher than the desired robustness radius). MNIST is trained on a single Nvidia V100 GPU. CIFAR-10, SVHN and IMAGENET are trained on 32 tensor processing units (TPU) [32] with 2 workers with 16 TPU cores each.

- For MNIST, we train on a single Nvidia V100 GPU for 100 epochs with batch sizes of 100. The total number of training steps is 60K. We decay the learning rate by $10\times$ at steps 15K and 25K. We use warm-up and ramp-up durations of 2K and 10K steps, respectively. We do not use any data augmentation techniques and use full 28×28 images without any normalization.
- CIFAR-10, we train for 3200 epochs with batch sizes of 1600 (training for 350 epochs with batch sizes of 50 reaches 71.70% verified error rate when $\epsilon = 8/255$). The total number of training steps is 100K. We decay the learning rate by $10\times$ at steps 60K and 90K. We use warm-up and ramp-up durations of 5K and 50K steps, respectively. During training, we add random translations and flips, and normalize each image channel (using the channel statistics from the train set).
- For SVHN, we train for 2200 epochs with batch sizes of 50 (training for 240 epochs with batch sizes of 50 reaches within 1% of the verified error rate). The total number of training steps is 100K. The rest of the schedule is identical to CIFAR-10. During training, we add random translations, and normalize each image channel (using the channel statistics from the train set).
- For IMAGENET, we train for 160 epochs with batch sizes of 1024. The total number of training steps is 200K. We decay the learning rate by $10\times$ at steps 120K and 180K. We use warm-up and ramp-up durations of 10K and 100K steps, respectively. We use images downscaled to 64×64 (resampled using pixel area relation, which gives moiré-free images). During training, we use random crops of 56×56 and random flips. During testing, we use a central 56×56 crop. We also normalize each image channel (using the channel statistics from the train set).

The networks trained using Wong et al. [25] were trained using the schedule and learning rate proposed by the authors. For Madry et al. [9], we used a learning rate schedule identical to IBP and, for the inner optimization, adversarial examples are generated by 7 steps of PGD with Adam [31] and a learning rate of 10^{-1}. Note that our reported results for these two methods closely match or beat published results, giving us confidence that we performed a fair comparison.

Figure 5 shows how the empirical PGD accuracy (on the test set) increases as training progresses for IBP and Madry et al.. This plot shows the median performance (along with the 25$^{\text{th}}$ and 75$^{\text{th}}$ percentiles across 10 independent training processes) and confirms that IBP is stable and produces consistent results. Additionally, for IBP, we clearly see the effect of ramping the value of ϵ up during training (which happens between steps 2K and 12K).

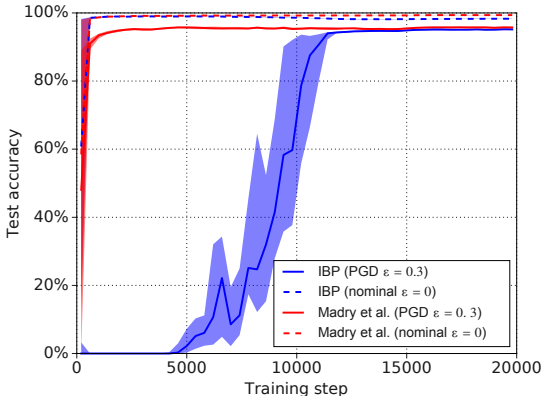


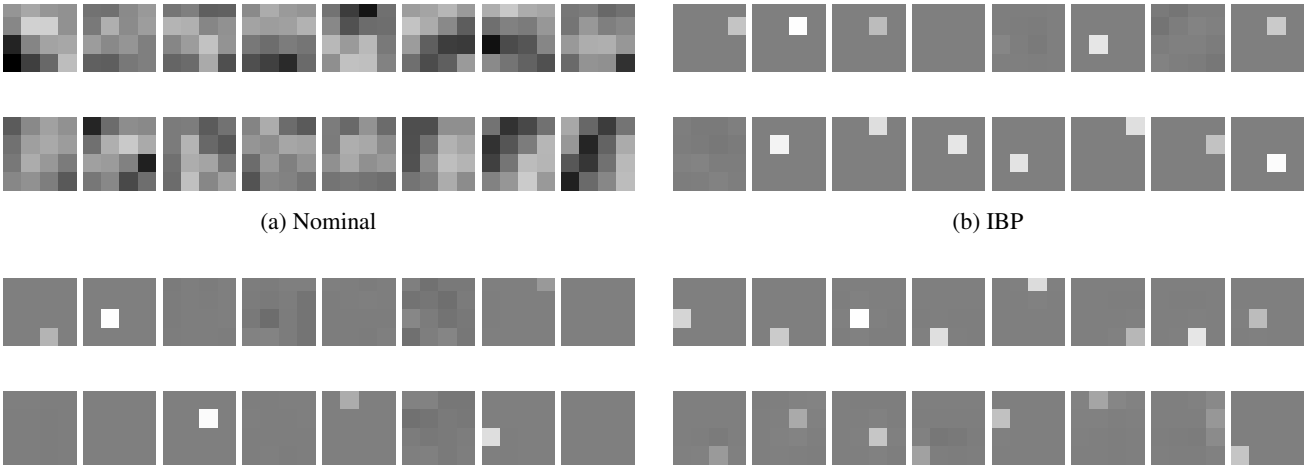
Figure 5: Median evolution of the nominal (no attacks) and empirical PGD accuracy (under perturbations of $\epsilon = 0.3$) as training progresses for 10 independently trained large models on MNIST. The shaded areas indicate the 25$^{\text{th}}$ and 75$^{\text{th}}$ percentiles.

B. Convolutional filters

Figure 6 shows the first layer convolutional filters resulting from training a small robust model on MNIST against a perturbation radius of $\epsilon = 0.1$. Overall, the filters tend to be extremely sparse – at least when compared to the filters obtained by training a nominal non-robust model (this observation is consistent with [25]). We can qualitatively observe that Wong et al. produces the sparsest set of filters.

Similarly, as shown in Figure 7, robust models trained on CIFAR-10 exhibit high levels of sparsity in their convolutional filters. Madry et al. seems to produce more meaningful filters, but they remain sparse compared to the non-robust model.

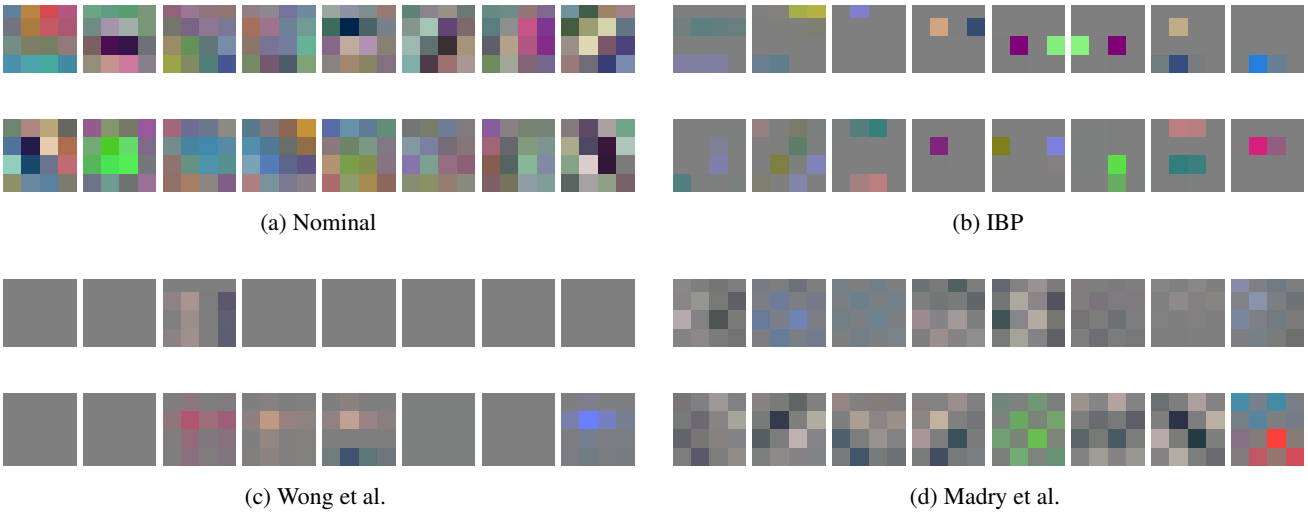
This analysis suggests that IBP strongly limits the capacity of the underlying network. As a consequence, larger models are often preferable. Larger models, however, can lead to the explosion of intermediate interval bounds – and this is the main reason why values of ϵ must be carefully scheduled. Techniques that combine tighter (but slower) relaxations with IBP could be used in the initial training stages when training deeper and wider models.



(a) Nominal (b) IBP

(c) Wong et al. (d) Madry et al.

Figure 6: First layer convolutional filters resulting from training a small robust model on MNIST against a perturbation radius of $\epsilon = 0.1$ for all methods.



(a) Nominal (b) IBP

(c) Wong et al. (d) Madry et al.

Figure 7: First layer convolutional filters resulting from training a small robust model on CIFAR-10 against a perturbation radius of $\epsilon = 2/255$ for all methods.

C. Ablation study

Table 5 reports the performance of each model for different training methods on MNIST with $\epsilon = 0.4$ (averaged over 10 independent training processes). We chose MNIST for ease of experimentation and a large perturbation radius to stress the limit of IBP. The table shows the effect of the elision of the last layer and the cross-entropy loss independently. We do not report results without the schedule on ϵ as all models without the schedule are unable to train (i.e., reaching only 11.35% accuracy at best).

We observe that all components contribute to obtaining a good final verified error rate. This is particularly visible for the small model, where adding elision improves the bound by 3.88% (12.9% relative improvement) when using a softplus loss. Using cross-entropy (instead of softplus) also results in a significant improvement of 4.42% (14.7% relative improvement). Ultimately, the combination of cross-entropy and elision shows an improvement of 5.15%. Although present for larger models, these improvements are less visible. This is another example of how models with larger capacity are able to adapt to get tight bounds. Finally, it is worth noting that elision tends to provide results that are more consistent (as shown by the 25th and 75th percentiles).

D. Additional results

Table 6 provides complementary results to the ones in Table 4. It includes results for each individual model architecture, as well as results from the literature (for matching model architectures). The test error corresponds to the test set error rate when there is no adversarial perturbation. For models that we trained ourselves, the PGD error rate is calculated using 200 iterations of PGD and 10 random restarts. The verified bound on the error rate is obtained using the MIP/LP cascade described in Section 4.

We observe that with the exception of CIFAR-10 with $\epsilon = 2/255$, IBP outperforms all other models by a significant margin (even for equivalent model sizes). For CIFAR-10 with $\epsilon = 2/255$, it was important to increase the model size to obtain competitive results. However, since IBP is cheap to run (i.e., only two additional passes), we can afford to run it on much larger models.

E. Runtime

When training the small network on MNIST with a Titan Xp GPU (where standard training takes 1.5 seconds per epoch), IBP only takes 3.5 seconds per epoch compared to 8.5 seconds for Madry et al. and 2 minutes for Wong et al. (using random projection of 50 dimensions). Indeed, as detailed in Section 3 (under the paragraph "interval bound propagation"), IBP creates only two additional passes through the network compared to Madry et al. for which we used seven PGD steps during training. Xiao et al.'s method takes the same amount of time as IBP as it needs to perform bound propagation too.

Model	**Training method**	**IBP verified bound**		
		median (0.5)	0.25	0.75
small	+ ϵ-schedule	30.09%	28.63%	32.03%
	+ elision	26.21%	24.99%	28.49%
	+ cross-entropy	25.67%	24.04%	26.71%
	+ elision	**24.94%**	**23.10%**	**26.63%**
medium	+ ϵ-schedule	20.24%	19.77%	20.66%
	+ elision	19.96%	19.92%	20.19%
	+ cross-entropy	18.10%	**17.60%**	18.38%
	+ elision	**18.02%**	17.61%	**18.11%**
large	+ ϵ-schedule	18.32%	17.88%	18.55%
	+ elision	18.03%	17.93%	18.58%
	+ cross-entropy	16.39%	**15.73%**	17.11%
	+ elision	**16.33%**	16.00%	**16.74%**

Table 5: **Ablation study.** This table compares the median verified bound on the error rate (obtained using IBP) for different training methods on MNIST with $\epsilon = 0.4$. The reported IBP verified error bound is an upper bound of the true verified error rate; it is computed using the elision of the last layer. The different training methods include combining the linear schedule on ϵ_{train} (as explained in Appendix A) with the elision of the last layer or the cross-entropy loss (the cross-entropy loss is replaced with a softplus loss otherwise, as done by Mirman et al. [20] and Dvijotham et al. [23]). We do not report results without the ϵ-schedule as all models without the schedule are unable to train (i.e., reaching only 11.35% accuracy at best).

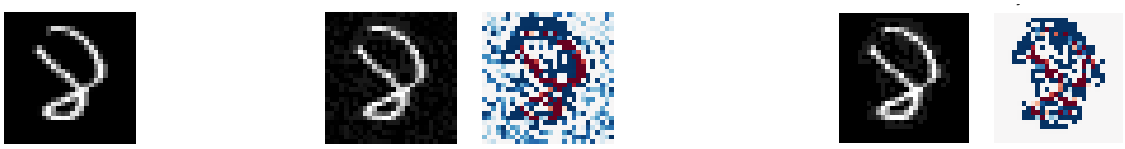
Dataset	Epsilon	Method	Test error	PGD	Verified
MNIST	$\epsilon = 0.1$	IBP (small)	1.39%	2.91%	**2.97%**
		IBP (medium)	1.06%	2.11%	**2.23%**
		IBP (large)	1.07%	1.89%	2.32%
		Reported in literature			
		Wong et al. [25] (small)	1.26%	–	4.48%
		Wong et al. [25] (medium)	1.08%	–	3.67%
		Mirman et al. [20]* (small)	2.4%	4.4%	5.8%
		Mirman et al. [20] (medium)	1.0%	2.4%	3.4%
		Wang et al. [33]** (small)	1.5%	3.7%	8.4%
		Wang et al. [33] (medium)	0.5%	1.8%	4.8%
MNIST	$\epsilon = 0.3$	IBP (small)	4.59%	9.09%	**10.25%**
		IBP (medium)	2.70%	7.98%	**9.74%**
		IBP (large)	1.66%	6.12%	8.05%
		Reported in literature			
		Wong et al. [25] (small)	14.87%	–	43.10%
		Wong et al. [25] (medium)	12.61%	–	45.66%
		Mirman et al. [20] (small)	3.2%	9.0%	19.4%
		Mirman et al. [20] (medium)	3.4%	6.2%	18.0%
		Wang et al. [33] (small)	4.6%	12.7%	48%
		Wang et al. [33] (medium)	3.4%	10.6%	41.6%
CIFAR-10	$\epsilon \approx 1/510$***	**Reported in literature**			
		Mirman et al. [20]*** (small)	42.8%	46.4%	47.8%
		Mirman et al. [20] (medium)	52.8%	59.2%	64.2%
CIFAR-10	$\epsilon \approx 2/255$***(*)	IBP (small)	39.54%	53.95%	56.43%
		IBP (medium)	37.28%	51.73%	54.78%
		IBP (large)	29.84%	45.09%	49.98%
		Reported in literature			
		Wong et al. [25] (small)	38.91%	–	**52.75%**
		Wong et al. [25] (medium)	31.28%	–	**46.59%**
		Mirman et al. [20]*** (small)	45.8%	60.0%	64.8%
		Mirman et al. [20] (medium)	51.6%	61.4%	75.8%
		Wang et al. [33]**** (small)	28.9%	45.4%	62.4%
		Wang et al. [33] (medium)	22.1%	36.5%	58.4%
CIFAR-10	$\epsilon = 8/255$	IBP (small)	61.63%	70.42%	**72.93%**
		IBP (medium)	58.23%	69.72%	**72.33%**
		IBP (large)	50.51%	65.23%	67.96%
		Reported in literature			
		Wong et al. [25] (small)	72.24%	–	79.25%
		Wong et al. [25] (medium)	80.56%	–	83.43%

Table 6: **Additional comparison with the state-of-the-art.** Comparison of the nominal test error (no adversarial perturbation), error rate under PGD attacks, and verified bound on the error rate. For IBP models, the PGD error rate is calculated using 200 iterations of PGD and 10 random restarts. For reported results, we report the closest setting to ours. We indicate in parenthesis the model architecture: for Wong et al. [25] small and medium are equivalent to "Small" and "Large"; for Mirman et al. [20] they are equivalent to the best of "ConvSmall/ConvMed" and best of "ConvBig/ConvSuper"; for Wang et al. [33] they are equivalent to "Small" and "Large" (we always report the best verified result when different verification methods are used). We indicate in bold the best results per model architecture.
* Mirman et al. [20] report results on 500 samples for both MNIST and CIFAR-10 (instead of the 10K samples in test set).
** The number of samples used in Wang et al. [33] is unknown.
*** We confirmed with Mirman et al. that [20] uses a perturbation radius of 0.007 post-normalization, it is roughly equivalent to 1/510 pre-normalization. Similarly they use a perturbation radius of 0.03 post-normalization, it is roughly equivalent to 2/255 pre-normalization.
**** We confirmed with Wang et al. that [33] use a perturbation radius of 0.0348 post-normalization, it is roughly equivalent to 2/255 pre-normalization.

F. When Projected Gradient Descent is not enough

For a given example in MNIST, this section compares the worst-case attack found by PGD with the one found using a complete solver. The underlying model is a medium sized network trained using IBP with $\epsilon = 0.1$. The nominal image, visible in Figure 8a, has the label "eight", and corresponds to the 1365^{th} image of the test set.

The worst-case perturbation of size $\epsilon = 0.1$ found using 200 PGD iterations and 10 random restarts is shown in Figure 8b. In this particular case, the robust network is still able to successfully classify the attack as an "eight". Without any verifiable proof, we could wrongly assume that our network is robust to ℓ_∞ perturbation on that image. However, when running a complete solver (using a MIP formulation), we are able to find a counter-example that successfully induces the model to misclassify the "eight" as a "two" (as shown in Figure 8c).



(a) Nominal image correctly classified as an "eight"

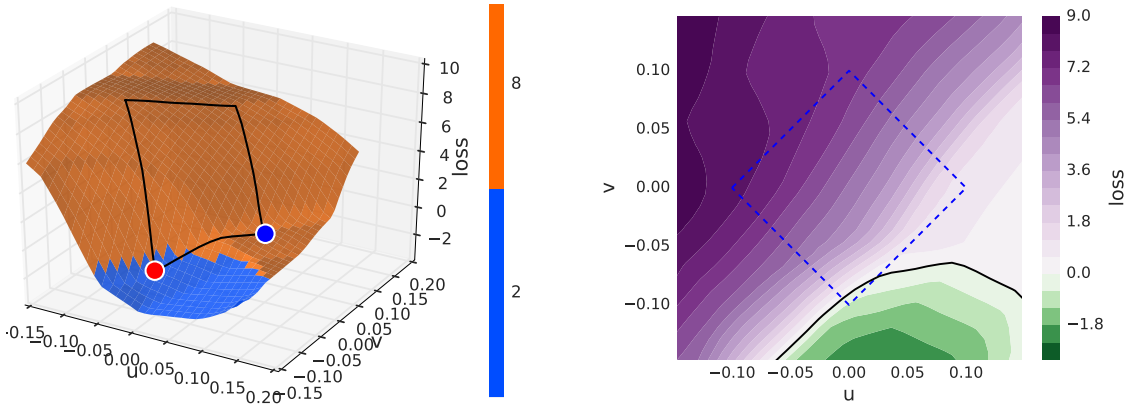
(b) Worst attack found using PGD still classified as an "eight"

(c) Actual worst attack found using a MIP solver incorrectly classified as a "two"

Figure 8: Attacks of size $\epsilon = 0.1$ found on the 1365^{th} image of the MNIST test set. For (b) and (c), the left pane shows the adversarial image, while the right pane shows the perturbation rescaled for clarity.

Figure 9 shows the untargeted adversarial loss (optimized by PGD) around the nominal image. In these loss landscapes, we vary the input along a linear space defined by the worse perturbations found by PGD and the MIP solver. The u and v axes represent the magnitude of the perturbation added in each of these directions respectively and the z axis represents the loss. Typical cases where PGD is not optimal are often a combination of two factors that are qualitatively visible in this figure:

- We can observe that the MIP attack only exists in a corner of the projected ℓ_∞-bounded ball around the nominal image. Indeed, since PGD is a gradient-based method, it relies on taking gradient steps of a given magnitude (that depends on the learning rate) at each iteration. That is, unless we allow the learning rate to decay to a sufficiently small value, the reprojection on the norm-bounded ball at each iteration will force the PGD solution to bounce between the edges of that ball without hitting its corner.
- The second, more subtle, effect concerns the gradient direction. Figure 9b, which shows a top-view of the loss landscape, indicates that a large portion of ℓ_∞ ball around the nominal image pushes the PGD solution towards the right (rather than the bottom). In other words, gradients cannot always be trusted to point towards the true worst-case attack.



(a) (b)

Figure 9: Loss landscapes around the nominal image of an "eight". It is generated by varying the input to the model, starting from the original input image toward either the worst attack found using PGD (u direction) or the one found using a complete solver (v direction). In (a), the z axis represents the loss and the orange and blue colors on the surface represent the classification predicted by the model. We observe that while the PGD attack (blue dot) is correctly classified as an "eight", the MIP attack (red dot) is misclassified as a "two". Panel (b) shows a top-view of the same landscape with the decision boundary in black. For both panels, the diamond-shape represents the projected ℓ_∞ ball of size $\epsilon = 0.1$ around the nominal image.